Information Leakage in Encrypted IP Video Traffic

Chris Wampler*, Selcuk Uluagac†, and Raheem Beyah*
*School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30313, USA
emails: wampler.chris@gatech.edu, rbeyah@ece.gatech.edu
†Electrical and Computer Engineering Department
Florida International University
Miami, FL 33074, USA
email: suluagac@fiu.edu

***Abstract*—Voice chat and conferencing services may be assumed to be private and secure because of strong encryption algorithms applied to the video stream. We show that information leakage is occurring in video over IP traffic, including for encrypted payloads. It is possible to detect motion and scene changes, such as a person standing up or walking past a camera streaming live video. We accomplish this through analysis of network traffic metadata including arrival time between packets, packet sizes, and video stream bandwidth. Event detection through metadata analysis is possible even when common encryption techniques are applied to the video stream such as SSL or AES. We have observed information leakage across multiple codes and cameras. Through measurements of the x264 codec, we establish a basis for detectability of events via packet timing. Our laboratory experiments confirm that this event detection is possible and repeatable with commercial video streaming software.**

***Index Terms*—Information Leakage, IP Video Traffic**

I. Introduction

Video streaming services continue to expand into wider audiences with personal chat applications, work from home interactions, and teleconferencing including both commercial and government use. The need to ensure the security of these communications becomes vital as the cost of data loss increases. Our investigation in this work explores a potential avenue of attack against streaming video through network traffic analysis (NTA).

We have been able to demonstrate that information leakage occurs in encrypted video over IP traffic. Across a range of codecs tested (i.e., MJPEG, H.264, VP8), it is possible to detect events such as a person standing up or walking past a camera streaming live video, through NTA. Our inspiration for this work stems from the voice over IP work by Wright et al. [1], who detected spoken phrases through analysis of audio network traffic. We extend this concept into the video domain by establishing the basis for information leakage in streaming video and performing several experiments to quantify what types of events are detectable.

We found that variations in packet size and arrival time could indicate activity in a video stream and tested what types of events are detectable with a variety of codecs, cameras, and processing hardware. We focused particularly on the H.264 codec both in the Skype peer-to-peer video chat network [2] [3] and the open source x264 video encoder [4] implementations. We demonstrate the link between varying encoding times based on events being encoded and the resulting variation in packet arrival times by collecting timing measurements from the x264 encoder. We then establish the repeatability and consistency of measurements for a single camera and computer in a laboratory environment. To the best of our knowledge, this is the first work to analyze information leakage in encrypted video traffic.

The remainder of this paper is organized as follows. In Section II, we give a background of relevant information for video capture and encoding. We discuss related work in Section III. channel attacks. Our original tests showing event detection in a variety of codecs are documented in Section IV. We give evidence for our encoder performance theory through time stamps added within x264 encoder in Section V. In Section VI, we demonstrate repeatability of event detection in a laboratory environment with Skype's H.264 codec. Finally, in Section VII we conclude with the results of our research and future work.

II. Background - Video Encoding

This section mentions some of the video encoding background relevant to this research as some understanding of video encoding and image processing is necessary to appreciate the context of the work. More detailed information can be obtained from [5], [6].

Video encoding begins with a raw image capture from a camera. The camera converts analog signals generated from striking photons into a digital image format. Video is simply a series of such images generally captured 5 to 30 times per second (referred to as frames per second or FPS). The stream of raw digital data is then presented to a video encoder.

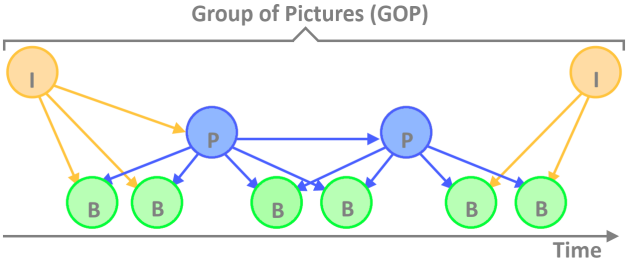

Fig. 1: Temporal compression with I, P, and B frames.

A video encoder is a set of algorithms which recognizes patterns within, and possibly between, frames in order to compress a video stream. Figure 1 demonstrates how temporal video compression compares content from a group of sequential pictures to achieve a higher compression ratio

using I, P, and B frames. Output from the encoder is in a standardized format which can be parsed and decompressed by a decoder for playback. There are many types of video encoding algorithms and as long as the output format is obeyed each algorithm can be implemented in a variety of ways.

Video is generally captured and encoded few times and is then stored, transmitted and played back many times. For this reason encoding algorithms are generally designed with greater complexity such that decoding hardware and software can be produced more cheaply. During the encoding process, images may be encrypted using a variety of algorithms. Of particular relevance to this work, the encoded and possibly encrypted data, can then be transmitted over a network. Many encoders (e.g. H.264) are capable of processing and transmitting in real time.

III. Related Work

Related and inspiring research by Wright et al. [1] investigates the ability to decode spoken phrases from voice over IP (VoIP) traffic. Video over IP transmissions are similar to VoIP in that IP traffic is expected to traverse untrusted domains. For this reason video and voice only chats are both encrypted in an attempt to prevent eavesdropping by an adversary. It was found, however, that due to the nature of the spoken word and common audio encoding techniques, current VoIP encryption practices are insufficient [1]. Wright et al. showed that through their techniques, it is possible to identify a specific spoken phrase from encrypted VoIP traffic with accuracies exceeding 50%. Due to the differences between video and voice traffic, our work focuses on identifying the leakage in video over IP. Through analysis of network traffic metadata such as packet size, inter-arrival time, and overall stream bandwidth, we see that information is being leaked. Similar to VoIP, the commonly used encryption techniques for video over IP do not pad or attempt to obfuscate the size or timing of their input data, leaving all of these metrics available for exploitation.

Another body of relevant work for our research is side-channel attacks. Side-channel attacks exploit predictability in the operation or output of cryptographic algorithms. This predictability allows an adversary to learn something about the plaintext or key without actually deciphering it. Early side-channel attacks allowed an eavesdropper to monitoring electromagnetic emissions from a computer monitor to determine what was being displayed [7]. Recently it has been demonstrated that using an iPhone's accelerometer, an attacker can determine passwords being entered on a nearby keyboard [8]. Other side-channel attacks exploit side effects of the encryption process and have even been used for encryption key retrieval. For example, a timing or power monitoring attack takes advantage of the differing complexity required to process a zero or one bit in a key. Variations in power consumption and execution time result from an effort to optimize performance in these areas [9]. Most optimizations, in both hardware and software, which allow an operation to complete more quickly, when possible, are preferable. This results in higher throughput and lower power consumption. However, when the difference in processing time and power use for processing a zero bit in an encryption key is significantly different than for processing a one bit, those optimizations can be taken advantage of to reveal the bits contained in the key itself. Kocher [10] demonstrated that this type of attack was possible against RSA decryption keys. Brumley [9] extended this work showing that timing attacks can even be executed remotely.

Our work does not attempt to reveal the original content of transmitted video, but instead uses NTA to show that there is a predictable network traffic response when events occur in view of a camera transmitting live video. Dyer et al. [11] discusses methods used to prevent identification of websites visited by a user through NTA. Their work stipulates that that there are no efficient countermeasures to side-channel traffic analysis for website identification. We demonstrate that side-channel attacks are possible against live video over IP transmissions and show what types of analysis reveal information about the encrypted content.

IV. Experimentation

This section details the experiments we performed to determine to what extent information leakage is occurring in live video streaming.

A. Automated Test Environment

Our initial experimentation aimed to produce the same phenomena observed in the previous VoIP work and determine both what types of events could be detected and what factors impacted the ability to detect recorded events in NTA.

In order to improve the ability to tune detection algorithms and understand how various types of events appeared in the network traffic, we utilized Python's Pcapy interface as it allows real time processing in lieu of a post-processing technique such as SharkTools [12]. Pcapy essentially wraps the libpcap library making live packet capture data, equivalent to that obtainable through Wireshark, available in a programming environment. Through use of Pcapy and Python's interface to MatPlotLib, real time graphs of network traffic statistics could be plotted and inspected for indication of events. This is the method used to obtain the captures shown in Figures 3 and 4.

With a software framework for inspecting live packet capture data established, the next step in experimentation was to create an environment where reproducible video captures could be performed. A small radio frequency isolation chamber was re-purposed as a light box where cameras could observe a subject in a controlled environment as shown in Figure 2. Using an Arduino micro-controller board with pulse width modulating outputs, several servos could be programmatically controlled through a serial connection. After creating a simple command interface to the Arduino, single key commands could be used to control the movement of a simple vehicle and analog dimming switch.

Each component of the test environment is labeled in Figure 2: (a), is a D-Link DCS-932L security camera which is configured for our tests to produce standard definition

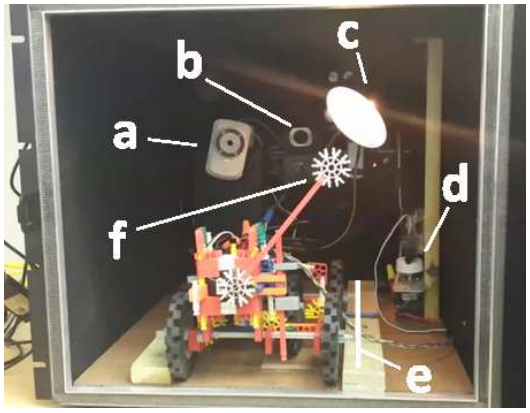

Fig. 2: Light box for controlled test environment.

(640x480 resolution) MJPEG video; (b), is a Logitech c615 web camera which captures raw high definition video at 1920x1080 resolution; (c), is an incandescent lighting source; (d), is an analog dimmer switch with an attached servo that allows light intensity in the test environment to be increased or decreased to arbitrary brightness; (e), shows the forward and backward path of motion for the test vehicle; finally, (f), is a pivoting arm which can be rotated left or right at a controlled speed. The combination of forward and backward motion with left to right movement is designed to simulate fast and slow objects of varying size moving past the camera at varying, but reproducible rates. When the chamber door is closed all light observed by the cameras in the environment is regulated. Not visible in this figure is a software controlled power switch which allows immediate switching from full brightness to no light and back as would be encountered when a light was switched on or off in a dark room.

B. Event Detection Across Multiple Codecs

Using this test environment, we observed network traffic graphs as video was transmitted using each of the cameras as well as through multiple encoding applications. For the security camera, which contains its own network interface, the traffic was transmitted directly from the device to the capturing computer platform through a single hop over a local area network. For the video chat applications, a USB webcam was connected to a second computer. Both the transmit and receive computers required an Internet connection in order to log into video chat services, but both Google Hangouts and Skype established a direct connection across the local building network once the connection was negotiated.

We verified that video transmission of a still scene yielded a stable measurement of packet average inter-arrival times (IAT) regardless of the camera or encoding application. Some variation from this stable level was observed at the beginning of the transmission, but after five seconds of capture average IAT measurements would level out under any of our capture methods. This stable rate varied between video sources, but is shown in Figures 3 and 4 as approximately 7500 μs. The MJPEG video from the D-Link camera was the most consistent in video transmission measurements with a similar measurement of IAT and bandwidth between measurements made on separate days. The video chat services, however, seemed to negotiate a different bit rate each time a new chat was started. This is consistent with our understanding of H.264 and video chat protocols. In order to ensure the best quality of service for a variety of users, an original video connection was made at a lower bit rate in order to ensure that that the call can be established. Once the connection is stable and there is no measurement of dropped packets or other factors limiting video quality, the protocol will attempt to negotiate a transition to a higher bit rate that is capable of providing a better picture to the user. Across multiple connections, we generally observed a traffic signature similar to that shown in Figure 3, but for connections originating from an outside network or for more congested local traffic conditions, the measurements would vary.

For each connection type, once a stable connection was established, the average inter-arrival time, packet sizes, and connection bandwidth varied little over time for a still scene. Once a stable connection was observed we would then initiate events by repositioning the test vehicle or changing the lighting conditions. By positioning the test vehicle close to the camera (a few inches away) and moving the arm in the field of view we could simulate a large object moving. Under good lighting conditions and with a perceived large object moving, we could observe the change in network traffic both from the Skype and Google applications. We did not see any noticeable change from the D-Link, but this is consistent with our understanding of the codecs.

For MJPEG, which does not take advantage of temporal compression, every frame is transmitted in its entirety. The images are compressed, but for an object that moves within the image where the background is uniform there is no significant change expected in the compressibility of the image. Thus, traffic measurements remain constant through the time frame of the motion.

For Skype's H.264 encoding, as shown in Figure 3, as well as for Google's VP8 encoding, no figure shown, we do observe this motion reflected in network traffic measurements. Most significantly, the average inter-arrival time of packets increases consistent with a short delay in transmission of the packets for the frame containing motion. This is again consistent with our understanding of the H.264 and VP8 codecs which do take advantage of temporal compression techniques. We discuss this phenomenon in greater detail in Section V but, conceptually, for a still scene the encoder can transmit a delta of the previous frame which is essentially identical to itself. Thus the computation needed to encode the new information is reduced, and the quantity of information that must be sent is similarly small. When motion is added to a scene, more time is required to compute the new encoding; during this time no new information is sent. The delay is short enough not to be noticed by a human observer but stands out in measurements accurate to the microsecond level. This appears as an increase in packet inter-arrival time, a decrease in average packet size during that gap, and a corresponding drop in the closely related

connection bandwidth measurement.

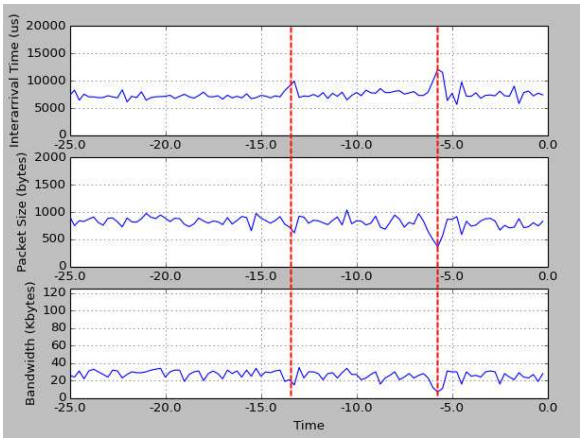

Fig. 3: Skype network capture marking time of object passed in front of camera.

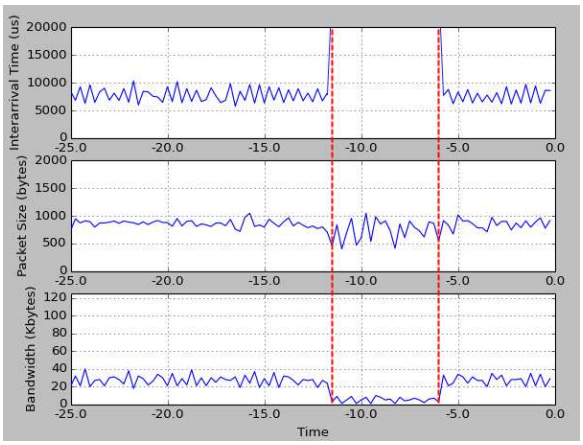

Fig. 4: D-Link network capture marking time when lights are turned off then on again.

Variations in the size and speed of the moving object were observed in NTA, but not with a linear relation to movement itself. When moving the test vehicle to the farthest position from the camera, most movements of the arm were not observable. It is expected that at that distance the moving object occupied such a small portion of the image as to be negligible in comparison to other noise factors. Also, in other experiments outside the light-box, a more dramatic change in the scene, which completely changed the scene content in a short time frame, would register as a large anomaly in the IAT of packets. For example, this would occur each time we opened the door to the light box.

Another significant factor observed in the measurements was the quality and intensity of lighting in the scene. In a very dim scene, even a large object's motion did not register in traffic measurements. Objects observed under low light lose color content and with low light the contrast between objects is reduced. This reduces the complexity of the scene and corresponding encoding effort. It follows that, if it is difficult to detect a moving object visually, it will be even more difficult to observe that change in the network traffic.

Correspondingly, dramatic changes in lighting were found to be very observable in network traffic and not only for the H.264 codec but also for MJPEG. Figure 4 shows a network capture from the D-Link camera's MJPEG video which demonstrates a dramatic change in all three of the metrics. At the time indicated by the dashed red line on the left, the lights in the test environment were turned quickly off and then back on again approximately 5 seconds later.

Similar reactions were observed for both the Skype and Google Hangouts recordings where changes in arrival time, packet size, and bandwidth all corresponded with the same moment that the light in the room was turned off and then back on again. The specific reason for the changes in network traffic between the different types of codecs is expected to be different, but the base understanding of how each of the codecs works is consistent with the network traffic analyzed. For MJPEG video, the dramatic change in the complexity and resulting compressibility of the captured image, based on brightness, would change the size of the transmitted frames and appear as a change in both packet size and bandwidth. The specific reasons for changes in inter-arrival time for an MJPEG codec are still uncertain, but could be a result of the codec vendor implementation that slows down traffic when there are only black frames to transmit.

For the Skype and Google applications, a change in inter-arrival time seems to be consistent with a change in frame rate. For low complexity frames with similar content, the frame rate was dramatically decreased. This is expected to be in relation to a conservation of network bandwidth where there would be almost identical frames transmitted. If this were detected, the frame rate could be significantly decreased while still presenting a good representation of recorded activity to the end user. In all three cases, the decrease in complexity of the image would result in higher compressibility and decreased bandwidth utilization. This would vary based on how much change in light there was. In our test setup as we varied the light in small increments, there was not generally any dramatic change in the network traffic. If the light were only slightly dimmed in the test setup, or in a real world case if the light in a room slowly dimmed as the sun set, there would be no immediate indication of an event occurring, but the overall change would be measurable as the bandwidth transitioned from a high to a low state.

TABLE I: Test cases and detection methods.

Source	**Codec**	**Detection Method**				**Encry-ption**
		Small Light Changes	Lights On/Off	Small Motion	Scene Content Change	
D-Link	MJPEG	A/B	A/B	B	B	None
Skype	H.264	-	A/B	A/B	A/B	AES
x264	H.264	-	A/B	A	A	None
Hangouts	VP8	A/B	A/B	A/B	A/B	SSL
WebRTC	VP8	-	A/B	A	A	SSL

A - Average Inter-arrival Time *B - Bandwidth*

Table I summarizes several video applications that were

tested. For each source we show the corresponding video encoding standard implemented by that application, effective methods for detecting various types of events, and by which metric they were detectable. Light intensity changes were detectable both through changes in the IAT of packets as well as bandwidth. These indicators were less well defined in the case of Skype traffic. Small changes in light intensity were observable in all of the codecs except Skype. Dramatic changes such as lights off to on were observable in all codecs tested. Motion is detectable through the IAT of packets for all but the MJPEG codec. For each codec where motion was detectable, a variable frame rate is supported and is expected to be the source of the IAT change. Even with MJPEG, a dramatic change in observed content – such as a complete background change or a large subject matter change as in a person occupying most of the frame moving out of view – would be observable, but this is more of a representation of change in image content compressibility than of motion itself. For the other codecs, smaller moments such as a hand wave or adjustment in sitting position could be observed through NTA.

It is again noteworthy that three applications noted in Table I used some form of encryption, and while the content of each individual frame observed is not decipherable, correlation can still be made between the images and events transmitted and the network traffic metrics indicated.

V. Encoder Timing Analysis of x264

To better understand the relationship between an encoder software implementation and the resulting network traffic signature, we measured the time required to encode frames compared to the resulting network packet arrival times. We chose to make our measurements for the x264 [4] open source implementation of the H.264 codec. This is the same codec standard used by the Skype application, and while the specific implementation of the standard is expected to be very different between the two software packages, the types of algorithms necessary to produce a video stream for each application are expected to be similar. Details for how we obtained our measurements are given in the Appendix. We show in this section the correlation between those measurements and corresponding network packet arrival times.

In Figure 5 we show the close connection between frame encoding and network packet arrival times. The topmost panel shows timing measurements of the encode function which demonstrates that time elapsed between calls to encode each frame is driven by the time required to complete the encoding of the corresponding frame at the host machine. The y-axis shows time measured in milliseconds while the x-axis indicates the frame number processed in the call to the encoder. In this test over 350 frames were encoded, each by a single call to the encode function.

The plot shows four spikes above 200 ms around frame numbers 75, 150, 200, and 300 as well as a smooth area in the measurements from frame 200 to 300. These features correspond to events occurring in view of the camera during the

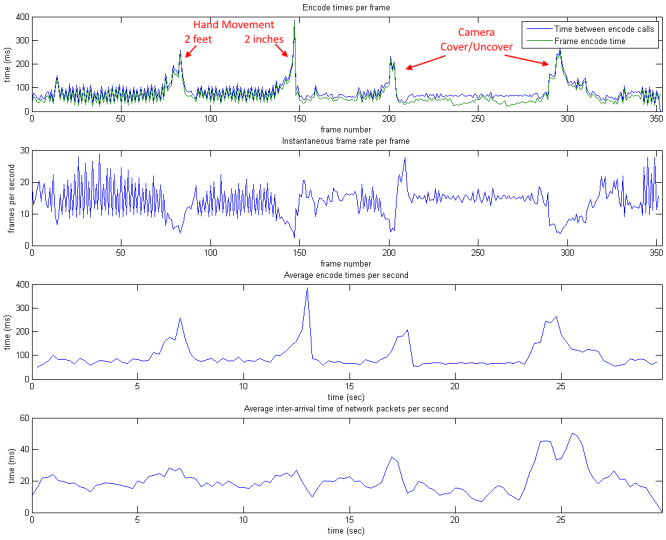

Fig. 5: Open source x264 encoder performance profiling.

encoding process. While sitting in front of a camera a subject passed their hand through the field of view approximately two feet from the lens corresponding with frame 75. This action was intended to affect approximately 50% of pixels in the frame. The spike at frame 150 corresponds to passing of the hand through the field of view approximately two inches from the lens. This causes the entire field of view to change as the hand enters and exits the frame. Starting at frame 200, the camera is completely covered to block any light and remains covered until approximately frame 300 where the camera is uncovered again. At other times during the test the subject sat motionless several feet from the camera.

We see from these results that the time required to encode a frame varies dramatically based on the motion and brightness of the scene in question. As an aid, the second panel shows the effective frames per second (fps) processed by the encoder. This is computed as the inverse of the time between encode calls from the first panel. The frame rate is fairly stable around 15 fps except when activity occurs. This is despite the target specification of 30 fps given in the pipeline description. We also observe that the x264 encoder is allowing a variable frame rate which has an upper limit bounded by the processing time for each frame. For the introduction of a moving object, which adds complexity to the encode operation, the frame rate drops momentarily. While covering the camera the frame rate is able to stabilize with the decreased complexity of processing an unchanging black image.

The bottom two sub-figures change the x-axis from frame number to a linear time scale in seconds. This is done by averaging data within a 250 ms window to create each point on the graph which is then smoothed to reduce noise. We see a correlation between the frame encode times and the network packet inter-arrival times consistent with the understanding that the encode time is related to the presence or absence of activity during the frame. When an event occurs, it causes the encoding to complete faster or slower; packets are then trans-

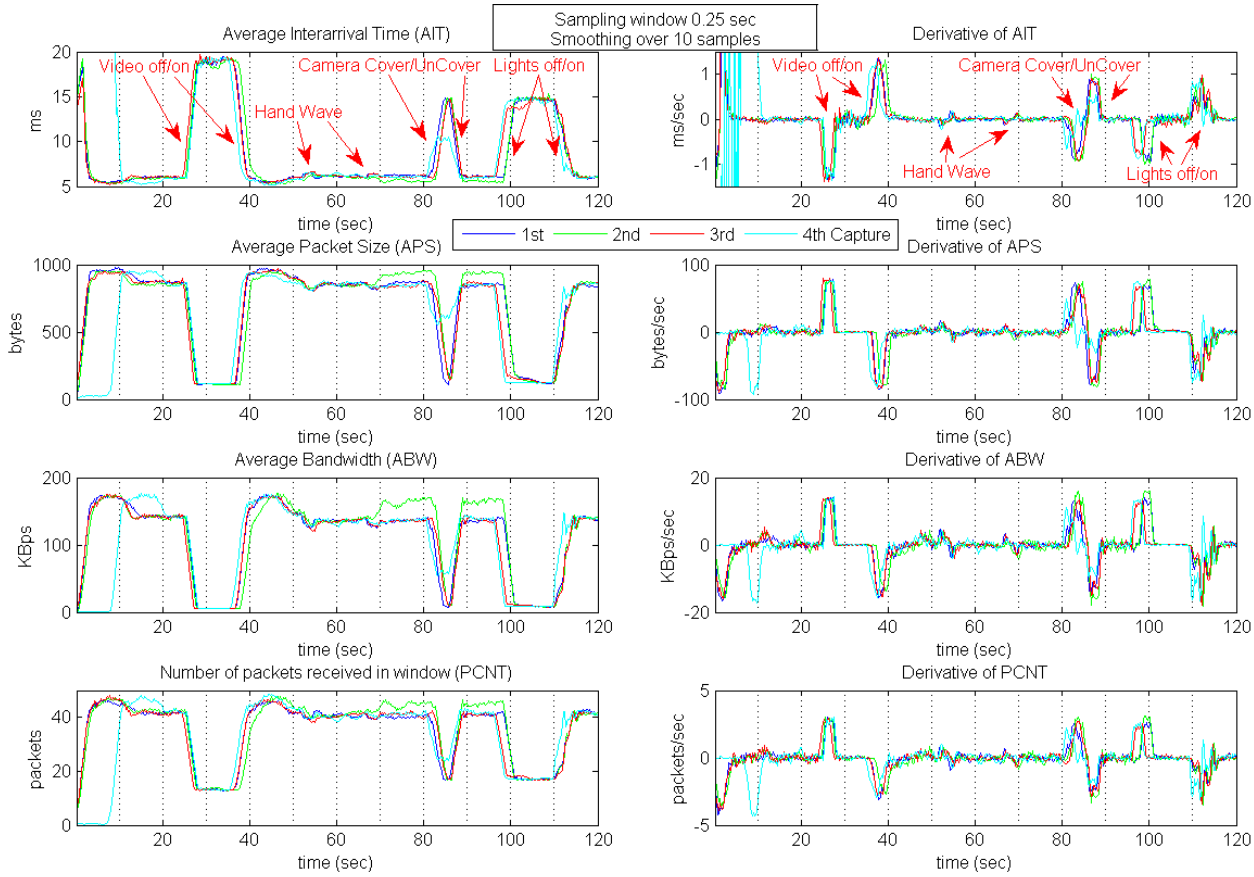

Fig. 6: Laboratory environment captures showing repeatability of measurements.

mitted immediately following encoding completion. Assuming consistent latency between packets, variations in packet arrival times correspond with events captured by the camera.

VI. Skype's H.264 Encoder

Although previous tests had shown that we could detect events in several types of codecs, our next step involved verifying the repeatability of tests on a single application. In order to verify that similar traffic signatures resulted from similar event types, we created an application to record network traffic and a corresponding video recording of Skype calls. We picked the Skype application due to its popularity and wide user base as well as its use of the H.264 encoding standard [13]. We note that we do not expect our experiments with the x264 encoder to be precisely representative of Skype's H.264 encoder performance, but since they are based on the same encoding standard the comparison is closer than most other options. As a commercial application advertising AES 256-bit encryption, Skype is expected by its users to be a secure and private means of communication. By testing against the Skype application, we take advantage of demonstrating the effect of strong commercial grade encryption on the ability to detect events through NTA, which we find to be minimal.

Using Skype4Py [14], a python wrapper to Skype's 3rd party API, we created event handlers for various activities in the Skype application including incoming and outgoing call initiation, call waiting, call ending and other standard Skype interactions. This application, which we have named Skype Auto-Answer, executes as a background service and attaches to a running Skype session which has already been logged into on the host machine. By detecting the beginning and end of a new video call, we could use Python to execute external applications. We used tcpdump to record all network traffic to a file, and FFmpeg to make a video screen capture of the Skype window.

With a method available for recording the video and network traffic, for purposes of comparison between calls, we then proceeded to make calls from various locations and hardware equipment to the lab server while executing the Skype Auto-Answer application. We were able to observe under varying conditions that the network traffic and corresponding recognizability of events through NTA was affected by camera type, computer hardware, and network connection bandwidth. Additionally, we observed that when these factors remained constant, the traffic measured was consistent between calls. Figure 6 shows plots for four such calls made from the same equipment. The calling computer, a Dell Latitude E6530 laptop with a high definition Logitech c615 external camera, initiated the call from within our network and the same sequence of events was recorded four times. The traffic analysis for all four packet captures are combined in the figure.

The Skype call is connected and we wait approximately

25 seconds for the video stream traffic to stabilize. Generally the stream stabilizes much more quickly than this as seen in the 1st, 2nd, and 3rd captures, where stable average IAT is observed after only a few seconds, but may take longer as in the 4th capture where stable traffic is not observed for 10 seconds. After waiting for the stream to stabilize, we turn off the video feed in the Skype application for ten seconds then turn it back on again. Between each action throughout the call, there is no movement in view of the camera. Next we move our hand past the camera approximately 12 inches from the lens, crossing through the field of view in approximately 1 second. After pausing, this action is repeated with the hand passing only 2-3 inches from the camera. Next the camera is completely covered to block out all light and remains covered for several seconds before uncovering. Finally, the lights in the room are turned off for several seconds and then back on again.

The annotations in Figure 6 show each event represented in the network traffic analysis. The eight plots show different event detection metrics. For all of these captures the sampling window is 250 ms. This sampling window is chosen based on the baseline number of packets being received. A window size that is too large will average too much data together preventing the detection of small variations such as those seen for the hand waves. A window size that is too small may not contain any packets at all. This sampling window allows us to plot the capture data on a linear time scale. On the left hand side of Figure 6 from top to bottom, we plot measurements of network traffic metadata including average inter-arrival time (IAT) between packets, average packet size (APS), average bandwidth (ABW), and network packet count (PCNT) for packets received during the sampling window. After averaging data in each window the data set is then smoothed across 10 samples. On the right hand side of the figure is the discrete derivative of the corresponding metric from the left hand side. These discrete derivative plots are additionally smoothed over 3 samples. The derivative plots are the most useful indicators for event detection by algorithm since they are a relative measurement.

Between the four captures we observe that although there are some changes in the exact manifestation of events in the NTA, each event is consistently detectable and events of different types are discernible from each other through careful observation. When the video is turned off we see the longest IAT since only audio and control packets are observed, whereas the camera being covered still causes some video frames to be transmitted, yielding slightly higher ABW and a corresponding drop in IAT. Covering the camera versus turning the lights on and off yields essentially the same signature with only a variation in the duration of the events. We can see, however, that as the lights are turned back on the IAT does not immediately return to the baseline level. This is due to the fact that the experiment room lighting is fluorescent rather than incandescent, and the lights turn on in a two stage process as the bulbs first warm up then reach full brightness. The hand waves yield the smallest variations and occasionally were not observable in the network traffic. The discrete derivative plots demonstrated by these captures become more important as we attempt to perform event detection with variations in the video transmission platform. Although baseline values across these figures are very similar between captures this is not the case especially for changes in camera resolution and network bandwidth. The high and low values are indicative of activity in the video but in a more general sense the change in measurement compared to the baseline value for a specific set of hardware is a more reliable indicator.

VII. Conclusion and Future Work

Our investigation of information leakage in encrypted video over IP traffic has found that for a variety of codecs and regardless of encryption for the tested datasets, we are able to detect events occurring in the field of view of a streaming video camera through network traffic analysis. This is possible through analysis of variations in packet sizes and arrival times. We also measured the time required to encode frames in the x264 encoder and shown the relationship between variations in encode time resulting in similar variations in network packet arrival time. Moreover, investigation of Skype's H.264 encoder through packet capture analysis has shown in a laboratory setting that event detection is repeatable between video calls. To the best of our knowledge, this is the first work analyzing encrypted video traffic for information leakage. Our future work will focus on building a more comprehensive framework to detect and classify events in encrypted video traffic.

References

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

[2] "Skype," http://www.skype.com/, Retrieved Dec 2014.

[3] S. A. Baset and H. G. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *the Proceedings of the IEEE INFOCOM*, 2006.

[4] L. Aimar, "x264," www.videolan.org/developers/x264.html, Retrieved Dec 2014.

[5] K. Jack, *Video Demystified: A Handbook for the Digital Engineer, 5th Edition*. Burlington, MA: Newness, 2007.

[6] D. Mitrovic, "Video compression," *University of Edinburgh*.

[7] W. Van Eck, "Electromagnetic radiation from video display units: an eavesdropping risk?" *Computers & Security*, vol. 4, no. 4, pp. 269–286, 1985.

[8] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *the Proceedings of the ACM CCS*, 2011.

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

[10] P. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *the Proceedings of the International Cryptology Conference on Advances in Cryptology*, 1996.

[11] K. Dyer, S. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *the Proceedings of the IEEE Sympposium on Security and Privacy*, 2012.

[12] A. Babikyan, "Sharktools," https://github.com/armenb/sharktools/, Retrieved Dec 2014.

[13] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi, "Tracking down skype traffic," in *the Proceedings of the IEEE INFOCOM*, 2008.

[14] "Skype4py 1.0.35," https://pypi.python.org/pypi/Skype4Py/, Retrieved Dec 2014.