Detecting Wireless Spy Cameras Via Stimulating and Probing

Tian Liu
Center for Energy-efficient Computing and Applications
Peking University

Ziyu Liu
Center for Energy-efficient Computing and Applications
Peking University

Jun Huang*
Center for Energy-efficient Computing and Applications
Peking University

Rui Tan
School of Computer Science and Engineering
Nanyang Technological University

Zhen Tan
Center for Energy-efficient Computing and Applications
Peking University

ABSTRACT

The rapid proliferation of wireless video cameras has raised serious privacy concerns. In this paper, we propose a *stimulating-and-probing* approach to detecting wireless spy cameras. The core idea is to actively alter the light condition of a private space to manipulate the spy camera's video scene, and then investigates the responsive variations of a packet flow to determine if it is produced by a wireless camera. Following this approach, we develop Blink and Flicker – two practical systems for detecting wireless spy cameras. Blink is a lightweight app that can be deployed on off-the-shelf mobile devices. It asks the user to turn on/off the light of her private space, and then uses the light sensor and the wireless radio of the mobile device to identify the response of wireless cameras. Flicker is a robust and automated system that augments Blink to detect wireless cameras in both live and offline streaming modes. Flicker employs a cheap and portable circuit, which harnesses daily used LEDs to stimulate wireless cameras using human-invisible flickering. The time series of stimuli is further encoded using FEC to combat ambient light and uncontrollable packet flow variations that may degrade detection performance. Extensive experiments show that Blink and Flicker can accurately detect wireless cameras under a wide range of network and environmental conditions.

CCS CONCEPTS

• **Security and privacy → Security services**; • **Human-centered computing → Ubiquitous and mobile computing**;

ACM Reference Format:
Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting Wireless Spy Cameras Via Stimulating and Probing. In *MobiSys '18: The 16th Annual International Conference on Mobile Systems, Applications, and Services, June 10–15, 2018, Munich, Germany*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3210240.3210332

*The first two authors contributed equally. The corresponding author is Jun Huang (jun.huang@pku.edu.cn).

MobiSys '18, June 10–15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5720-3/18/06…$15.00
https://doi.org/10.1145/3210240.3210332

Figure 1: Examples of commodity wireless spy cameras. The arrows point to the positions of camera lens. The labeled prices are quoted from an online retailer in April, 2018 [4].

1 INTRODUCTION

With the rapid proliferation of wireless video cameras, the technical bar and the economical cost of invading privacy using spy cameras have been significantly reduced, which has raised serious concerns. As an example of responses to such concerns, Airbnb prohibits the use of surveillance devices in all of its rental apartments [11]. Unfortunately, it is hard to enforce the compliance to such rules because of the difficulties in detecting wireless spy cameras [17]. In particular, without relying on cables for network connectivity, wireless spy cameras can be easily installed anywhere in a private space, or be implanted into daily used objects like smoke detectors, USB chargers, clocks, or power outlets (as shown in Fig. 1). Today, wireless spy cameras are widely available on the market and can be easily obtained from online retailers at around a hundred dollars [4].

Despite the increasing privacy concerns, to date, detecting wireless spy cameras has been an open problem of significant challenges. Traditional traffic analysis tools [12, 21, 26, 30, 32] identify video/audio streams based on traffic classification. However, considering the explosive growth and the ubiquitous existence of video traffic (*e.g.*, the traffics of video chatting, smart TV, or legal surveillance cameras deployed in neighborhood), the detection of video traffic does not reliably assert the presence of a wireless spy camera [15]. Conventional camera detectors [6, 8] assist users to search for the glint of camera lens or the RF signals emitted from the camera's wireless radio. However, the user has to scrutinize the entire private space, which requires a slow and meticulous sweep while providing no assurance for detection. Lacking the ability of identifying the signal of the wireless spy camera, existing RF-based camera countermeasures rely on indiscriminate jamming to block the entire wireless medium [3, 7], which inevitably degrades

the performance of legitimate network devices. To protect a scene from photographing, LiShield [33] enforces visual privacy protection by jamming the visible light spectrum. It degrades video quality by imposing a striped watermark on the images captured by the camera, but cannot black out the entire scene.

In this paper, we propose to detect wireless spy cameras by identifying their packet flows via *stimulating-and-probing*. This approach is motivated by two observations. First, in private spaces like bedrooms and bathrooms, the user usually has complete control of the light condition. Second, as the de-facto standard, wireless cameras encode the video in variable bitrate (VBR), where the sizes of produced pictures fluctuate with the variation of video scenes. Therefore, it is possible to actively alter the light condition of the user's private space to manipulate the video scene, and then investigates the responsive variation of a packet flow to determine if it is produced by a wireless camera that is spying on the user's space.

Realizing the stimulating-and-probing approach entails several key technical challenges. First, during daytime, the effect of light stimuli can be degraded in the presence of strong ambient light. Moreover, because of IP fragmentation where a picture larger than the maximum transmission unit (MTU) is split into multiple packets before transmission, it is difficult to measure the picture sizes of a video stream without inspecting the picture metadata, which is typically encrypted. Although the detector may estimate the data rate of a packet flow to infer the variation of picture sizes, the rate measurements are often highly noisy due to packet losses and jitters, as well as uncontrollable environmental variations that may cause picture sizes to fluctuate even without light stimuli.

In this paper, we tackle these challenges in the design and implementation of Blink and Flicker – two practical systems that can accurately identify the packet flows of wireless spy cameras.

Blink is a lightweight app that can be deployed on off-the-shelf mobile devices to detect wireless spy cameras operated in live streaming mode. Due to the lack of local storage, most miniature and implanted wireless spy cameras transfer pictures immediately once they are captured, therefore will respond to light stimuli in real-time. Motivated by this observation, Blink asks the user to turn on/off the light of her private space, and then uses the light sensor and the wireless radio of the mobile device to examine the responses of packet flows at the time instants of light stimuli. To improve accuracy and robustness, Blink combines the probing results using a statistical tool after multiple-rounds of detection.

Flicker is a robust and automated system that augments Blink to detect wireless spy cameras operated in both live and offline streaming modes. Flicker employs a cheap and portable circuit, which harnesses daily used LEDs to stimulate wireless spy cameras using human invisible light flickering. By controlling the on/off of light stimuli, Flicker modulates the picture sizes of the wireless spy camera to embed an identification code. In the probing phase, Flicker searches for the identification code by decoding the data rate variations of overheard packet flows. To combat strong ambient light, Flicker protects the embedded identification code using forward error correction (FEC), and adapts the FEC coding rate based on ambient light conditions. To further improve robustness, Flicker employs a probabilistic detector, which computes a detection score based on probabilistic models to quantify how likely a wireless spy camera exists even if some of the embedded identification codes are erased by disruptive network or environmental variations.

We present the prototype implementations of Blink and Flicker, and conduct extensive experiments to evaluate their performance using commodity wireless cameras. The results show that our systems can accurately detect wireless cameras in a wide range of settings, including different ambient light intensities, video streaming modes and configurations (*e.g.*, resolution, frame rates), wireless interference conditions, and uncontrollable environmental variations.

2 SPY CAMERA MODEL

Our objective is to detect off-the-shelf wireless cameras that are compliant with existing wireless and video coding standards. We note that specially customized spying tools that use proprietary protocols or coding algorithms are often costly, difficult to deploy, and much less accessible. In this section, we specify the model of target wireless spy cameras from four aspects, including video streaming mode, video compression scheme, wireless standard, and packet encryption.

Video streaming. Commodity wireless video cameras typically support two streaming modes, i.e., *live streaming* and *progressive downloading*. For most miniature and implanted wireless cameras that have no local storage, the recorded video is live streamed to the receiver (e.g., a remote video player or a cloud storage), where each picture is transferred immediately once captured. In progressive downloading, the recorded video is first stored locally, and then transferred to the player upon request. During downloading, the video is transferred block by block, which allows the client to play the video before the download is complete.

Video compression. As the de-facto standard, wireless cameras encode videos using *variable bitrate* (VBR), where the coding rate is dynamically adjusted based on the complexity of scene. After encoding, the video is further compressed to save bandwidth. One of the most popular video compression scheme is *motion compensation* [1, 2, 9], which compresses video based on the transformation of scene. For instance, in H.264 [2], pictures are classified into three categories including I-, P-, and B-pictures. Video data is compressed based on group of pictures (GoP). Each GoP consists of one I-picture and multiple P- and B-pictures. The I-picture independently encodes a complete scene, which serves as the reference point for other pictures in the same GoP. P- and B-pictures only encode the difference between scenes, such that their sizes are much smaller than that of the I-picture.

Wireless standard. In this paper, we focus on detecting wireless cameras that transfer videos using Wi-Fi [10] – one of the most prevalent wireless access technologies. In general, our approach works for cameras that use other wireless technologies, as long as the detector has a compliant wireless radio to sniff the camera's traffic.

Packet encryption. As a common practice, wireless cameras encrypt their packet payloads to prevent eavesdropping. Our approach does not require the detector to decrypt and inspect video data. Instead, it only decodes the sender and receiver addresses from the

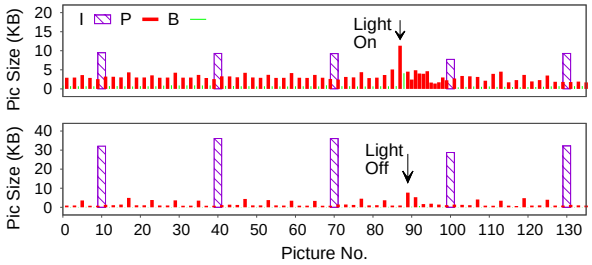


Figure 2: Picture size measured at night (top) and day (bottom). P- and B- picture sizes increase at the time instant of light condition change.

MAC-layer header of each overheard packet, which allows the detector to divide network traffic into packet flows and then measure the data rate variation of each flow. Fortunately, in all major wireless standards like 802.11 [10] and LTE [5], the MAC-layer header is encryption-free.

3 UNDERSTANDING CAMERA TRAFFIC

In this section, we present a measurement-based study to characterize the traffic of wireless cameras that use H.264 [2] – a representative motion compensation-based video compression algorithm. To understand how wireless spy cameras respond to light stimuli, we instrument a media player to measure the variation of picture size when the light condition changes. Then, we investigate the response behaviors from the perspective of a detector, by measuring the data rate variation under different video streaming modes, and then understanding the probing challenges caused by uncontrollable packet flow variations.

3.1 Response to Light Stimuli

The measurements are conducted in a room both during daytime and at night with significantly different ambient light conditions. To generate light stimuli, the ceiling light of the room is turned on and off to change the camera's video scene.

Fig. 2 shows the trace of picture size before and after changing light condition. The video is streamed at about 15 frames per second (fps), where each GoP contains 30 pictures. We observe that the picture sizes are similar no matter the light is on or off as long as the video scene remains relatively stable. In contrast, at the exact time instants of the light stimuli, the video encoder devotes a large number of bits to encoding the transformation of scene, which causes a significant increase of the P- and B- picture sizes.

Fig. 3 compares the picture size under light stimuli with that of stable scenes for video streams of different resolutions. To identify the pictures that are subject to light stimuli, we replay the recorded video frame by frame to locate the pictures that record the light condition change. As shown in Fig. 3, the impact of light stimuli on P- and B- pictures are significant even in daytime. For instance, for the 640x480 video stream, the average size of P-pictures increases by 6.3 times from 0.3 KB to 1.9 KB. This effect is even more pronounced at night. For instance, for the 1280x728 video stream, the size of P-pictures is only 1.1 KB to 3.6 KB when the scene is stable, and is 5.3 KB to 10.9 KB at the time instants of light stimuli.

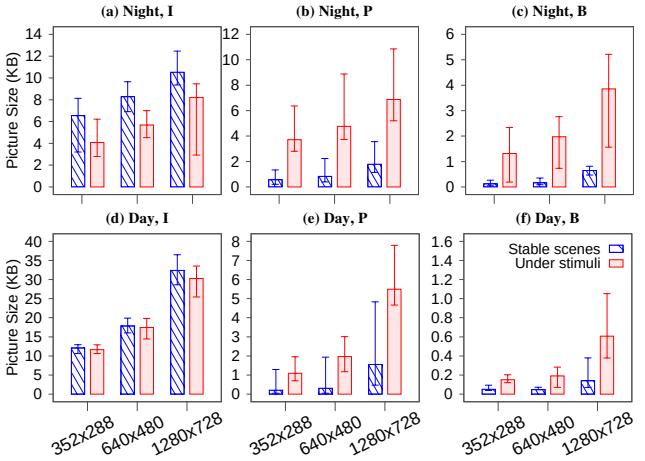


Figure 3: The impact of light stimuli on the picture size of video streams with different resolutions.

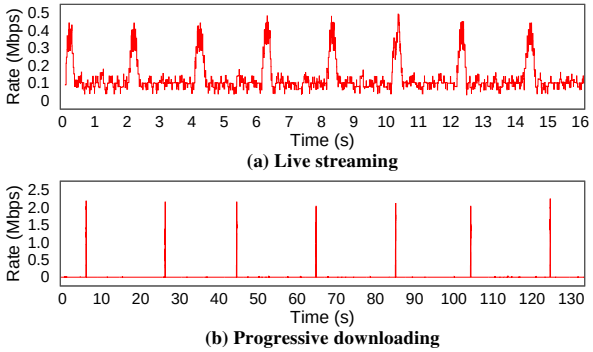


Figure 4: Data rate measurements of the camera's traffic under different streaming modes.

3.2 Data Rate Patterns

Because of IP fragmentation and packet encryption, it is often difficult to directly measure the picture sizes of a video stream without inspecting application-layer video traffic. Instead, the detector may estimate the data rate of a packet flow to infer the variation of picture size. Fig. 4 plots the trace of data rates measured for two video streams under different streaming modes. We sniff the wireless camera's packet flow and then estimate its data rate by applying a moving window of 100 ms. During live streaming, the transmissions of I-pictures produce periodic peaks, where each period corresponds to one GoP. Therefore, it is possible to estimate the size of each picture by probing the internal structure of each period. During progressive downloading, the video is transferred block by block periodically, where each block contains a video segment that lasts for about 20 seconds. In practice, the duration of each block is determined by the available memory space at the video player, and is typically 20 to 30 seconds long in multimedia applications. In comparison, the duration of GoP is typically one to two seconds. Because one block may contain multiple GoPs, it is impossible to use data rate measurements to estimate picture size.

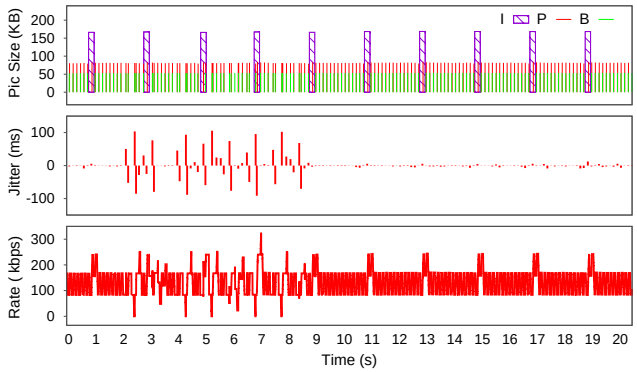


Figure 5: Picture size (top), jitter (middle), and rate (bottom) measured for a live video stream. Rate measurements vary in the presence of jitters even when the scene remains stable.

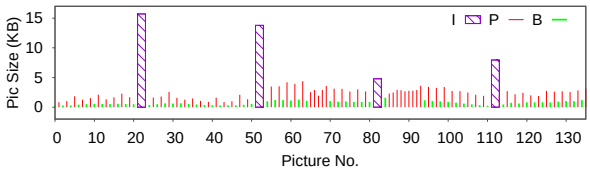


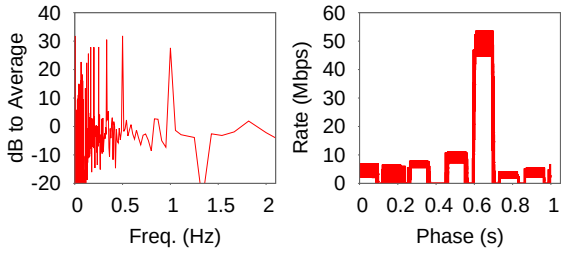
Figure 6: Picture size variation as people moving close to the camera from a distance of five meters.

3.3 Probing Challenges

The key of detecting a wireless camera is to identify the data rate variation of its packet flow under light stimuli. One factor that makes this challenging is the strong ambient light during daytime, which may degrade the effect of light stimuli. In addition, the data rate measurements might be interfered with packet flow variations caused by network and environmental factors.

First, in wireless networks, rate measurements can be polluted by various noise sources such as packet losses and jitters. Fig. 5 shows the picture sizes, jitters, and data rates measured for a live video stream. The jitter is calculated as the deviation of picture interval from the true picture periodicity. In the presence of jitters, packet transmissions suffer random dispersion and clumping, leading to significant data rate variation when measured at the detector. For example, in Fig. 5, we observe a burst of jitters ranging from −100 ms to +100 ms starting from the 2nd second to the 9th second. During this interval, data rate measurements vary significantly despite that the video scene remains relatively stable (as shown in the trace of picture size). This will cause strong interference in the probing phase. Furthermore, When there exist uncontrollable scene variations, the picture size itself can be misleading. For example, Fig. 6 plots the trace of picture size when a person moves into the video scene at the 50th second from a distance of about 5 m. The size of P- and B-pictures increases as the person moves close. In particular, when the person is moving at a distance of about 2 m, the average P-picture size increases to 2.9 KB, which is about 7.2 times of that when the scene keeps stable.

In the following sections, we will discuss how to address these challenges in the design of Blink and Flicker.



(a) PSD. **(b) Rate profile.**

Figure 7: The power spectral density and rate profile for a video stream where the GoP frequency is 1Hz. The error bars in the rate profile indicate max and min.

4 BLINK: THE DETECTOR AS AN APP

Blink is a lightweight mobile app that can detect wireless spy cameras operated under live streaming mode. During detection, Blink first profiles the data rate variations of packet flows and then probes their responses to user-triggered light stimuli. The detection results obtained in multiple rounds of light stimuli are combined to improve accuracy and robustness. In this section, we present the design of Blink in detail.

4.1 Packet Flow Profiling

In this stage, Blink first scans wireless channels to discover surrounding wireless networks, and then profiles the data rate variations for all overheard packet flows. Based on profiling results, it identifies a set of suspicious packet flows for further probing. If a wireless spy camera exists, Blink will profile the camera's responses to uncontrollable environmental stimuli such as curtain blowing or plant movements in the wind. It also learns the camera's data rate variations caused by network factors like packet jitters. The profiled data rate variations will be treated as noise floors when examining the responses of packet flows during probing.

Rate measurement. For each packet flow, Blink applies a sliding window to estimate its data rate. Because of IP fragmentation, a picture that is larger than MTU will be divided into multiple packets. To avoid underestimating the picture size, the sliding window must be large enough to accommodate all fragmented packets of the same picture. In practice, the frame rate of commodity cameras is typically higher than 10 fps to satisfy the flicker fusion threshold of human eyes. Based on this observation, Blink sets the size of the sliding window to 100 s.

Blink may underestimate the data rate in the presence of packet losses. To address this issue, Blink keeps monitoring the packet sequence numbers in MAC headers. The estimated data rate is then rectified using the number of lost packets identified from the sequence numbers, multiplied with the average packet size estimated from a small set of recently received packets.

Rate profiling. In live streaming mode, the packet flows of wireless spy cameras manifest strong periodicities, where each period corresponds to one GoP. The goal of rate profiling is to learn the periodic data rate patterns of each packet flow, which allows Blink to characterize the GoP structure of the wireless spy camera.

To measure the period of a packet flow, Blink computes the fast Fourier transform (FFT) of its data rate series to obtain the normalized power spectral density (PSD). The estimated PSD can be noisy in the presence of jitters. For example, Fig. 7(a) shows the PSD of a video stream with a GoP frequency of 1 Hz. Because of noise, the PSD peak occurs at the second harmonic of the true GoP frequency, leading to a wrong estimation of period. Blink addresses this issue using a two-step heuristic. First, Blink identifies the frequency f of the PSD peak, and then computes its integer factorization to obtain a list of candidate frequencies. Second, Blink evaluates all candidate frequencies in an ascending order. A candidate is identified as the fundamental frequency if the PSD values of all its harmonics until f are larger than a pre-defined threshold. In Blink, the threshold is set to 20 dB-to-average based on empirical measurements.

After obtaining the period of a packet flow, Blink folds the data rate measurements at the boundaries of the periods and then profiles the periodic pattern. As an example, Fig. 7(b) plots the data rate profile established based on data rate measurements of 20 s. As shown in Fig. 7(b), the periodic transmission of I-pictures yields the highest peak at the phase around 0.6 s. We can also observe lower peaks corresponding to the P- and B- pictures.

Suspect flow identification. A packet flow is considered suspicious if it has a peak in its data rate profile, which resembles the structure of a typical GoP where the data rate peak corresponds to the I-picture. To characterize how salient the peak is, Blink computes the Crest factor as follows,

$$C_{\text{dB}} = 20\log_{10}\frac{\mathsf{r}_{\text{peak}}}{\mathsf{r}_{\text{rms}}},$$

where r_{peak} is the peak data rate and r_{rms} is the root mean square of the data rate profile. Because the size of I-picture is always larger than that of non-key pictures in the same GoP, the fewer the number of non-key pictures, the larger the Crest factor. When the GoP contains only one non-key picture, we have the minimum of the Crest factor, which can be derived as,

$$C_{\text{dB}} = 20\log_{10}\frac{s_{\text{I}}}{\sqrt{\frac{1}{2}(s_{\text{I}}^2 + s_{\text{non key}}^2)}} = 20\log_{10}\frac{4\gamma}{\sqrt{1+\gamma^2}}, \quad (1)$$

where s_{I} and $s_{\text{non key}}$ are the sizes of I- and non key pictures, and γ is the ratio between s_{I} and $s_{\text{non key}}$. During profiling, Blink requires the user to avoid intensive environmental changes (*e.g.*, people moving or light condition change), which assures that the video scene remains relatively stable. In this case, non key picture sizes are small and γ is typically greater than 2. Taking it into Eq. (1), we can derive a threshold of the Crest factor. A packet flow is considered suspicious if the Crest factor of its data rate profile is greater than the threshold.

4.2 Suspect Flow Probing

After profiling, Blink asks the user to turn on/off the light to generate stimuli. To assure the effectiveness of stimuli, the user is suggested to tune the ambient light level as low as possible. In daytime, it is often enough to assure the sensitivity of Blink by simply drawing the curtain to keep the sun off. After applying a stimulus, Blink uses the light sensor of the mobile device to identify the time instant of stimulus, and then engineers statistical tests to probe suspect packet flows.

Input: Packet flow period T; data rate profile $\{\mathbf{r}_1, ..., \mathbf{r}_\mathsf{T}\}$; detection window T_{det}; the time instant of stimulus t_s; the profiling time T_{profile};
Output: p-value;
1: Filter data rate series to remove periodic peaks;
2: **for each** r_τ in the filtered data rate series **do**
3: Use r_τ and $\mathbf{r}_{\tau \bmod T}$ to compute a t-score s_τ;
4: **end for**
5: Find the max in $s_{t_s}, ..., s_{t_s+T_{\text{det}}}$, denoted as $s_{\max}$;
6: Divide the data rate series measured in the profiling stage into periods, assume there are K periods;
7: **for** $k = 1 : K$ **do**
8: **for** $t = (k-1)T_{\text{det}} : kT_{\text{det}}$ **do**
9: Use r_t and $\mathbf{r}_{t \bmod T}$ to compute a t-score s'_t;
10: **end for**
11: Find the max in $s'_{(k-1)T_{\text{det}}} ... s'_{kT_{\text{det}}}$, denote it as $s'_{max,k}$;
12: **end for**
13: Compute p-value using s_{max} and $s'_{max,1}, ..., s'_{max,\frac{T_{\text{profile}}}{T_{\text{det}}}}$.

Algorithm 1: The probing algorithm of Blink**.**

Sensing light stimuli. During profiling, Blink keeps sampling the light sensor of the mobile device to learn the distribution of light variation. In the probing phase, whenever a sample of light level is obtained, Blink computes its difference with the previous sample, and performs a statistical test to determine if the observed variation is different with the profiled distribution. To this end, Blink employs the t-test, a statistical tool commonly used to assess the effect of a stimulus [14]. Given two distributions, the t-test computes a t-score using the distributions' means and standard deviations, and then maps it to a p-value based on the degrees of freedom. If the p-value is larger than a threshold chosen based on an alpha level, the two sets of data generating the two distributions are considered statistically different. To detect light stimuli, Blink uses an alpha level of 0.05, typical for scientific and medical studies.

Statistical testing. Blink probes suspect packet flows by examining whether their data rates have abnormal increases around the time instants of stimuli. Algorithm 1 describes the probing process.

In the first step, Blink filters data rate series for each packet flow using Algorithm 2 to remove data rate peaks yielded by the transmissions of I-pictures. The reason to do this is two-fold. First, I-pictures do not encode scene differences, thus do not response to light stimuli. Second, because I-pictures are larger than others, they have much higher impacts when conducting statistical tests in the presence of packet jitters. In particular, any small timing shift of an I-picture transmission will be treated as an abnormal rate increase. To address this issue, Blink identifies the data rate peak in each period and then removes the packet cluster corresponding to the peak. Specifically, as shown from Line 4 to Line 11 in Algorithm 2, for each period of the packet flow, Blink first initializes a small packet cluster, denoted as P_{drop}, in which the receiving times of packets are closest to the time instant of the rate peak. Then it analyzes the packet interval and gradually grows P_{drop} to include more packets belong to the same picture. Finally, it removes P_{drop} from packet flow and then re-estimates the data rate.

1: Divide the data rate series based on its period;
2: **for each** period **do**
3: Denote the packet set of this period as P;
4: Let $\mathsf{P}_{\text{drop}} = \emptyset$ and $\Theta = \emptyset$;
5: Identify the time instant of the peak; denote it as t_{p};
6: Find four packets whose receiving times are closest to t_{p}, then insert them to P_{drop}.
7: Compute their intervals and insert them to Θ;
8: **while** P is not empty **do**
9: Find the packet p whose receiving time is closest to t_{p}, denote its interval to the closest packet in P_{drop} as θ;
10: End the loop if θ is an outlier of Θ, otherwise insert p and θ into P_{drop} and Θ, respectively.
11: **end while**
12: Remove P_{drop} from packet flow.
13: **end for**
14: Re-estimate data rate.

Algorithm 2: Filter data rate series to remove periodic peaks.

In the second step, at the time instant of light stimulus, Blink computes a t-score using t-test to characterize how different the data rate is from the profiled rate distribution. Fig. 8 shows the light level, rate series, and the computed t-scores for a live video stream before and after removing the periodic rate peaks. As shown in the figure, because of packet jitters, the periodic transmission of I-pictures results in substantial misleading t-scores. After the filtering, the response of the video stream can be clearly observed, as the t-score increases significantly around the time instants of light stimuli.

We note that it is often difficult to accurately identify the exact time instant at which the camera's packet flow starts to respond to a stimulus. This is because of the unknown streaming delay caused by picture buffering at the camera. To address this issue, when evaluating the response at a time instant t_s, Blink examines the maximum t-score within a *detection window* of $[t_s, t_s + \mathsf{T}_{det}]$, which is employed to tolerate the maximum streaming delay. In Blink, the detection window size T_{det} is set to 4 s based on an empirical survey of commodity cameras' buffering delays. Then, the reference set of the statistical test is derived in a compliant way from the profiling results (as shown from Line 6 to Line 12 in Algorithm 1). Blink then performs the t-test to compare the maximum t-score within the detection window with that of the reference set, and evaluates the result against the threshold under a given alpha level to determine if a wireless spy camera is detected.

4.3 Multiple Rounds of Stimuli

To improve detection accuracy and robustness, Blink combines the probing results obtained after multiple rounds of stimuli to improve detection performance. Following the Fisher's method [18], the results of the t-tests are combined by $X^2_{2k} = -2\sum_{i=1}^{k} \ln(p_i)$, where p_i is the p-value computed in the ith round, k is the number of rounds, and X^2 follows a chi-squared distribution with $2k$ degrees of freedom. A new p-value is then determined based on the degrees of freedom and the chi-squared distribution. Then, the p-value is compared with the threshold under a given alpha level to determine the test result.

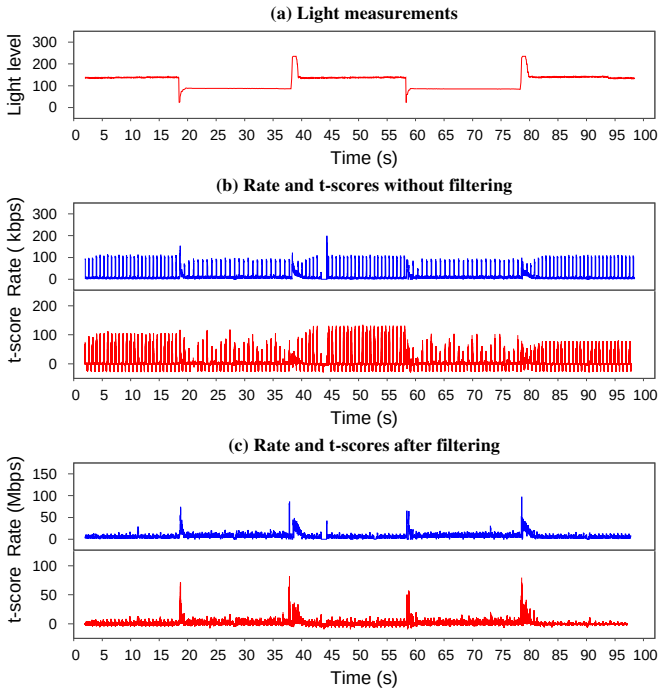


Figure 8: The traces of light levels, rate measurements, and t-scores when probing the packet flow of a live video stream during daylight.

5 FLICKER: ROBUST & AUTOMATED DETECTION

Flicker is a system that augments Blink to achieve robust and automated detection of spy cameras. Flicker can detect a spy camera even if it does not stream the video in live mode. In this section, we introduce the design of Flicker's stimulator, and then discuss how to identify the response pattern by examining the data rate variation of a packet flow. Finally, we present a probabilistic detector that further improves detection robustness in the presence of strong ambient light and disruptive network variations.

5.1 Stimulator Design

To achieve the design objective of Flicker, the stimulator must meet the following requirements.

- *Self-identifiable.* When the recorded video is not lively streamed, the spy camera will not respond to light stimuli in real-time. To determine whether a fluctuation of data rate is a response to stimuli, the stimulator of Flicker should produce a self-identifiable response pattern.
- *Human invisible.* When the recorded video is streamed block by block in progressive downloading mode, the light stimuli must last for at least one block period in order to yield a measurable response pattern. As a result, the stimuli must be human invisible to avoid being disruptive.

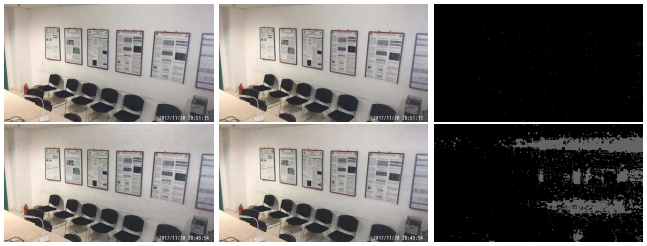


Figure 9: Consecutive pictures captured by the camera and the difference of their pixels without (the top row) and with (the bottom row) the stimuli of Flicker**.**

- *Error resistance.* To improve detection accuracy and robustness, the stimuli must be effective in the presence of strong ambient light and disruptive network variations.

Next, we discuss how to engineer and encode light stimuli to meet the above requirements.

Engineering flickering. Flicker harnesses daily used LEDs to stimulate spy cameras using high frequency flickering, which is human-invisible but can be captured spy cameras. Unlike LiShield [33] that only works for rolling shutter cameras, Flicker is effective against bot rolling shutter and global shutter cameras. Specifically, for rolling shutter cameras that capture the different parts of a image at different time instants, the flickering effect will be observed as the difference of brightness across pixels in different rows or columns, producing a striped pattern as long as the flickering frequency is higher than the video frame rate [33]. When the video frame rate is not a harmonic of the flickering frequency, the striped pattern will keep sweeping across the scene. For global shutter cameras that capture the entire image at the same time instant, the flickering effect will be observed as the difference of brightness between consecutive pictures, as long as the video frame rate is not a harmonic of the flickering frequency.

Flicker employs a cheap and portable circuit to harness daily used LEDs as stimulators. The circuit regulates the power supply of the LED, which first converts alternating current (AC) to direct current (DC) using a rectifier, and then employs a power MOSFET to switch the power supply at a high-frequency. We set the flickering frequency to 82.5 Hz, which is sufficient to avoid being disruptive to human eyes [20, 29]. Fig. 9 illustrates the flickering effect captured by a rolling shutter camera. We observe that although the striped pattern is of extremely low-contrast because of ambient light, the flickering effect remains pronouncing when comparing two consecutive pictures. In this case, the camera will devote a higher number of bits to encoding the scene transformation, resulting in a notable response pattern in the data rate of its packet flow.

Layered embedding of identification code. By controlling the on/off of stimuli, Flicker modulates the picture size of a video stream to embed an identification code. Different bit values are represented using different flickering frequencies, which in turn lead to different response patterns. Specifically, the flickering frequencies for bit '1' and '0' are set to 82.5 Hz and 1000 Hz, respectively. We note that an alternative is to represent bit '0' or bit '1' as flickering free. However, this will cause a varying light intensity of the LED due to different duty ratios, resulting in human visible disruptions.

To make stimuli effective across different video streaming modes, Flicker employs a layered coding structure, where different coding layers have different bit periods. Specifically, to detect a progressive downloading stream where the video is transfered block by block, the identification code is embedded at the first layer, with a relative long bit period of 1-minute to modulate block size. To detect a live stream where the video is transfered picture by picture, the identification code is nested inside first-layer bits, with a short bit period of 2*s* to modulate picture size while reducing detection delay. As an example, Fig. 10 illustrates the effect of Flicker's layered coding on the picture size and data rate of spy camera's video streams. The size and data rate peaks of I-picture are removed and filtered because they do not respond to light stimuli.

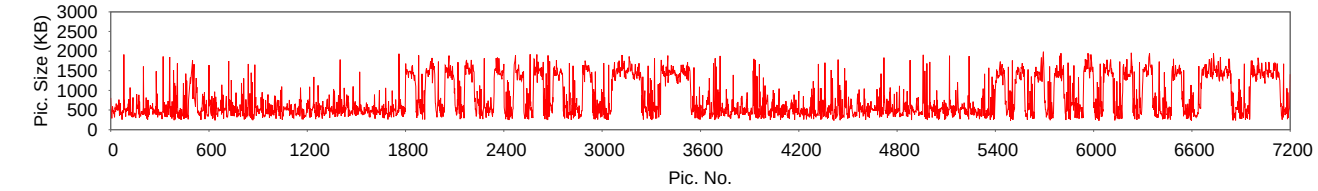
Adaptive FEC. To combat ambient light, Flicker protects the embedded identification code with adaptive FEC. Specifically, we set the length of identification code as 4-bit and 8-bit at layer-1 and layer-2, respectively. At both layers, Flicker protects the embedded identification code using Reed-Solomon (RS) code. Based on empirical experiments with daily used LEDs, Flicker sets the coding rate of RS code to 2/3 and 1/2 at night and during daytime, respectively. At runtime, the stimulator reads the ambient light level from a light meter, and then compares it with a pre-defined threshold to determine which coding rate should be used. After encoding, a three-bit rate header is appended at the beginning of the identification code, where the first two bits are set as '01' as a synchronization word, and the third bit indicates RS coding rate. We note that the coding rate of RS can be further optimized if Flicker can measure the intensity of the LED. In this case, it is possible to estimate the worst-case signal-to-noise-ratio, by using a physical model to estimate the degradation of stimuli intensity in a given detection range. This is left for our future work.

5.2 Rate Variation Decoder

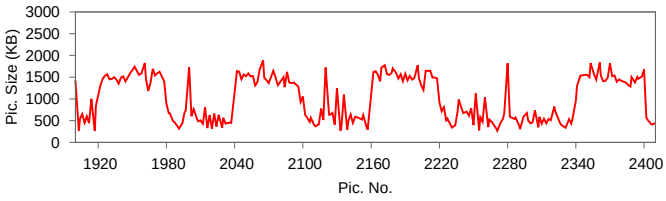
In the probing phase, Flicker first decodes the data rate variations of a packet flow, and then searches for the identification code to determine if the packet flow is produced by a wireless spy camera.

Classifying streaming modes. As the first step of probing, Flicker needs to classify the streaming mode to determine at which layer the decoding should be performed. To this end, Flicker leverages the fact that the mute period of progressive downloading is typically much longer than that of live streaming. Therefore, the detector classifies streaming modes by comparing the measured mute period with a pre-defined threshold. Based on empirical measurements, we set the threshold to 200ms.

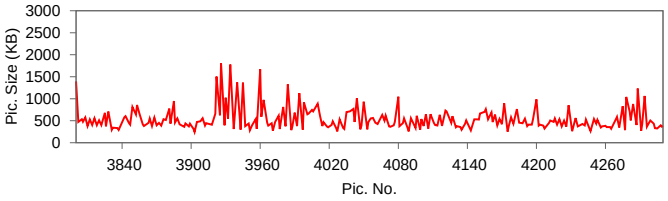
Filtering rate series. For each bit period, Flicker estimates the bit value based on the average data rate. When the rate measurements corresponding to the I-pictures are included in the estimation, the decoding sensitivity will degrade.For packet flows of live streaming, it is possible to eliminate I-pictures by reusing the filter described in Section 4.2. However, the filter requires a profiling process to learn packet flow period. In Flicker, we use a lightweight, aggressive filter to relieve profiling overhead. For each bit period,



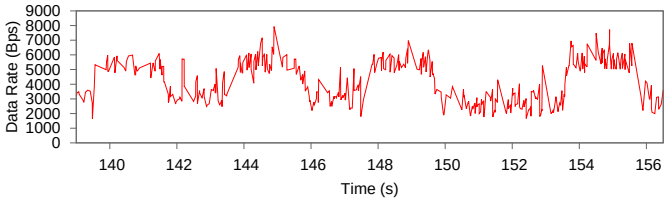
(a) An identification code of '0101' embedded at layer-1. The video is streamed at 30 s. Each layer-1 bit lasts for 1-minute, which contains about 1800 pictures. Each layer-2 bit lasts for 2 s. One layer-1 bit nests 30 layer-2 bits.



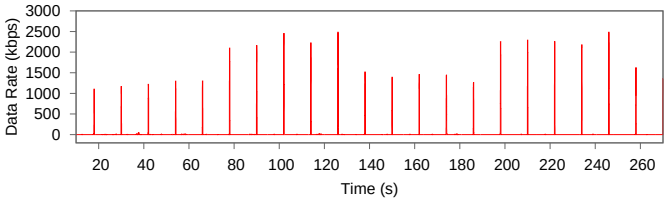
(b) A layer-2 byte of 0xa9 nested in the 2nd layer-1 bit of Fig. 10(a).



(c) The zoom-in of a part of the 3rd layer-1 bit of Fig. 10(a).



(d) The data rate of a live video stream, which is embedded with a layer-2 byte of '0xa9'.



(e) The data rate of a progressively downloaded video stream, which is embedded with four layer-1 bits of '0101'.

Figure 10: The effect of Flicker's layered coding on the picture sizes and data rates of spy camera's video streams.

the filter deletes all rate measurements that are larger than the median. This is assured to remove all data rate measurements corresponding to the I-pictures, because each GoP contains at least one non-key picture. We note that the aggressive filter may also remove some data rate measurements corresponding to P- and B-pictures, but this will not affect Flicker's decoding, because all P- and B-pictures within a bit period are under the stimuli of flickering.

Extracting identification code. To extract identification code, Flicker attempts decoding at every time instants of the data rate series. Specifically, at a specific time instant, Flicker first reads in a segment of rate measurements of three bit periods, and then measures the average rate within each period. Denote the results as r_1, r_2, and r_3. The decoding process terminates if r_1 is larger than r_2, which implies a mismatch of synchronization word. Otherwise, the detector compares r_3 with $\frac{r_1+r_2}{2}$ to estimate the value of the third bit, and then sets the rate of the RS decoder accordingly.

After parsing the header, the detector reads in all data rate measurements of the packet, and then measures the average rates within each bit period. To estimate bit values, a naive approach is to slice bits based on the threshold of $\frac{r_1+r_2}{2}$, which will lead to poor performance when the synchronization word is subject to noise. Flicker addresses the problem using the 2-means clustering. At the beginning of decoding, the two bits in synchronization word are used to set the initial centroids of each cluster. Then the clusters are gradually refined until the assignment no longer change. Finally, the cluster with higher average rate is assigned as '1', the other is assigned '0'.

With the decoded bit sequence, the detector calls the RS decoder to recover the identification code. In the following, we will discuss how to derive the detection result based on the receiving rate of identification code.

5.3 Probabilistic Detector

When there exists uncontrollable network variations, identification code can be erased if the number of flipped bits exceeds the error correcting capability of RS code. In addition, Flicker may wrongly decode a sequence of noise bits as the identification code, leading to false alarms. Flicker addresses these problems using a probabilistic detector.

Specifically, for an n-bit identification code, the probability of observing a false alarm, i.e., a sequence of noise bits that is wrongly identified as the identification code, is $\rho = \frac{1}{2^n}$. In a given time period T, the number of received fake identification code follows the Binomial distribution with a mean of $\frac{\rho T}{nT_{bit}}$ and a variance of $\frac{T\rho(1-\rho)}{nT}$, where T is the bit period. Based on this observation, Flicker computes a detection score by evaluating if the number of received identification code is statistically higher than the one characterized by the corresponding Binomial model. The probability that a packet flow is produced by a spy camera can be expressed as $1 - F(x \leq n_{rx})$, where $F(\cdot)$ is the cumulative distribution function of the Binomial distribution, and n_{rx} is the total number of received

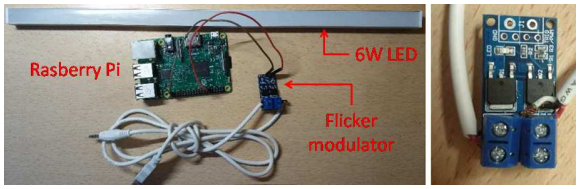


Figure 11: A prototype of the Flicker's stimulator when integrated with a portable LED.

identification codes in a given time period. Finally, Flicker compares the computed score with a pre-defined confidence level to determine detection result.

6 EVALUATION

We have implemented two prototypes of Blink and Flicker. Blink is implemented as a user-space application on an off-the-shelf Linux tablet. The prototype of Flicker is implemented based on a Raspberry Pi, which controls a portable circuit through GPIO to drive the lighting source (as shown in Fig. 11). In our evaluation, a 20 Watt portable LED is employed as the stimulator. The detector of Flicker is also implemented as a user-space application and deployed on the same Linux tablet.

Our evaluation is based on nine commodity wireless IP cameras and four wireless spy cameras of different models and manufacturers (as shown in Fig. 1). All of these cameras encode video in H.264 – the most popular formats for recording and wirelessly distributing video content. Our experiments are conducted in four rooms including a bedroom, a bathroom, a conference room, and an office room, which differ in sizes, wireless network conditions, and ambient light levels during daytime. Fig. 12 shows the snapshots of the four rooms captured by one wireless camera.

6.1 Blink Performance

We first evaluate the performance of Blink. During experiments, we turn on/off the ceiling lights of the rooms to stimulate wireless cameras. We conduct experiments under different video streaming settings. Specifically, three combinations of video resolution and frame rate are studied, including 1280x720 30 fps, 640x480 20 fps, and 352x288 10 fps. The bitrates of the produced video streams are approximately 512 kbps, 196 kbps, and 128 kbps, respectively. In each room and for each setting, we conduct a total of 40 stimuli to study the performance of Blink both in daytime and at night. The detection rates and false alarm rates are calculated under 95% confidence level.

6.1.1 Detection rate. Fig. 13 shows the detection rate of Blink after one round of stimulus. The error bars show standard deviations calculated for different cameras. We observe that the first round detection rate ranges from 18% to 72% across different settings and rooms. The detection rate is better for video streams of lower bitrates. For example, for video streams of 128 kbps and 512 kbps, the detection rates at night are from 54% to 72% and 18% to 31%, respectively. This is because video streams of higher bitrates typically involve in more frequent contentions with other network traffics, therefore their packet transmissions are subject to higher packet jitters, which makes rate measurements more noisy than

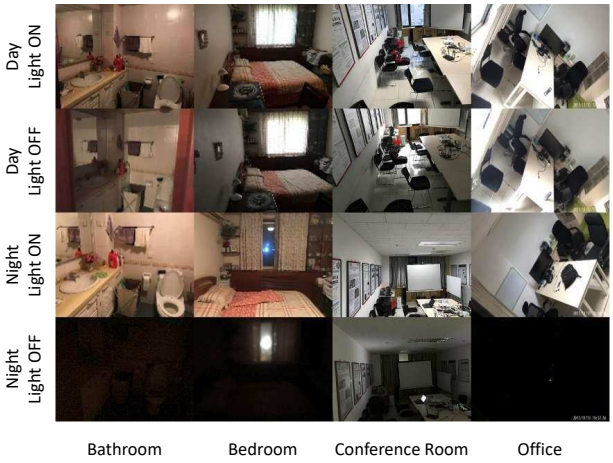
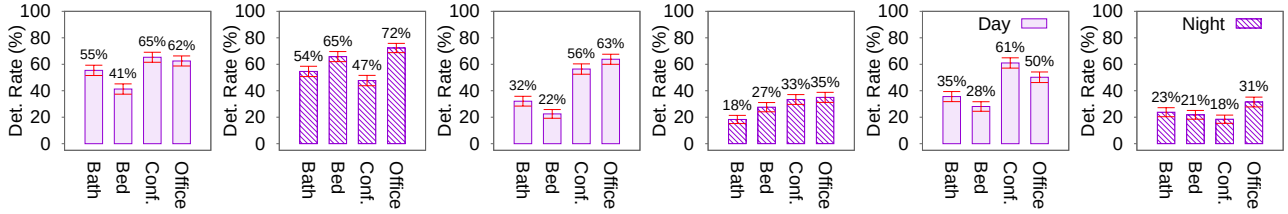


Figure 12: Snapshots of the experimental sites captured by the wireless spy camera.

those of video streams with lower bitrates. Surprisingly, we observe that the detection rate in daytime is slightly better that that at night. This is particularly obvious for video streams of higher bitrates. The reason is that, at night, the response of video stream is much more intensive than that during daytime, which results in higher bitrates and more noisy rate measurements at the time instants of stimuli.

We further study the detection rates after multiple rounds of stimuli. Fig. 14 shows the experimental results for video streams of different bitrates. The error bars show standard deviation calculated from the experimental results across different rooms and cameras. We observe that the detection rate improves quickly as the number of applied stimuli increases. For example, the detection rate for the 128 kbps video stream increases to 100% after only three rounds of stimuli. Even for the worst case when detecting the 512 kbps video stream at night, the detection rate increases to above 90% after seven rounds of stimuli. We note that the detection rate can be further improved by conducting more rounds of stimuli.

6.1.2 False alarm rate. Because of the explosive growth and the ubiquitous existence of video traffics, it is important for Blink to reduce the false alarm rate when probing live video streams other than that of the wireless spy camera. To this end, we study the false alarm rate of Blink when probing four representative live video streams, including a film (Fantastic Four), a cartoon (The Simpons Movie), a soccer game (2014 World Cup Final), and an interview program (Bill O'Reilly's Super Bowl interview with Trump). The results are shown in Fig. 15. We observe that the false alarm rate is the worst for the video stream of Fantastic Four because of the more frequent video scene variations, which causes intensive bitrate fluctuations that might be wrongly identified by Blink as responses to light stimuli. In comparison, the false alarm rate is much lower for the interview program because the video scene remains relatively stable throughout the program. In addition, we observe that the false alarm rate decreases quickly as the number of probing rounds increases. Specifically, for the video stream of Fantastic Four, the false alarm rate reduces to below 0.8% after only eight rounds of probing.



(a) Day, 128kbps. (b) Night, 128kbps. (c) Day, 192kbps. (d) Night, 192kbps. (e) Day, 512kbps. (f) Night, 512kbps.

Figure 13: Single round detection rate of Blink.

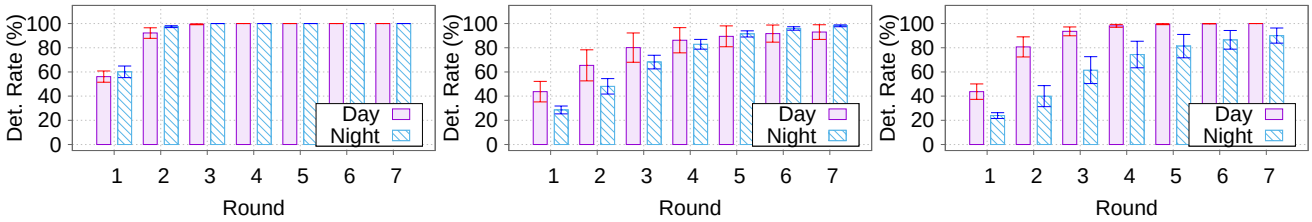
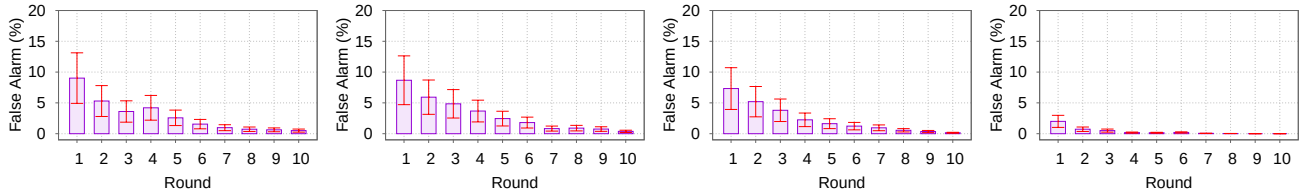


Figure 14: Multiple rounds detection rates of Blink for video streams of 128 kbps (left), 192 kbps (middle), and 512 kbps (right).



(a) Fantastic four: Rise of the Silver Surfer. (b) Animated film: The Simpsons Movie 2007. (c) Soccer game: 2014 World Cup Final. (d) Fox news: Bill O'Reilly's Super Bowl interview with Trump.

Figure 15: The false alarm rates of Blink when probing different live video streams.

6.2 Flicker Performance

To evaluate the performance of Flicker, we conduct experiments in the conference room, which is the largest room, has the strongest ambient light condition, and suffer the worst network condition among the four experimental sites. During experiments, we observe that a total of 34 active Wi-Fi access points are deployed around the experimental site, working on five different channels in the 2.4 GHz and 5 GHz band. The wireless cameras are deployed on three top corners of the room. The snapshots of the experimental site captured by one of the wireless cameras are shown in Fig. 16. To test the robustness of Flicker against ambient light, we intentionally keep one of the room's three ceiling lights on when conducting experiments at night. The strong ambient light condition forces Flicker to protect embedded identification codes using the RS code of rate-$\frac{1}{2}$. In addition, we tune the cameras' angles to avoid directly shooting the LED stimulator in the video scene.

6.2.1 Different video settings. We first evaluate the detection performance of Flicker across different video settings and deployment scenarios. The results are shown in Fig. 17. The error bars show standard deviations calculated for different cameras. Fig. 17a and Fig. 17c show the detection rate of identification code after correcting bit decoding errors using the RS code. We observe that the average detection rates are maintained above 80% and 73% for live and progressively downloading video streams, respectively. The

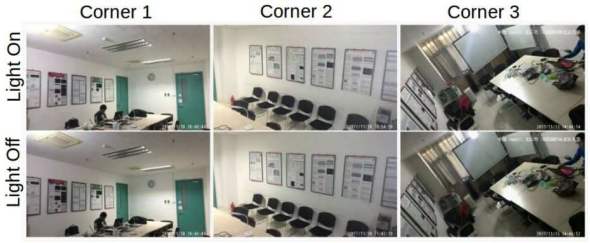
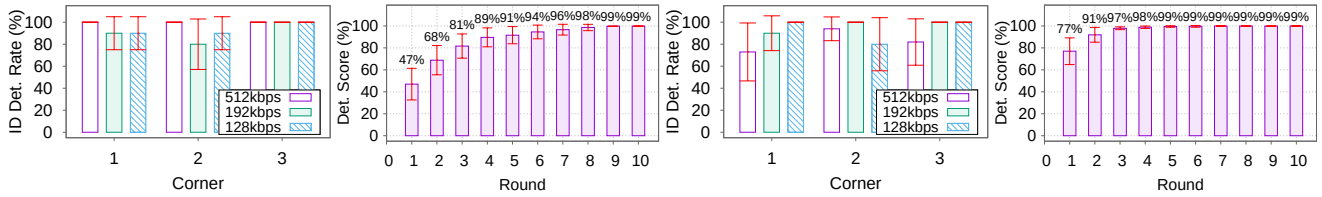


Figure 16: Video scenes captured by one of the wireless cameras when deployed in three different corners of the conference room.

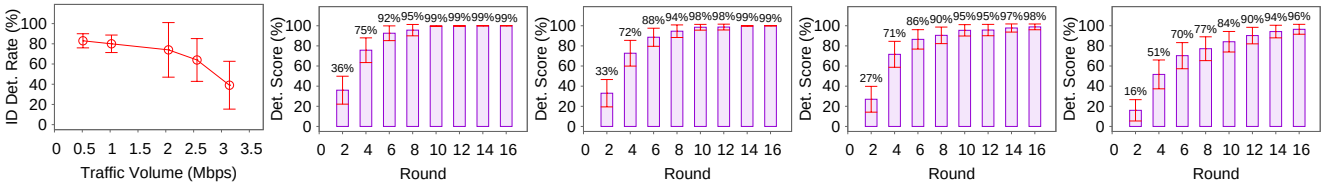
detection rate is the best when the cameras are deployed on corner-3, where the angle of the cameras is the closest to the direction of the stimulator.

We further evaluate the detection scores calculated by the probabilistic detector described in Section 5.3. Fig. 17b and Fig. 17d plot the worst case detection score among all settings (*i.e.*, 192 kbps at Corner-2 for live streaming and 512 kbps at Corner-1 for progressive downloading). When measuring the detection score for live video streams, we embed random identification codes at layer-1. The detection score is then averaged across 100 layer-1 codes and all cameras. We observe that the worst case detection scores increase quickly after multiple rounds of detection. Specifically, for



(a) ID code detection rate for live streaming. **(b) Worst case detection score for live streaming.** **(c) ID code detection rate for progressive downloading.** **(d) Worst case detection score for progressive downloading.**

Figure 17: The detection performance of Flicker for different video streaming settings and deployment scenarios.



(a) ID code detection rate. **(b) Det. scores, 1.024 Mbps.** **(c) Det. scores, 2.048 Mbps.** **(d) Det. scores, 2.560 Mbps.** **(e) Det. scores, 3.143 Mbps.**

Figure 18: The identification code detection rates and detection scores of Flicker under different volumes of contending network traffics.

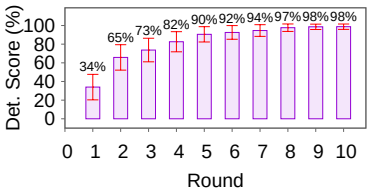


Figure 19: The detection scores of Flicker in the presence of scene variations.

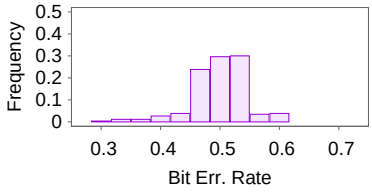


Figure 20: Bit error rate when using Flicker to probe the live video stream of Fantastic Four.

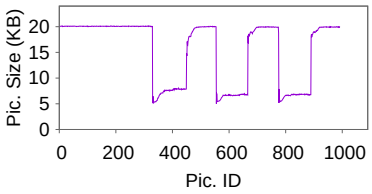


Figure 21: The picture size variations of a MJPEG video stream under light stimuli.

the live and progressively downloading video streams, the detection scores increase to above 91.6% and 91.9% after only five and two rounds of detection, respectively.

6.2.2 Impact of network contentions. To study the robustness of Flicker against network variations, we evaluate the performance of Flicker when probing a 512 kbps video stream under different volumes of contending network traffics. We intentionally inject ping packets into the channel of the wireless camera, and control the volume of contending traffic by tuning the lengths and time intervals of ping packets. The detection rates and scores are shown in Fig. 18, where the error bars show standard deviations calculated for different cameras. As expected, Flicker performs better under less network contentions. Specifically, the detection rate drops from 83% to 39% when the volume of contending traffics increases from 0.512 Mbps to 3.143 Mbps. As a result, the first round detection score drops from 48% to 16%. However, the detection score increases quickly after multiple rounds of detection even in the presence of strong network contentions. For example, when the volumes of contending traffics are 1.024 Mbps and 3.143 Mbps, the detection scores of Flicker increase to above 90% after 4 and 12 rounds of detection, respectively.

6.2.3 Impact of scene variations. In the presence of uncontrollable environmental variations, the bitrate of the video stream may fluctuate even without light stimuli. The impact is equivalent to an increased noise floor, which may increase error rate when decoding rate variations. To study the impact of scene variations, we intentionally place one LCD display into the scene of the wireless camera, and then play a clip of film on the display. The display takes about 20% pixels of the video scene. Fig. 19 shows the detection scores when probing a 512 kbps live video stream. We observe that, although the detection scores are slightly lower than the worst case we observed in Fig. 17, Flicker remains robust against uncontrollable scene variations after multiple rounds of detection. Specifically, the detection score increases to above 90% after only five rounds of detection.

6.2.4 False alarm rate. Finally, we evaluate the false alarm rate of Flicker. To this end, we use Flicker to decode the bitrate variations when streaming Fantastic Four in progressive downloading mode. Fig. 20 shows the histogram of bit error rate. We observe that the bit error rate is around 46.7% to 53.3% in 83.5% cases. Because of the high bit error rate, the detection rate of identification code is maintained at 0% consistently throughout our experiment that involves 217 packets.

7 DISCUSSION

Wireless scanning. In real deployment, the detector does not know the channel of the wireless spy camera a priori. A scan of wireless spectrum is therefore needed before stimulating and probing. Our

measurement shows that a 20s of profiling is often enough to identify suspect packet flows. This process can be further accelerated by first scanning the channel 1, 6, and 11 of Wi-Fi, which are the common options to deploy Wi-Fi in practice.

Other video codecs. The design of Blink and Flicker can be easily extended to detect wireless spy cameras that use other VBR-based video codecs. For example, Fig. 21 plots the picture size variations for a video stream encoded using MJPEG. During measurement, the light condition is changed at the time instants of the 330th, the 450th, the 550th, and the 660th pictures. We can clearly observe the response of MJPEG to light stimuli. Different from motion compensation based compression algorithms that encode the transformation of scene, MJPEG compresses each picture separately and independently. As a result, the produced pictures have different sizes under different light conditions.

Detecting non-wireless spy cameras. Even if the spy camera does not stream the recorded video wirelessly (*e.g.*, streams via wired networks or stores recorded videos locally in a SD card), Flicker can be employed as a countermeasure to limit the distribution of privacy-intrusive videos. Specifically, the identification code embedded by Flicker allows an authenticated cloud server to identify videos recorded by spy cameras by simply analyzing the picture sizes of the video stream. This approach is inspired by LiShield [33]. However, LiShield requires the cloud server to inspect each picture of the video stream to search for a watermark, which is privacy-intrusive and infeasible when the video content is encrypted. In comparison, Flicker is more practical as it avoids the decryption and inspection of video content.

Countermeasure. Blink and Flicker are designed to detect wireless cameras that use standard-compliant wireless and video coding chips, which have been the most frequently used tools for spying due to their low cost, wide availability, and easy deployability. We note that the attacker may design countermeasures against Blink and Flicker by developing proprietary wireless protocols, video encoding and processing algorithms, which will however substantially increase the manufacturing and deployment cost. From this perspective, Blink and Flicker will significantly rise the bar of visual privacy invasion.

8 RELATED WORK

Traffic analysis. Prior studies exploit the statistical features of network traffic to infer application protocols [31], fingerprint users [16, 25, 27] or websites [13, 19], detect drones [23, 24], and identify the type of wireless radios [28]. Our work is most closely related to prior studies that classify audio/video traffics [12, 21, 26, 30, 32]. However, considering the explosive growth and the ubiquitous existence of video traffic (*e.g.*, the traffics ofvideo chatting, smart TV, or legal surveillance cameras deployed in neighborhood), the detection of video traffic doe not reliably assert the presence of a wireless spy camera [15]. To address this issue, the stimulating-and-probing approach proposed in this paper exploits the association between the variations of network traffic and the user's private space. The focus is on designing the pattern of stimuli and decoding the response of the spy camera, which go beyond the scope of traditional traffic analysis.

Camera detectors. Prior camera detectors are cumbersome to use and require significant user involvement. A common approach is to first illuminate a suspicious place using laser or flashlight, and then ask the user to search for the tiny glint of the camera lens [6]. The user has to scrutinize the entire private space, which typically requires a slow and meticulous sweep while providing no assurance for detection. Other detectors assist the user to localize wireless sensors by tracking or analyzing the pattern of RF signals [8, 22]. However, these approaches cannot associate the sensor with the user's context to determine whether the sensor is spying on the user's private space.

Camera countermeasures. Existing RF-based countermeasures against wireless cameras rely on indiscriminate jamming to block the entire wireless medium [3, 7], which inevitably causes severe performance degradation of legitimate network devices. By accurately identifying the packet flow of the wireless spy camera, Blink and Flicker can be integrated with existing RF-based countermeasures, enabling smart jamming that corrupts the spy camera's packet flow while minimizing the interference with other devices.

LiShield [33] protects a physical scene (*e.g.*, a document or a paint) from photographing by jamming the visible light spectrum. It interferes with the rolling-shutter image sensor by modulating a lighting source with high-frequency flickering, which imposes a striped watermark on the images captured by the camera. However, LiShield cannot black out the entire video scene, and may require a customized lighting source with high intensity when protecting a large space. When the effect of flickering is overwhelmed by ambient light, LiShield relies on an authorized server to detect the embedded watermark when the video is uploaded to the cloud. However, the server must inspect the image content to search for the watermark, which is privacy-intrusive, and is impossible when the video is encrypted. In comparison, Blink and Flicker can detect the wireless spy camera's packet flow without inspecting the video content.

9 CONCLUSION

In this paper, we propose a stimulate and probe approach to detecting wireless spy cameras. Following this approach, we design and implement two practical systems, namely Blink and Flicker. Blink is a lightweight mobile app that leverages user-triggered light stimuli to detect wireless spy cameras in live streaming mode. Flicker augments the design of Blink by using commodity LEDs to generate human invisible light flickering as the stimuli. Flicker can detect wireless spy cameras across different streaming modes, and improves detection robustness by encoding light flickering to combat disruptive network variations. Extensive experiments show that our systems can accurately detect wireless spy cameras under a wide range of settings, including different ambient light intensities, video streaming modes and configurations (*e.g.*, resolution, frame rates), wireless interference conditions, and uncontrollable environmental variations.

10 ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers and our shepherd Marco Gruteser for providing valuable feedback.

REFERENCES

[1] Divx. http://www.divx.com/.
[2] H.264. http://www.itu.int/rec/T-REC-H.264.
[3] Jammerall. https://www.jammerall.com.
[4] Jd.com. https://jd.com.
[5] LTE. https://tinyurl.com/l67mdc9.
[6] Pimall. http://www.pimall.com.
[7] The signal jammer. https://www.thesignaljammer.com.
[8] Spygadgets. http://www.spygadgets.com/counter-surveillance/.
[9] Xvid. https://www.xvid.com/.
[10] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, 2016.
[11] AirBnb. Electronic surveilance devices. https://www.airbnb.com/help/article/887/.
[12] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing skype traffic: When randomness plays with you. ACM SIGCOMM, 2007.
[13] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. ACM CCS, 2012.
[14] G. Casella and B. Roger. *Statistical Inference*. Duxbury, 2nd edition, 1990.
[15] Cisco. Vni global fixed and mobile internet traffic forecasts. http://www.pimall.com.
[16] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. ACM CODASPY, 2015.
[17] Dailymail. Couple horrified to discover they are being recorded with hidden cameras while renting an airbnb apartment. http://www.dailymail.co.uk/news/article-4791366/.
[18] R. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd (Edinburgh), 1925.
[19] X. Gong, N. Kiyavash, and N. Borisov. Fingerprinting websites using remote traffic analysis. ACM CCS, 2010.
[20] Y. Jiang, K. Zhou, and S. He. Human visual cortex responds to invisible chromatic flicker. 10:657–62, 05 2007.
[21] W. Li and A. W. Moore. A machine learning approach for efficient traffic classification. MASCOTS, 2007.
[22] Z. Li, Z. Xiao, Y. Zhu, I. Pattarachanyakul, B. Y. Zhao, and H. Zheng. Adversarial localization against wireless cameras. ACM HotMobile, 2018.
[23] B. Nassi, R. Ben-Netanel, A. Shamir, and Y. Elovici. Game of drones - detecting streamed POI from encrypted FPV channel. *CoRR*, 2018.
[24] P. Nguyen, H. Truong, M. Ravindranathan, A. Nguyen, R. Han, and T. Vu. Matthan: Drone presence detection by identifying physical signatures in the drone's rf communication. ACM MobiSys, 2017.
[25] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. ACM MobiCom, 2007.
[26] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network characteristics of video streaming traffic. ACM CoNEXT, 2011.
[27] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. Usenix WOOT, 2016.
[28] S. Siby, R. R. Maiti, and N. O. Tippenhauer. Iotscanner: Detecting privacy threats in iot neighborhoods. IoTPTS, 2017.
[29] C. Tyler. Analysis of visual modulation sensitivity. ii. peripheral retina and the role of photoreceptor dimensions. 2:393–8, 04 1985.
[30] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 2006.
[31] C. V. Wright, F. Monrose, and G. M. Masson. On inferring application protocol behaviors in encrypted network traffic. *J. Mach. Learn. Res.*, 2006.
[32] W. Zai-jian, Y. n. Dong, H. x. Shi, Y. Lingyun, and T. Pingping. Internet video traffic classification using qos features. ICNC, 2016.
[33] S. Zhu, C. Zhang, and X. Zhang. Automating visual privacy protection using a smart led. ACM MobiCom, 2017.