Resource Thrifty Secure Mobile Video Transfers on Open WiFi Networks

George Papageorgiou[*]
UC Riverside
Riverside, CA
gpapag@cs.ucr.edu

John Gasparis[*]
UC Riverside
Riverside, CA
ioannis.gasparis@email.ucr.edu

Srikanth V. Krishnamurthy
UC Riverside
Riverside, CA
krish@cs.ucr.edu

Ramesh Govindan
USC
Los Angeles, CA
ramesh@usc.edu

Tom La Porta
Penn State University
University Park, PA
tlp@cse.psu.edu

ABSTRACT

Video transfers using smartphones are becoming increasingly popular. To prevent the interception of content from eavesdroppers, video flows must be encrypted. However, encryption results in a cost in terms of processing delays and energy consumed on the user's device. We argue that encrypting only certain parts of the flow can create sufficiently high distortion at an eavesdropper preserving content confidentiality as a result. By selective encryption, one can reduce delay and the battery consumption on the mobile device. We develop a mathematical framework that captures the impact of the encryption process on the delay experienced by a flow, and the distortion seen by an eavesdropper. This provides a quick and efficient way of determining the right parts of a video flow that must be encrypted to preserve confidentiality, while limiting performance penalties. In practice, it can aid a user in choosing the right level of encryption. We validate our model via extensive experiments with different encryption policies using Android smartphones. We observe that by selectively encrypting parts of a video flow one can preserve the confidentiality while reducing delay by as much as 75% and the energy consumption by as much as 92%.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication

Keywords

Wireless Video Transmission; Encryption; Distortion; Delay; Power

[*]The first two authors are listed alphabetically as per their first names.

1. INTRODUCTION

There has been a recent explosion in the number of video transfers from smartphones. In [8], it is reported that there was a fourteen-fold increase in the number of mobile video transfers between 2010 and 2011. Apps such as Facetime are becoming increasingly popular. Preserving the privacy or confidentiality of video transfers from eavesdroppers requires some form of encryption. Encryption however, comes at the cost of increased delays due to processing and battery consumption. Our thesis is that, one does not have to encrypt the entire video stream to be transferred to prevent an eavesdropper from accessing the content. If one were to encrypt only certain appropriately chosen parts of a video stream, there would be a sufficiently high distortion at an eavesdropper, that would protect the confidentiality of the content.

In this paper we seek to answer the following questions: (a) What parts of a video stream (or flow) should be encrypted in order to ensure the confidentiality of any eavesdropped content? and (b) What are the performance benefits (in terms of delay and battery savings) that one can reap, from only encrypting part of the video stream? Towards answering the above questions we develop a mathematical framework that quantifies the effect of the encryption process on the experienced video transfer delay and the expected distortion at an eavesdropper's site. Our framework takes into account both the network (wireless related effects) and video content (slow versus fast motion video) characteristics. We validate the analytical framework via extensive experimentation using Android smartphones. While our analytical framework does not at this time quantify the energy savings from reducing the encryption costs, we quantify the energy savings via experiments.

In more detail, our approach hinges on the insight that different packets in a video flow, carry varied significance with respect to the decoding process. For example, I-frames are critical for decoding, and encrypting only these frames (and sending the P-frames in the open) could potentially cause the video flow to be significantly distorted and thus, useless at an eavesdropper's site. Encrypting only parts of a video flow can drastically decrease the video transfer delay, as well as provide significant energy savings as discussed above. We consider the possibility of encrypting different types of packets towards determining the strategy that is most effective; each such strategy, wherein we consider encrypting a specific subset of packets from a video flow, is referred to as either "the encryption policy", or "the mode of encryption".

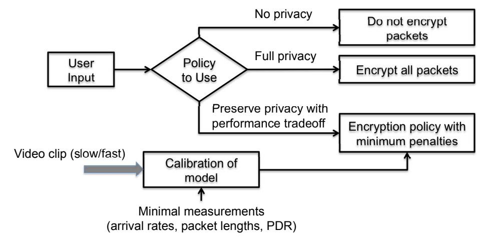

No privacy
Do not encrypt packets
User Input
Policy to Use
Full privacy
Encrypt all packets
Preserve privacy with performance tradeoff
Encryption policy with minimum penalties
Video clip (slow/fast)
Calibration of model
Minimal measurements (arrival rates, packet lengths, PDR)

Figure 1: Applicability of our framework.

Applying our analytical framework in practice: We envision our framework to be used as follows (see Fig. 1). The user captures video with her mobile device and initiates the streaming or transfer process. The UI prompts her with the choices available with respect to privacy. The set of choices could include the two extreme cases where all packets are either encrypted or transmitted in the open. A third choice would allow the user to minimize performance penalties (in terms of processing delay and energy consumption) while largely preserving confidentiality. If this option is chosen, the analytical framework is used to determine the appropriate encryption policy. The model is first calibrated with a few sample measurements to estimate scenario parameters; a tool such as AForge [1] can be used to estimate the motion level in the video, while the device capabilities and network conditions are estimated to determine the penalties with each partial encryption choice. The correct level of encryption is then applied while transferring the video.

In summary, our contributions in this paper are:

- We develop an analytical framework that captures the impact of an encryption policy (which packets are encrypted) on key performance metrics of a secure video transfer, viz., the delay due to the encryption process and the distortion at an eavesdropper's site. The framework provides a quick and effective way of quantifying these performance metrics for any given generic encryption process, wireless channel parameters, and the type of video content.
- We implement various encryption policies and demonstrate the effectiveness of only encrypting a part of the video flow via extensive experiments. The experiments also validate our analytical framework. Specifically, by using the Android Native Development Kit (NDK) we implement a mechanism that allows us to deploy different encryption policies on Android smartphones. We evaluate the performance in several scenarios that span differently encoded video streams (various GOP sizes) with different characteristics (slow-motion, fast-motion video streams) and with different encryption policies.

Key results: The key observations from our evaluations are the following:

- By only encrypting parts of a video flow, confidentiality can be preserved while achieving both battery savings and reduced transfer latency. In many cases, these performance benefits are significant; the transfer latency reductions were in some cases 75% and the energy consumed was reduced by 92% in the best case.
- The right strategy in terms of what parts of a video flow to encrypt, depends on the content itself. We find that the encryption of *I*-frames distorts slow motion video more than fast motion video; the encryption of just the *P*-frames distorts fast motion video more than the slow motion video. This is because, rapid changes between scenes in fast-motion videos cause the *P*-frames to carry significant information regarding the content, whereas with slow motion video, these frames do not carry much information.
- Due to the same reasoning as above, with slow-motion video the encryption of the *I*-frames sufficiently protects the content from an eavesdropper; significant savings in cost in terms of the delay and power consumption are possible with this strategy. With fast-motion video, 20% of the *P*-frames need to be encrypted in addition to the encryption of the *I*-frames, to ensure confidentiality. As a consequence, the savings in cost are less significant.

2. BACKGROUND AND RELATED WORK

Video Standards and Terminology: Standards such as MPEG-4 [21] and H.264/AVC [34] specify the encoding and transmission of video flows over a network. Typically, the initial video stream has a repetitive structure called Group of Pictures (GOP), which contains an initial *I*-frame followed by *P* and *B* frames (*B*-frames are optional). The size of the *I*-frame varies from GOP to GOP, while the sizes of the *P* and *B* frames differ within and across GOPs. Depending on the Maximum Transmission Unit (MTU) of the network, each frame is segmented into a number of packets that are transmitted over the network[1]. With predictive source encoding, the *I*-frame can be decoded independently of any other information within the GOP and each of the *P* and *B* frames use the *I*-frame as a reference [14]. In this work, we assume an *IPP*. . . *P* encoding structure for each GOP. We refer to the distance between consecutive *I*-frames as the GOP size.

Encryption in Commercial Video Delivery: Various solutions have been proposed for commercial video delivery (not mobile or wireless specific). The HTTP Dynamic Streaming (HDS), developed by Adobe, allows different levels of encryption for the content (low, medium, high). While at a high level it appears that with lower settings only a subset of the frames are encrypted to provide performance improvements during decryption at a receiving client, no details are readily available. In our work, we consider the appropriate frames for encryption during user transfers, with the primary purpose of protecting the content against an eavesdropper.

The HTTP Live Streaming (HLS), implemented by Apple as part of the QuickTime software, also specifies an encryption mechanism that uses AES and a method of secure key distribution based on HTTPS. Finally, the Dynamic Adaptive Streaming over HTTP (DASH) is an MPEG standard (ISO/IEC 23009-1) which enables media content delivery over HTTP. All these platforms operate over HTTP; however each of these uses different segment formats and therefore, to receive the content from each server, a device must support its corresponding proprietary client protocol.

Capturing the impact of wireless links on video: There has been some work on analyzing video communications over wireless links. In [22], the authors model the effect of wireless channel fading on video distortion. Video streaming over a multi-hop IEEE 802.11 wireless network is studied in [23, 27]. These efforts however, do not consider the impact of encryption on video distortion.

In [24] the selective encryption paradigm is discussed and various consumer applications are presented where compression and encryption occur together. However, no analysis is given regarding the behavior of key performance attributes in any case. In [32], the authors focus on wireless sensor networks and propose a scheme to selectively encrypt data based on the channel condition in order to improve video quality. However, in their study they do not take into account the characteristics of the video (slow vs fast motion) or the delay incurred due to the encryption process.

[1]Thus, one can assume that the number of packets in a GOP is a random variable.

Impact of Encryption on Battery: There are studies on the energy consumption due to encryption on wireless devices. In [15], a comparison of the energy consumption due to common encryption algorithms (AES, DES, 3DES, RC2, Blowfish and RC6) on wireless devices is presented. In [28], an analysis of energy consumption of RC4 and AES algorithms in wireless LANs is provided. However, these works do not consider selective encryption of video as we do here.

Other Related Work: Partial encryption of content in photos is considered in [29]. An encoding algorithm that extracts and encrypts a small but significant component of a photo, is proposed. In contrast, we consider the structure of a video stream towards preserving its confidentiality.

3. THREAT MODEL AND ASSUMPTIONS

Threat model: In this paper, we focus on a specific *threat* to privacy arising when transferring video flows or streams from a wireless device over a WiFi connection. Specifically, we consider the capture of content by an eavesdropper who is on the same open WiFi network. This situation arises typically in public places where WiFi is offered for free, e.g., cafes, malls, libraries, airport terminals (e.g., see [4, 7]).

Our *goal* is to secure wireless video transfers from eavesdroppers in a resource efficient way, with respect to the delay and the energy consumption on the wireless device. We expect the user to select the level of protection based on the sensitivity of the content. We *define* a selection policy $\mathscr{P}$ to be (i) the encryption algorithm that is used for protecting the transmitted packets, and (ii) the set of packets to be encrypted. In the extreme case where the content is highly sensitive and no information is to be leaked, all packets are to be encrypted. If the information is not sensitive, no packets are encrypted and this eliminates the performance penalties due to the encryption process. A user may simply seek to distort the video flow at an eavesdropper's site, thus risking the leakage of some information, but preserving the confidentiality to a large extent. Thus, we seek to provide the user with a control mechanism over the protection level for his content by defining various encryption policies. Each such policy results in the encryption of a specific sub-set of packets, (*I*-frame packets, *P*-frame packets, mixture of both) based on the required protection level and the associated performance cost. Depending on the level of distortion induced, an eavesdropper may be able to glean some information from the flow, but not all information; for example, he may determine that the flow is that of a football game, but may be unable to identify the players in the stream[2].

We focus on symmetric key encryption and *assume* that the mutual authentication and the agreement on the symmetric encryption method has been completed a priori (before the video is to be transferred). We also *assume* that the user has a valid key that has been established either using Public Key Infrastructure (PKI) or the standard Diffie-Hellman (DH) key exchange algorithm. For each encryption policy $\mathscr{P}$, the sender selects the appropriate set of packets to be encrypted. For each of these packets, the payload is encrypted using the symmetric key algorithm defined by $\mathscr{P}$ and the 'a priori established' secret key and transmitted on the wireless uplink. We emphasize that establishment of keys or authentication are not the focus of our work.

We *assume* that an unauthorized eavesdropper using the same WiFi network, is able to overhear transmissions but cannot decrypt packets, encrypted by the sender under $\mathscr{P}$. This affects the video distortion at the eavesdropper, since the encrypted packets cannot be used towards reconstructing the video during the video decoding phase.

We do not consider a traffic analysis attack by the eavesdropper. Specifically, we make no attempt to encrypt the headers that contain information such as IP addresses; while this can be easily accomplished this is not the goal of our work. The eavesdropper may be able to distinguish packets as belonging to either *I*-frames or *P*-frames based on their size or other characteristics. While the sender can obfuscate these features by using techniques such as padding the payload, we do not consider these possibilities in this work.

Other assumptions: Our key idea and approach is agnostic of whether the video flow is transferred using HTTP or RTP, and the transport layer in use. It can be applied with real-time streaming (e.g., Facetime) or for transfers to a server. Facetime uses RTP (or Secure RTP) over either UDP or TCP [9]. Other applications for video transfers (e.g., Google hangouts) also use UDP [5]. Commercial content delivery systems operate over HTTP and therefore, TCP. For tractability (so that we do not have to model TCP behaviors) we assume the use of RTP and UDP in our analysis. However, we experimentally demonstrate in Section 6 that our key ideas hold with HTTP/TCP.

We also defer the problem of jointly encoding and encrypting video. Furthermore, we expect that the volume of audio content is going to be much lower than video and thus, all of it can be encrypted. However, we do not consider this here and will explore these issues in future work.

4. OUR ANALYTICAL FRAMEWORK

In this section, we present our mathematical framework to characterize the effect of an encryption policy $\mathscr{P}$ on (i) the increase in delay due to encryption and (ii) the distortion at an eavesdropper due to a chosen encryption policy. An encryption policy $\mathscr{P}$ defines (i) the symmetric key algorithm that is used for encrypting packets at the sender and (ii) the packets to be selected for encryption.

4.1 Packet Success Rate

A key parameter to our delay and distortion analysis is the packet success rate of the network. As discussed above, we consider a WiFi network based on the IEEE 802.11 standard. There are various models (e.g., [13, 17]) that attempt to capture the operations of the IEEE 802.11 protocol. We use the model in [13] to represent the operations of the PHY and MAC layers. The model consists of three sets of equations (representing scheduling, channel access and routing) which are solved through a fixed point method. The solution is an approximation to the packet success rate p_s under the assumption that the traffic at the source nodes are persistent.

4.2 Delay

Video transfer delay is important especially for streaming applications. To compute the delay for each packet at the sender, we characterize the process as packets traversing a 2-MMPP/G/1 queue. The arrivals to the queue correspond to reading video file segments from the disk and storing them to a buffer in the memory. Based on the encryption policy $\mathscr{P}$, the server decides whether to encrypt the packet at the head of the queue or not, and then transmits the packet over the channel. This implies that the service time consists of a possible delay due to the encryption process, a backoff time due to possible collisions at the shared medium and the transmission time.

[2]We reiterate that a user may choose to encrypt all packets in an extreme case.

4.2.1 Arrival Process

The arrival process models the time instances where video files segments are read from the local disk and enqueued in a buffer for transmission. A segment that corresponds to an *I*-frame is typically larger than the MTU of the network and is thus fragmented into several packets with lengths equal to the MTU. On the other hand, a *P*-frame typically corresponds to a single packet that is much smaller than the MTU of the network. In the first case, the interarrival times of the packets that belong to an *I*-frame are much smaller compared to those that are associated with the arrival of *P*-frame packets. Therefore, there is a need to capture the two different phases of the arrival process. A natural choice is the Markov modulated Poisson process (MMPP), which is a doubly stochastic Poisson process where the rates are determined by the state of a continuous-time Markov chain [19].

We use a two-state Markov chain where the rate of transition from state 1 to state 2 is ρ_1 and from state 2 to state 1 is ρ_2. When the chain is in state 1 the arrival rate is λ_1 and the process models the arrival of *I*-frame packets (small interarrival times). When the chain is in state 2 the arrival rate is λ_2 and the process models the arrival of *P*-frame packets (larger interarrival times). The MMPP is then parameterized by the infinitesimal generator R associated with the Markov chain and the rate matrix Λ:

$$R = \begin{bmatrix} -\rho_1 & \rho_1 \\ \rho_2 & -\rho_2 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}. \tag{1}$$

The equilibrium probability vector π (representing the probabilities of being in state j, $j \in \{1,2\}$) is given by:

$$\pi = \frac{1}{\rho_1+\rho_2}(\rho_2, \rho_1). \tag{2}$$

4.2.2 Service Times

The total service time T of a packet is the time from the moment the packet reaches the server until it is successfully transmitted. This time includes the encryption time $T_e^{(\mathscr{P})}$ in case the packet is scheduled for encryption based on the encryption policy $\mathscr{P}$, the backoff time T_b that the packet may need to wait at the transmitter due to collisions and the transmission time T_t,

$$T = T_e^{(\mathscr{P})} + T_b + T_t. \tag{3}$$

The encryption time $T_e^{(\mathscr{P})}$ of a packet depends on the packet size and the encryption policy $\mathscr{P}$. In a typical GOP structure the initial *I*-frame is larger than the following *P*-frames (e.g., an *I*-frame can be 100 times larger than a *P*-frame) [14]. Therefore, as discussed earlier, an *I*-frame is typically fragmented into a sequence of packets that have lengths equal to the MTU of the network, while a *P*-frame corresponds to a single packet of a much smaller length. Moreover, the use of different encryption algorithms result in different delays and in general, affects the encryption time.

If $q^{(\mathscr{P})}$ is the probability that a packet is selected for encryption under an encryption policy $\mathscr{P}$ and p_I is the probability that a packet belongs to an *I*-frame, then the distribution $F_e^{(\mathscr{P})}(\cdot)$ of the encryption time $T_e^{(\mathscr{P})}$ is a mixture distribution derived from the distributions $F_{e,I}^{(\mathscr{P})}(\cdot)$ and $F_{e,P}^{(\mathscr{P})}(\cdot)$ of the encryption time $T_{e,I}^{(\mathscr{P})}$ of a packet that belongs to an *I*-frame and the encryption time $T_{e,P}^{(\mathscr{P})}$ of a packet that belongs to a *P*-frame, respectively:

$$\begin{aligned} F_e^{(\mathscr{P})}(\tau) &= P\{T_e^{(\mathscr{P})} < \tau\} \\ &= P\{T_e^{(\mathscr{P})} < \tau, I\text{-frame pkt, encrypted}\} \\ &\quad + P\{T_e^{(\mathscr{P})} < \tau, P\text{-frame pkt, encrypted}\} \\ &= q^{(\mathscr{P})} p_I P\{T_{e,I}^{(\mathscr{P})} < \tau\} + q^{(\mathscr{P})}(1-p_I) P\{T_{e,P}^{(\mathscr{P})} < \tau\} \\ &= q^{(\mathscr{P})} p_I F_{e,I}^{(\mathscr{P})}(\tau) + q^{(\mathscr{P})}(1-p_I) F_{e,P}^{(\mathscr{P})}(\tau). \end{aligned} \tag{4}$$

We compute the Laplace-Stieltjes transform $H_e^{(\mathscr{P})}(\cdot)$ of $T_e^{(\mathscr{P})}$ by using (4) and the statistical independence of $T_{e,I}^{(\mathscr{P})}$ and $T_{e,P}^{(\mathscr{P})}$:

$$\begin{aligned} H_e^{(\mathscr{P})}(s) &= \int_0^{+\infty} e^{-s\tau}\, dF_e^{(\mathscr{P})}(\tau) \\ &= q^{(\mathscr{P})} p_I \int_0^{+\infty} e^{-s\tau} dF_{e,I}^{(\mathscr{P})}(\tau) \\ &\quad + q^{(\mathscr{P})}(1-p_I) \int_0^{+\infty} e^{-s\tau} dF_{e,P}^{(\mathscr{P})}(\tau). \end{aligned} \tag{5}$$

The time T_b corresponds to the time the packet has to wait at the transmitter due to collisions at the MAC layer. The packet is successfully transmitted without any collisions with probability p_s and when this happens $T_b = 0$. Otherwise, the backoff time can be approximated by the sum $\sum_{j=1}^{K} \tau_j$ of independent, exponentially distributed random variables $\{\tau_j,\ j = 1,2\ldots,K\}$ with mean $1/\lambda_b$, each corresponding to a waiting interval after a collision. The number K of collisions experienced by a packet, and therefore, the number of the waiting times τ_j, is distributed according to

$$P\{K = k\} = (1-p_s)^k\, p_s, \quad k = 0,1,2,\ldots. \tag{6}$$

The Laplace-Stieltjes transform $H_b(\cdot)$ of T_b can be computed to be:

$$H_b(s) = p_s \frac{\lambda_b + s}{\lambda_b\, p_s + s}, \quad s < \lambda_b. \tag{7}$$

The transmission time T_t of a packet depends on the packet size. The distribution $F_t(\cdot)$ of the transmission time T_t is a mixture distribution derived from the distributions $F_{t,I}(\cdot)$ and $F_{t,P}(\cdot)$ of the transmission time $T_{t,I}$ of a packet that belongs to an *I*-frame and the transmission time $T_{t,P}$ of a packet that belongs to a *P*-frame, respectively. If p_I is the probability that a packet belongs to an *I*-frame, then

$$\begin{aligned} F_t(\tau) &= P\{T_t < \tau\} \\ &= P\{T_t < \tau, I\text{-frame pck}\} + P\{T_t < \tau, P\text{-frame pck}\} \\ &= p_I P\{T_{t,I} < \tau\} + (1-p_I) P\{T_{t,P} < \tau\} \\ &= p_I F_{t,I}(\tau) + (1-p_I) F_{t,P}(\tau). \end{aligned} \tag{8}$$

We compute the Laplace-Stieltjes transform $H_t(\cdot)$ of T_t by using (8) and the statistical independence of $T_{t,I}$ and $T_{t,P}$:

$$\begin{aligned} H_t(s) &= \int_0^{+\infty} e^{-s\tau}\, dF_t(\tau) \\ &= p_I \int_0^{+\infty} e^{-s\tau} dF_{t,I}(\tau) + (1-p_I) \int_0^{+\infty} e^{-s\tau} dF_{t,P}(\tau). \end{aligned} \tag{9}$$

Assuming the random variables $T_e^{(\mathscr{P})}$, T_b and T_t are mutually independent, the Laplace-Stieltjes transform $H(\cdot)$ of the service time T can be computed from (5), (7) and (9) to be:

$$H(s) = H_e^{(\mathscr{P})}(s)\, H_b(s)\, H_t(s), \quad s < \lambda_b. \tag{10}$$

Special Cases:

Constant encryption and transmission times: If the encryption times $T_{e,I}^{(\mathscr{P})}$ and $T_{e,P}^{(\mathscr{P})}$ for the packets that belong to an I and a P-frame, respectively, are constant such that:

$$T_{e,I}^{(\mathscr{P})} = \mu_{e,I}^{(\mathscr{P})}, \quad T_{e,P}^{(\mathscr{P})} = \mu_{e,P}^{(\mathscr{P})}, \tag{11}$$

then (5) becomes:

$$H_e^{(\mathscr{P})}(s) = q^{(\mathscr{P})} p_I e^{-s\mu_{e,I}^{(\mathscr{P})}} + q^{(\mathscr{P})}(1-p_I)e^{-s\mu_{e,P}^{(\mathscr{P})}}. \qquad (12)$$

Similarly, if the transmission times $T_{t,I}$ and $T_{t,P}$ of the packets that belongs to an I-frame and a P-frame, respectively, are constant such that:

$$T_{t,I} = \mu_{t,I}, \quad T_{t,P} = \mu_{t,P}, \qquad (13)$$

then (9) becomes:

$$H_t(s) = p_I e^{-s\mu_{t,I}} + (1-p_i)e^{-s\mu_{t,P}}. \qquad (14)$$

Accounting for minor variations: If we want to account for minor variations of the encryption and transmission times (seen to occur due to minor variations in packet size in our practical experiments described in Section 6) about some typical values, we can represent these variations by independent Gaussian random variables, such that:

$$T_{e,I}^{(\mathscr{P})} = \mu_{e,I}^{(\mathscr{P})} + r_{e,I}^{(\mathscr{P})}, \qquad T_{e,P}^{(\mathscr{P})} = \mu_{e,P}^{(\mathscr{P})} + r_{e,P}^{(\mathscr{P})}, \qquad (15)$$

where $\mu_{e,I}^{(\mathscr{P})}$ is constant and equal to the time needed to encrypt a packet of size equal to the MTU of the network under the encryption policy $\mathscr{P}$ and $\mu_{e,P}^{(\mathscr{P})}$ corresponds to a typical encryption time for a packet that belongs to a P-frame. The quantity $r_{e,I}^{(\mathscr{P})}$ is a normal random variable with zero mean and variance $(\sigma_{e,I}^{(\mathscr{P})})^2$ that represents small variations in the encryption time of a packet that belongs to an I-frame and is selected for encryption. Similarly, $r_{e,P}^{(\mathscr{P})}$ is a normal random variable with zero mean and variance $(\sigma_{e,P}^{(\mathscr{P})})^2$ that represents variations in the encryption time from packet to packet for the P-frames. Clearly, $T_{e,I}^{(\mathscr{P})} \sim \mathscr{N}(\mu_{e,I}^{(\mathscr{P})}, (\sigma_{e,I}^{(\mathscr{P})})^2)$ and $T_{e,P}^{(\mathscr{P})} \sim \mathscr{N}(\mu_{e,P}^{(\mathscr{P})}, (\sigma_{e,P}^{(\mathscr{P})})^2)$.

Representing the transmission times of packets that belong to I and P-frames in a similar way, we have:

$$T_{t,I} = \mu_{t,I} + r_{t,I}, \qquad T_{t,P} = \mu_{t,P} + r_{t,P}, \qquad (16)$$

where $\mu_{t,I}$ is constant and equal to the time needed to transmit a packet of length equal to the MTU of the network and where $\mu_{t,P}$ corresponds to a typical transmission time for a P-frame packet. The quantity $r_{t,I}$ represents minor random variations in the transmission time of an I-frame packet, modeled as a normal random variable with zero mean and variance $\sigma_{t,I}^2$ and $r_{t,P}$ is a normal random variable with zero mean and variance $\sigma_{t,P}^2$ that represents minor variations in the transmission time of a P-frame packet. Clearly, $T_{t,I} \sim \mathscr{N}(\mu_{t,I}, \sigma_{t,I}^2)$ and $T_{t,P} \sim \mathscr{N}(\mu_{t,P}, \sigma_{t,P}^2)$.

Using the representations in (15) and (16), the Laplace-Stieltjes transforms $H_e^{(\mathscr{P})}(\cdot)$ and $H_t(\cdot)$ of the encryption and transmission times $T_e^{(\mathscr{P})}$ and T_t, respectively, become:

$$\begin{aligned} H_e^{(\mathscr{P})}(s) &= q^{(\mathscr{P})} p_I e^{-\mu_{e,I}^{(\mathscr{P})}s + \frac{1}{2}(\sigma_{e,I}^{(\mathscr{P})})^2 s^2} \\ &\quad + q^{(\mathscr{P})}(1-p_I)e^{-\mu_{e,P}^{(\mathscr{P})}s + \frac{1}{2}(\sigma_{e,P}^{(\mathscr{P})})^2 s^2}, \qquad (17) \\ H_t(s) &= p_I e^{-\mu_{t,I}s + \frac{1}{2}\sigma_{t,I}^2 s^2} + (1-p_I)e^{-\mu_{t,P}s + \frac{1}{2}\sigma_{t,P}^2 s^2}, \qquad (18) \end{aligned}$$

where we used the fact that the Laplace-Stieltjes transform of a normal distribution with mean μ and variance σ^2 is $e^{-\mu s + \frac{1}{2}\sigma^2 s^2}$. Note that we use this second model in our evaluations described in Section 6.

4.2.3 2-MMPP/G/1 Queue Model

The delay experienced by each packet at the sender can be estimated by the 2-MMPP/G/1 queueing model described above. An algorithmic approach that solves the n-MMPP/G/1 queue model is given in [18] and refined in [16] for the case $n = 2$. The algorithm describes a numerical procedure that is shown to converge to the solution of the model. It is based on a general method introduced in [25] and applied in [30] to provide a detailed statistical analysis of the N/G/1 queue.

The method takes as input the infinitesimal generator R and the rate matrix Λ of the MMPP and the Laplace-Stieltjes transform of the service time given by (10). The algorithm computes the distribution function and the moments of the delay seen by the video packets. In particular, the expected value of the queueing delay W is given by

$$\begin{aligned} \mathrm{E}[W] = \frac{1}{2(1-\rho)} \Big[& 2\rho + \mu^{(2)}\pi\lambda \\ & - 2\mu^{(1)}(y + \mu^{(1)}\pi\Lambda)(R + e\pi)^{-1}\lambda \Big], \qquad (19) \end{aligned}$$

where $\rho = \pi\lambda\mu^{(1)}$ is the traffic intensity, $\mu^{(1)}, \mu^{(2)}$ are the first and second moments about the origin respectively, of the service time that can be computed directly from (10), λ is the vector with the diagonal elements of Λ, $e = (1,1)^T$ and the vector y is computed by the algorithm.

4.3 Distortion

There are two parameters that control the video distortion: (i) the packet *decryption rate* p_d and (ii) the decoder *sensitivity* s. The packet decryption rate represents the probability that a packet is received without errors at a node and that the node is able to correctly decrypt the packet. A legitimate receiver has all the necessary information to correctly decrypt packets from the sender; on the other hand an eavesdropper lacks this capability. Therefore, an eavesdropper can only use packets that the sender has decided not to encrypt towards reconstructing the video, when the latter follows a specific encryption policy $\mathscr{P}$. If we denote by $q^{(\mathscr{P})}$ the percentage of packets encrypted by the sender under an encryption policy, then the decryption rates p_d^l and p_d^e of a legitimate receiver and an eavesdropper, respectively, are: $p_d^l = p_s$ and $p_d^e = (1 - q^{(\mathscr{P})})\ p_s$, where p_s is the packet success rate (recall Section 4.1).

The parameter s represents the sensitivity of the decoder to packets that are missing in the receiving stream (either due to interference-induced losses, or in the case of the eavesdropper due to the lack of decryption capabilities). It is the minimum number of packets that the decoder needs to receive (and decrypt) without errors in order to decode the corresponding frame correctly. The sensitivity is associated with the video content itself and specifically with the motion level. When a video flow is characterized by high (or fast) motion, the sensitivity s has a higher value compared to a low (or slow) motion video. This is because in a high motion video flow, the difference between successive frames in the GOP structure is large and the loss of a frame has a higher impact on the overall video quality.

4.3.1 Video Frame Success Rate

We map the packet decryption rate p_d to the video frame (referred to as simply 'frame') success rate P_f, which denotes the probability a frame is successfully received over the wireless link. As was mentioned in Section 2, we assume that each GOP has an $IPP\ldots P$-structure.

If n is the number of packets in each frame, then to successfully decode a frame, (a) the first packet of that frame needs to be received without channel-induced errors and successfully decrypted, and (b) the same should hold true for $0 \le s \le n-1$ of the remaining $n-1$ packets. The success probability of a frame is given by:

$$P_f = p_d \sum_{i=s}^{n-1} \binom{n-1}{i} p_d^i (1-p_d)^{n-1-i}. \tag{20}$$

In general, the I-frame is much larger than a P-frame. As a result, the frame success probabilities for an I and a P-frame also differ. We denote by P_I, the success probability of an I-frame and by P_P, the success probability of a P-frame. We have validated the above model via extensive experiments using the EvalVid tool [2].

4.3.2 *Mean Square Error*

Let the GOP structure contain $G-1$ P-frames that follow the I-frame. We consider predictive source coding where, if the i^{th} frame is the first lost frame in a GOP, then the i^{th} frame and all its successors in the GOP are replaced by the $(i-1)^{st}$ frame at the decoder. If the I-frame of the GOP cannot be decoded correctly, then the entire GOP is considered unrecoverable and is ignored. In this case, these lost video frames are replaced by the most recent frame from a previous GOP that is correctly received. In all cases, the similarity between the missing frames and the reference frame (substitute frame) affects the distortion [33].

We compute the video distortion as the *mean square error* of the difference between the missing frame and the substitute frame. We have the following cases:

Case 1 – Intra-GOP distortion: The I-frame of the current GOP is successfully received. The distortion for the current GOP depends on which, if any, of the P-frames of the GOP cannot be decoded without errors. If the first unrecoverable P-frame is the i^{th} frame in the GOP, the corresponding distortion is given by [22]:

$$d_i = (G-i)\frac{i \cdot G \cdot d_{min} + (G-i-1)\cdot d_{max}}{(G-1)\cdot G}, \tag{21}$$

for $i = 1,2,\ldots,(G-1)$, where $d_{\max}$ is the maximum distortion when the first frame is lost and d_{min} is the minimum distortion when the last frame is lost. The values of $d_{\max}$ and $d_{\min}$ can be estimated given the probability of a packet loss. The probability P_i that the i^{th} frame is lost is

$$P_i = P_I P_P^{i-1}(1-P_P), \quad i = 1,2,\ldots,(G-1). \tag{22}$$

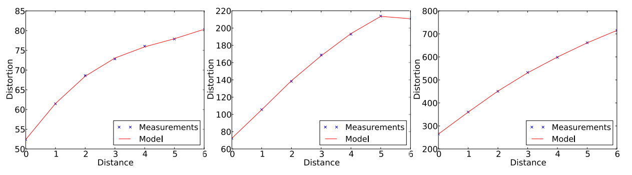
Using (21) and (22), the expected value of the distortion can be computed to be: $D^{(1)} = \sum_{i=1}^{G} d_i \cdot P_i$

Case 2 – Inter-GOP distortion: The I-frame of the current GOP is lost and a frame from a previous GOP is used as the reference frame. Here, the difference between the reference frame and the missing frames determine the distortion.

Similar to the work in [33], we expect to see the motion characteristics of the video affecting distortion. To capture the dependence of the inter-GOP distortion on the motion level of the video we perform a set of experiments and use the collected results to statistically describe this association.

Specifically, we select a set of video streams from [11] and categorize them into three groups according to their motion level: low, medium and high, using the tool in [1]. All video streams have 300 frames each, with a frame rate of 30 frames per second. We use FFmpeg [3] to convert the video streams from the initial, uncompressed YUV format to the MP4 format. Then, we artificially create video frame losses in order to achieve reference frame substitutions from various distances. Finally, we use the Evalvid toolset [2] to measure the corresponding video distortion.

In Fig. 2 the dependence of the average distortion on the distance between the missing frame and the substitute is shown for



(a) Low motion (b) Medium motion (c) High motion

Figure 2: Average distortion with distance.

the three categories. In order to use these empirical results in other experiments, we approximate the observed curves with polynomials of degree 5 using a multinomial regression (use of higher degree polynomials does not increase accuracy). In particular, we define the approximate distortion $D^{(2)}$ as a function of the distance d: $D^{(2)}(d) = \sum_{i=0}^{5} a_i d^i$, and compute the coefficients $a_0,\ldots,a_5$, using the regression.

Case 3 – Initial GOP: The I-frame of the current and all previous GOPs (including the first GOP) are lost. In this case the distortion D is maximized. If $\{D_{\max}^{(1)}, D_{\max}^{(2)}, \ldots, D_{\max}^{(\|\mathscr{G}\|)}\}$, where $\mathscr{G}$ is the set of all GOPs in the video flow, is the set of the maximum distortion values in all GOPs, then $D^{(3)} = \max_{k\in\mathscr{G}} D_{\max}^{(k)}$.

4.3.3 *Computing Average Distortion*

Suppose the video flow has N GOPs and each GOP consists of an I-frame followed by $G-1$ P-frames. For each GOP of the flow define the state $S_i, i = 1,2,\ldots,N$, such that $S_i \in \mathscr{S} = \{0,1,\ldots,G\}$. The state S_i for the i^{th} GOP indicates which is the first unrecoverable frame in that GOP. Specifically,

$$S_i = \begin{cases} 0, & I\text{-frame is lost,} \\ k, & k^{th}\ P\text{-frame is lost, } 1 \le k \le (G-1), \\ G, & \text{none of the frames is lost,} \end{cases} \tag{23}$$

for $i = 1,2,\ldots,N$. The initial state for each GOP is G. The transition probability $p_i(G,g)$ of state S_i from G to $g \in \mathscr{S}$ is

$$p_i(G,g) = \begin{cases} 1-P_I, & g = 0, \\ P_I P_P^{k-1}(1-P_P), & g = k, 1 \le k \le (G-1), \\ P_I P_P^{G-1}, & g = G, \end{cases} \tag{24}$$

for $i = 1,2,\ldots,N$.

To compute the expected value of the distortion for the transmission of the video stream over the wireless channel we need to consider the states of all the GOPs. We define the vector $S = (S_1,S_2,\ldots,S_N) \in \mathscr{S}\times\mathscr{S}\times\cdots\times\mathscr{S}$. The initial state of S is $G = (G,G,\ldots,G)$ and its transition probability $p(G,g)$ to a new state $g = (g_1,g_2,\ldots,g_N)$ is

$$p(G,g) = \prod_{i=1}^{N} p_i(G,g_i). \tag{25}$$

The overall distortion for the video stream transmission depends on the final state g. As was discussed earlier in Case 2, the distortion of a GOP may depend not only on missed frames in that GOP but on frames that are missing in previous GOPs as well. Therefore, if D_i is the distortion of the i^{th} GOP, it is a function of the vector g and not only of the i^{th} component of g. We define the random variable $D(g) = (D_1(g), D_2(g), \ldots, D_N(g))$ consisting of the distortions of each of the GOPs of the video flow. Using (25) we

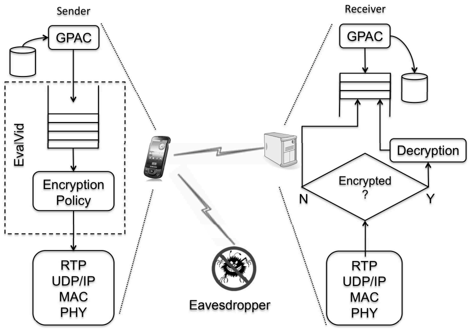


Figure 3: Block diagram for sender and receiver.

have:

$$\mathrm{E}[D] = (\mathrm{E}[D_1], \ldots, \mathrm{E}[D_N]) = \sum_g p(G,g)D(g) \qquad (26)$$

The average distortion that corresponds to the video file is

$$\bar{D} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{E}[D_i]. \qquad (27)$$

4.3.4 *Mapping Distortion to PSNR*

In all the results we present in the sequel, we use the Peak Signal-to-Noise Ratio (PSNR) which is an objective video quality measure [14]. The relationship between distortion and PSNR (in dB) is given by [14]:

$$\mathrm{PSNR} = 20 \cdot \log_{10} \frac{255}{\sqrt{\mathrm{Distortion}}} \qquad (28)$$

The goal then is to encrypt enough frames to drive the PSNR to be as low as possible at an eavesdropper's site.

5. IMPLEMENTATION

We implement a software framework on the Android NDK and SDK that allows us to test various encryption policies. To achieve this we use GPAC [6] and the EvalVid [2] toolset. The former is a cross-platform open source multimedia framework which provides support for creating, parsing and streaming multimedia packaging formats, such as MP4. The latter provides tools for evaluating the quality of video which is transmitted over a network. Our application modifies the EvalVid tool to read and then securely transmit a video stream according to the selected encryption policy.

With our application, the user selects which video to transmit, the receiver and the encryption policy. The application uses the GPAC library to read the video from the disk into the internal memory. There are two threads that access the memory; the producer thread reads the video segments from the disk and stores them in a queue, and the consumer thread reads the segment from the head of the queue and forwards it to the block of code that implements the encryption policy selected by the user. Our code checks whether the video segment satisfies the encryption selection rule defined by the policy in effect or not. If it does, it uses the GPAC API to encrypt the segment according to the encryption algorithm (AES128, AES256, 3DES) using the *Output Feedback Mode* (OFB). The OFB encryption mode is applied to each segment separately, and therefore a possible error at the receiver does not propagate to the following segments during the decryption process. By default, we assume the use of RTP and UDP; we discuss experiments with HTTP/TCP transfers in Section 6.4. Whether the video segment is encrypted or not, it is encapsulated in an RTP packet. In the case that encryption has been performed, the Marker Bit in the RTP header is set denoting the event to the receiver. The RTP packet is transmitted over UDP to the receiver.

Table 1: Experimental Setup

Frame Size	CIF (352x288)
GOP Size	30, 50
Video Motion	slow-motion, fast-motion
Encryption Algorithm	AES128, AES256, 3DES
Encryption Level	none, I-frame, P-frame, all
Wireless Devices	Samsung Galaxy S-II, HTC Amaze 4G
Android Version	Ice Cream Sandwich (4.0)

Upon the reception of an RTP packet, the receiver checks the Marker Bit in the RTP header to decide if the RTP payload is encrypted. If the Marker Bit is set, the receiver uses the GPAC API to decrypt the packet according to the encryption policy. The received packets are then combined in order to reconstruct the MP4 video file. Fig. 3 depicts the operations performed at the sender and receiver.

The eavesdropper (see Fig. 3) overhears the transmission on the channel by using `tcpdump` on his rooted phone or laptop. Only the unencrypted packets can be used towards reconstructing the overheard video stream. Because of this, the eavesdropper experiences significantly higher video transmission distortion than that at the legitimate receiver.

6. EVALUATION

This section demonstrates the viability of the approach, quantifies the trade-off between the transfer delay, and the distortion at an eavesdropper's site and discusses the impact of the video type (slow vs fast motion) on the mode of encryption needed. Experimental results on the battery savings with different modes of encryption are also presented. Results with HTTP/TCP are presented at the end of the section.

6.1 Methodology

We validate our analysis through extensive experiments using smartphones running our Android application over WiFi connections (IEEE 802.11g). Table 1 lists the parameters considered.

Wireless devices: All the experiments are repeated on two different smartphones, viz., (i) the Samsung Galaxy S-II that has a 1.2 GHz dual-core ARM Cortex-A9 CPU, an ARM Mali-400 MP4 GPU, and 1 GB RAM and (ii) the HTC Amaze 4G equipped with a 1.5 GHz dual core Qualcomm Snapdragon S3 CPU, Adreno 220 GPU and a 1 GB RAM. Both devices run Android 4.0 (Ice Cream Sandwich).

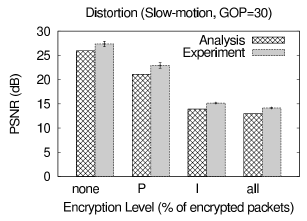
Strategies tested: Twelve encryption policies are tested; they consist of all possible combinations of three different encryption algorithms and four modes of packet encryption. In particular, the three symmetric key encryption algorithms that are considered are the AES128, AES256 [31] and 3DES [12]. Due to space constraints we only show the results for AES256 and 3DES. The results for the AES128 encryption algorithm are similar and follow the same trends. The complete set of results is in our technical report [26]. For the packet encryption selection rules, we consider the two extreme cases where either all or none of the packets are encrypted. We also consider the case where only the packets that belong to an *I*-frame are encrypted and the case where only the packets that belong to *P*-frames are encrypted. Finally, we consider encrypting the *I*-frames *and* different fractions of the *P*-frames. We did limited experiments with other possibilities (only partial encryption of *I*-frames) but did not pursue these beyond that since the behavioral results could be extrapolated based on the results that we present here.

Types of video flows: The experiments are performed on two kinds of video flows: slow-motion and fast-motion video flows. Slow-motion video flows are characterized by slow changes from picture-frame to picture-frame and therefore the size of the *P*-frames in each GOP are typically small (tens to hundreds of bytes). On the other hand, fast-motion video flows contain rapid changes between picture-frames having as a result larger *P*-frames. We use the AForge [1] tool to dynamically categorize the motion level in different parts of the video clip. The motion level of a video flow affects not only the GOP structure (i.e. percentage of *I*-frame and *P*-frame packets in the GOP structure) but also the sensitivity of the video decoder to the packet loss ratio. A fast-motion video flow is more susceptible to packet losses and therefore the distortion at the receiver (or eavesdropper) can be naturally higher compared to the case where a comparable in size, slow-motion video flow is transmitted over the same wireless link. All video flows are of the same picture-frame size (CIF-352x288 pixels) and are encoded using the publicly available `x264` [10] software library and application into different GOP sizes (30, 50 frames).

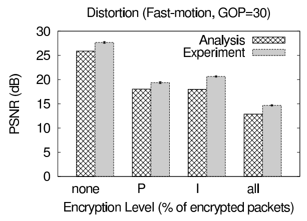
Experimental methodology: We use the Android application that we have developed to measure the delay due to the encryption and the EvalVid toolset to compute the distortion at the eavesdropper. We also run `tcpdump` on the wireless device to capture the time when each packet is transmitted over the wireless link. EvalVid supports performance metrics such as the Peak Signal to Noise Ratio (PSNR) [14], which we use to represent video quality. Note that the lower the PSNR, the higher the distortion.

To compute the delay and distortion we follow a sequence of steps: we start with the initial uncompressed video files which consist of a sequence of YUV [20] frames. Using the EvalVid toolset, we transform the YUV format, first to the H.264 format and then to a MP4 video file. Next, we use the Android application we have developed to transmit the video stream to the legitimate receiver. During this phase, we select the set of packets to be encrypted based on the encryption policy that is in effect. We keep track of the time instances at which each packet reaches different parts of our application. These statistics include the time instances when the packet enters and leaves the queue that is shown in Fig. 3, the time duration needed to encrypt the packet, in case this packet is selected for encryption, and the time instance when the packet is forwarded to the transport layer. Furthermore, we use `tcpdump` to capture the time instance the packet is transmitted over the wireless link. At the legitimate receiver, all the successfully received packets are used to reconstruct the initial video using the EvalVid toolset. At the eavesdropper, only the successfully received unencrypted packets contribute in the reconstruction of the pilfered video stream. Encrypted packets are treated as erasures. Each experiment is repeated 20 times and the values of the queueing delay and distortion are used to compute the averages and the 95% confidence intervals.

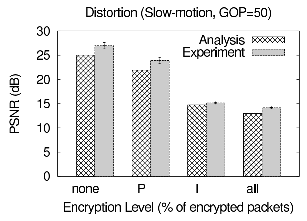
Applying the mathematical framework: We use an initial sequence of events to tune the parameters of our mathematical model. The times of insertion of video segments into the internal queue (see Fig. 3) and their type (*I*, *B*-frames) are used to estimate the 2-MMPP parameters, R and Λ in (1). The sequence of times that are necessary for the encryption of an initial set of packets and the fraction of packets that are encrypted, are used to estimated the mean and variance of the encryption time T_e. Similarly, the observation of the transmission of an initial set of packets can provide



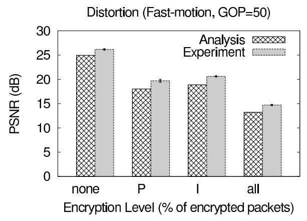
(a) Slow-motion, GOP=30.



(b) Fast-motion, GOP=30.

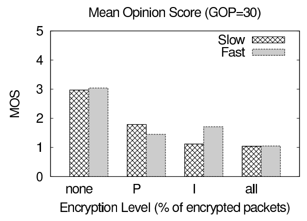


(c) Slow-motion, GOP=50.

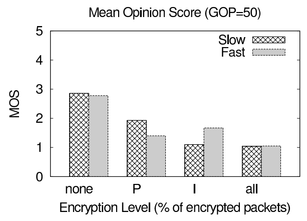


(d) Fast-motion, GOP=50.

Figure 4: Distortion at an eavesdropper's site for slow and fast motion video flows.

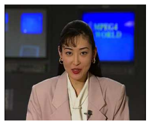
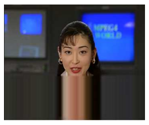
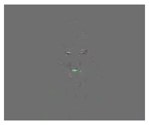
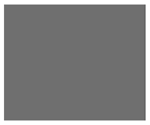


(a) GOP=30.



(b) GOP=50.

Figure 5: Mean Opinion Score at an eavesdropper's site for slow and fast motion video flows.


(a) Slow,none.

(b) Slow, P.

(c) Slow, I.

(d) Slow, all.

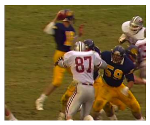
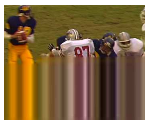
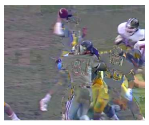
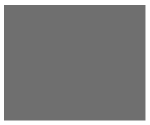
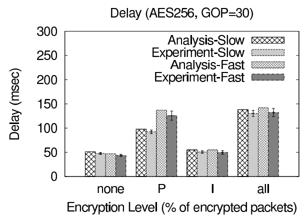

(e) Fast, none.

(f) Fast, P.

(g) Fast, I.

(h) Fast, all.

Figure 6: Screenshots of video flow at an eavesdropper's site (slow vs fast, GOP=30).

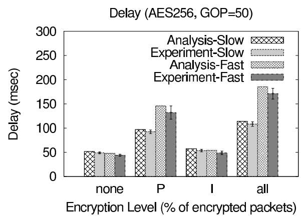
estimates for the mean and variance of the transmission time T_t and the parameter λ_b for the backoff time T_b, characterizing this way the service time T in (3). Note, the client has access locally to all the necessary information to compute these estimates.

6.2 Delay vs Distortion

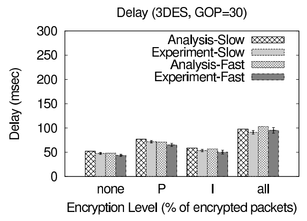
Since the legitimate receiver is capable of decrypting the packets the distortion is only affected by the packet loss ratio on the wireless link. The video distortion at the eavesdropper also depends on the percentage of packets that are encrypted at the sender according to the specific encryption policy that is in use. In order to compute the distortion at each end we use the EvalVid tools to



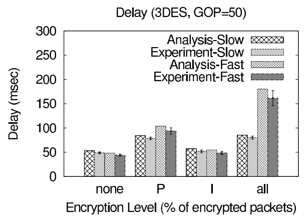
(a) AES256, GOP=30.



(b) AES256, GOP=50.

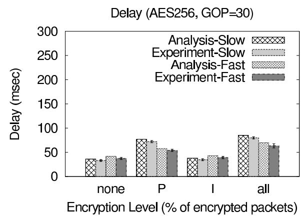


(c) 3DES, GOP=30.

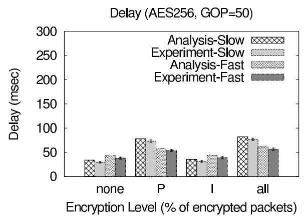


(d) 3DES, GOP=50.

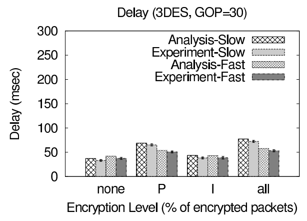
Figure 7: Comparison of transfer latency in various cases (analysis and experiments with Samsung S-II).



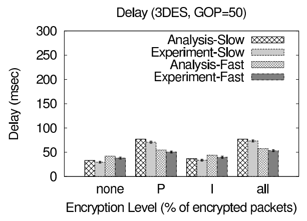
(a) AES256, GOP=30.



(b) AES256, GOP=50.



(c) 3DES, GOP=30.



(d) 3DES, GOP=50.

Figure 8: Comparison of transfer latency in various cases (analysis and experiments with HTC Amaze 4G).

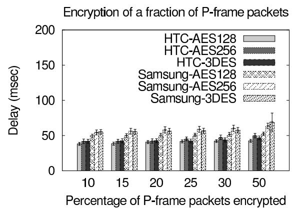
reconstruct the YUV file based on the successfully received and decrypted packets.

Distortion at an eavesdropper due to the encryption of *I* and *P* -frame packets: The distortion results are shown in Fig. 4 for both slow and fast-motion video flows. The factor that determines the video distortion is the percentage of packets that are correctly received and successfully decrypted. The legitimate receiver decrypts all the packets successfully delivered over the channel and the distortion here corresponds to the first bar in each plot, labeled "none". In contrast, the eavesdropper experiences higher distortion since it cannot correctly decrypt packets. As a general observation, the analytical results closely match the experimental results. The encryption of *I*-frame packets plays a more significant role in degrading the video quality (up to 80%) at the eavesdropper compared to the case where only *P*-frame packets are encrypted (the largest decrease observed here is 40%). This is to be expected since the *I*-frames carry a lot more information regarding the video content. Moreover, the encryption of *I*-frame packets degrades the video quality at the eavesdropper to a greater extent for the case of slow-motion video (80%) compared to the fast-motion video (30%). This is because the *I*-frames carry most of the information in the former case. The loss of *P*-frames affects video with fast motion to a higher extent, since in this case, these frames carry a lot more information (as compared to slow motion video flows). The Mean Opinion Score (MOS), which is a subjective metric that represents the quality of the video at the eavesdropper's site is given in Fig. 5, while Fig. 6 contains video screenshots as seen at the eavesdropper's site, for both slow and fast motion video flows. The Mean Opinion Score (MOS) gives a numerical indication of the perceived quality of the received video clip. It is expressed as a number from 1 to 5, where 1 indicates bad quality and 5 the best quality. Although MOS is subjective, there is software that measures the MOS on network transfers. For our experiments we report MOS values as measured by the EvalVid toolset. Note that the MOS drops to the lowest levels (≈ 1) with partially encrypted flows. This essentially implies that the video is practically unviewable by the eavesdropper.

Latencies with *I* and *P* frame encryption: Figures 7 and 8 show the average delay per packet for each device, GOP size, motion level and encryption policy. A general observation is that the incurred delay when the *P*-frame packets are encrypted is larger than the delay for the case where *I*-frame packets are encrypted. In the case of the HTC Amaze 4G, this delay is almost equal to the extreme case where all the packets in the transmission are encrypted. The same is true for Samsung S-II for the AES256 encryption algorithm, but not for the 3DES encryption scheme. Furthermore, the delay in the case where the *I*-frame packets are selected for encryption is small and close to the delay when none of the packets are encrypted.

Table 2: Delay vs distortion.

	Delay	**PSNR**	**MOS**
I	48.41 msec	20.65 dB	1.71
I* + 10% *P	53.06 msec	17.8684 dB	1.26
I* + 15% *P	53.90 msec	17.6895 dB	1.24
I* + 20% *P	54.91 msec	17.3359 dB	1.20
I* + 25% *P	55.47 msec	17.1776 dB	1.17
I* + 30% *P	56.51 msec	16.4268 dB	1.15
I* + 50% *P	61.76 msec	16.0106 dB	1.14



(a) Upload latency.

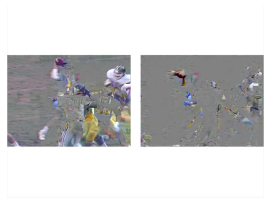
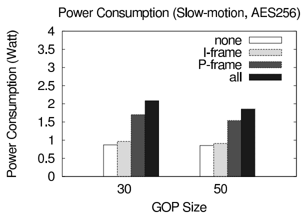
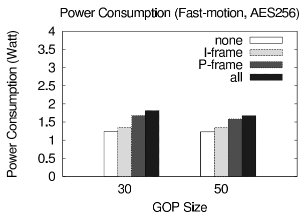

(b) Screenshots for *I* (left) and *I*+20%*P* (right) case.

Figure 9: Encrypting all *I*-frame and a fraction of the *P*-frame packets (GOP=30).

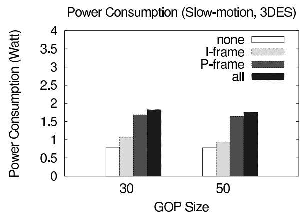
Finer control of protection for fast-motion video: An encryption policy where we encrypt a mixture of *I* and *P* frame packets can provide a finer control over the protection levels of the content. Going back to Figs. 4b and 4d, we observe that for fast motion video flows, the distortion at the eavesdropper is lower when we encrypt *I*-frame packets compared to the case where *P*-frame packets are encrypted. This is in contrast to what happens in the case of slow



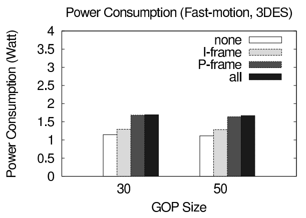
(a) Slow-motion, AES256.



(b) Fast-motion, AES256.

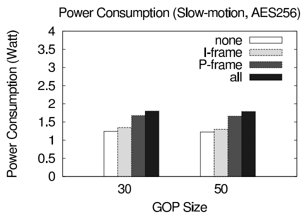


(c) Slow-motion, 3DES.

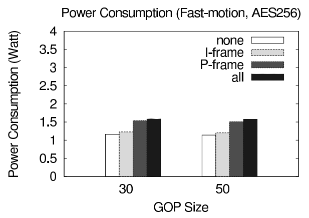


(d) Fast-motion, 3DES.

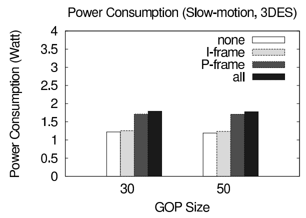
Figure 10: Power consumption with Samsung S-II.



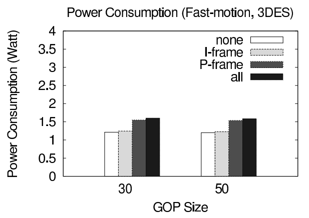
(a) Slow-motion, AES256.



(b) Fast-motion, AES256.



(c) Slow-motion, 3DES.



(d) Fast-motion, 3DES.

Figure 11: Power consumption with HTC Amaze 4G.

motion video flows. However, the delay for encrypting *I*-frame packets is lower than the delay that the *P*-frame packet encryption incurs (given the much larger volume of *P* frames). To achieve a better trade-off between delay and distortion, we examine the case where we encrypt all the *I*-frame packets and a fraction α, of the *P*-frame packets in a GOP for fast motion video. We experiment with different values of α and we show in Fig. 9a the corresponding transfer latency for each encryption algorithm and wireless device. Table 2 shows the delay, distortion and the Mean Opinion Score for Samsung S-II. We observe that the minimum value of α that provides an almost complete obfuscation of the video flow due to distortion is 20%. For that value of α, the power consumption is 1.48 Watt, while the power consumption is 1.28 Watt when only *I*-frame packets are encrypted (power consumption is discussed in detail later in Section 6.3). The change in delay due to this additional encryption is only about 6.5msec. Figure 9b depicts screenshots at the eavesdropper's site in the case where only the *I*-frames are encrypted (left) and the mixture of *I* and 20% of *P*-frame packets encryption (right).

For slow motion video flows we observe from Figs. 4a and 4c that encrypting all *I*-frame packets results in a high distortion, to almost make the content invisible, at an eavesdropper's site. In order to save on energy consumption and delay, we examined the case where half of the *I*-frame packets are encrypted. We found that the distortion levels are similar to the case where all the *P*-frame packets are encrypted and thus does not provide adequate obfuscation.

6.3 Power Consumption

To compute the power consumption we use the power monitor tool by Monsoon Solutions, Inc. and measure the amount of energy the mobile phone consumes during the video streaming. The reading v from the power monitor is in μAh which we convert into Watts as follows:

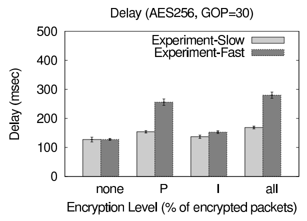
$$\frac{v \cdot \text{Voltage} \cdot 3600}{\text{stream duration}} \cdot 10^{-6}; \tag{29}$$

the Voltage is set to 3.9 Volts.

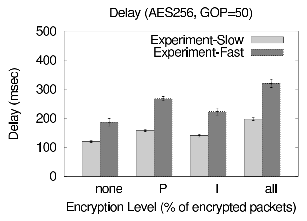
Due to the different sizes of the slow and fast motion video flows, we do not compare the power consumption between them; instead, we perform the comparison within the same type (slow or fast motion) of flows but with different encryption policies. The power consumption measurements for the Samsung Galaxy S-II phone are shown in Fig. 10, while those for the HTC Amaze 4G are in Fig. 11. The results are for slow and fast motion video and for the three encryption algorithms, for each GOP size. As can be seen, when the video stream is unencrypted the energy consumption is the lowest due to the fact that fewer CPU cycles are needed in order to process a frame. On the other hand, a fully encrypted stream consumes the highest amount of energy. Furthermore, more energy is necessary when only the *P*-frames are encrypted compared to the case where only the *I*-frames are selected for encryption. This is so because the overall size of the *P*-frame packets together is larger than the overall size of the *I*-frame packets together. Considering the Samsung S-II, and for a slow motion video, an increase in the power consumption by 140% can be seen comparing the two extreme cases where none of the packets are encrypted and all the packets are encrypted. If only the *I*-frames are encrypted, the increase is only 11%. This translates to a savings of 92%. The power consumption increase for a fast motion video flow is lower, where the largest increase (by 50%) in the power consumption is observed when all the packets are encrypted. For the HTC Amaze 4G the increase in the power consumption is not as steep; the largest increase is by 50% and 38%, for the slow motion and fast motion video, respectively. For fast motion video, when all the *I*-frames and 20% of the *P*-frames are encrypted (to provide almost complete confidentiality), we find the energy savings to be 26% (reduction from 2 Watts to 1.48 Watt).

6.4 Experiments with HTTP/TCP

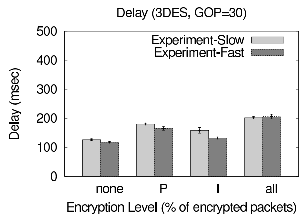
Next, we experimentally evaluate selective encryption for video traffic based on HTTP/TCP. A Marker bit is used again (in the option header) to indicate whether or not a packet is encrypted. The average delay per packet is shown in Fig. 12 and Fig. 13 for the Samsung S-II and the HTC Amaze 4G phones, respectively. The distortion and the mean opinion score for both the slow and fast motion video flows are shown in Fig. 14 and Fig. 15, respectively. The trend that is observed when RTP/UDP are used is also seen for the case of HTTP/TCP. While the latency is slightly higher (due to TCP retransmissions), it is reduced significantly, especially for fast motion video where the volume of packets is more. Since the fraction of packets encrypted remain the same, the energy benefits are



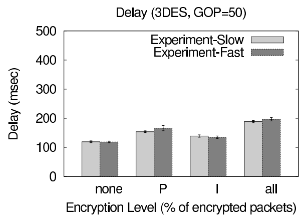
(a) AES256, GOP=30.



(b) AES256, GOP=50.

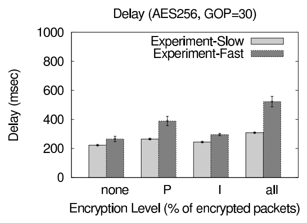


(c) 3DES, GOP=30.

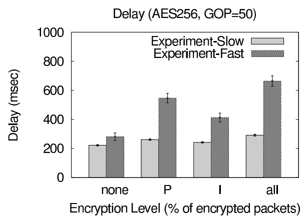


(d) 3DES, GOP=50.

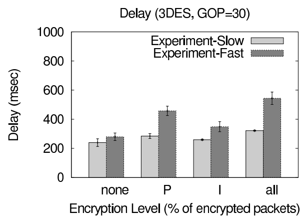
Figure 12: Comparison of transfer latency for HTTP/TCP (Samsung S-II).



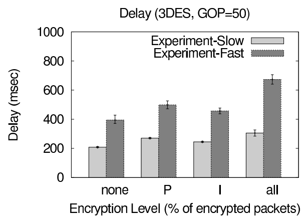
(a) AES256, GOP=30.



(b) AES256, GOP=50.



(c) 3DES, GOP=30.



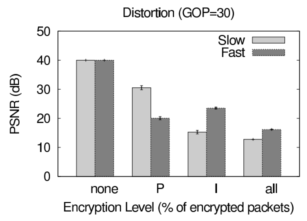
(d) 3DES, GOP=50.

Figure 13: Comparison of transfer latency for HTTP/TCP (HTC Amaze 4G).

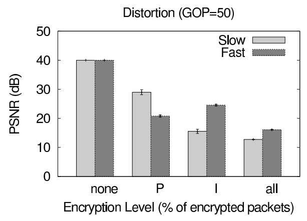
identical to that with UDP/RTP; thus, we do not present these plots here.

7. CONCLUSIONS

Due to the widespread use of smartphones, video transfers over WiFi connections are becoming increasingly popular. We argue that only encrypting parts of a video flow can sufficiently distort the stream at an eavesdropper's site and thus render the content useless; at the same time such approaches can reduce performance penalties in terms of delay and energy. We refer to encrypting different parts of the stream as different modes of encryption. We develop a mathematical framework to characterize the effect of different modes of encryption on the delay at the client and the distortion at an eavesdropper's site. The framework provides an efficient way of determining the volume of video traffic that needs to be encrypted to preserve confidentiality at minimum performance cost.

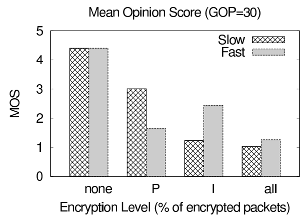


(a) GOP=30.

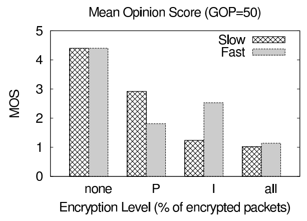


(b) GOP=50.

Figure 14: Distortion at an eavesdropper's site for slow and fast motion video flows with HTTP/TCP.



(a) GOP=30.



(b) GOP=50.

Figure 15: Mean Opinion Score at an eavesdropper's site for slow and fast motion video flows with HTTP/TCP.

We validate our model via extensive experiments using Android smartphones.

8. ACKNOWLEDGMENT

This work was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

9. REFERENCES

[1] AForge.NET. `http://www.aforgenet.com/framework/features/motion_detection/_2.0.html`.
[2] EvalVid with GPAC. `http://www2.tkn.tu-berlin.de/research/evalvid/EvalVid/docevalvid.html`.
[3] FFmpeg. `http://ffmpeg.org/`.
[4] Firesheep. `http://codebutler.com/firesheep/`.
[5] Google apps documentation and support. `http://support.google.com/a/bin/answer.py?hl=en&answer=1279090`.
[6] GPAC. `http://gpac.wp.mines-telecom.fr/`.
[7] A guide to sniffing out passwords and cookies. `http://lifehacker.com/5853483`.
[8] Report: Mobile uploads up fourteen-fold. `http://www.pcmag.com/article2/0,2817,2392007,00.asp`.
[9] What's APPening wtih Apple FaceTime. `http://researchcenter.paloaltonetworks.com/2010/08/whats-appening-with-apple-facetime/`.
[10] x264. `http://www.videolan.org/developers/x264.html`.
[11] YUV CIF reference videos (lossless H.264 encoded). `http://www2.tkn.tu-berlin.de/research/evalvid/cif.html`.

[12] American National Standards Institute. *ANSI X3.92-1981 American National Standard, Data Encryption Algorithm*, 1981.

[13] J. S. Baras, V. Tabatabaee, G. Papageorgiou, and N. Rentz. Performance metric sensitivity computation for optimization and trade-off analysis in wireless networks. In *IEEE GLOBECOM*, 2008.

[14] A. C. Bovik. *The Essential Guide to Video Processing*. Academic Press, 2009.

[15] D. S. A. Elminaam et al. Performance evaluation of symmetric encryption algorithms. *IJCSNS International Journal of Computer Science and Network Security*, 8(12):280–286, Dec. 2008.

[16] W. Fischer et al. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18(2):149–171, Sept. 1993.

[17] M. Garetto, T. Salonidis, and E. W. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *IEEE INFOCOM*, 2006.

[18] H. Heffes and D. M. Lucantoni. A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE J. Sel. Areas Commun.*, 4(6), Sept. 1986.

[19] O. C. Ibe. *Markov Processes for Stochastic Modeling*. Academic Press, 2008.

[20] International Telecommunications Union. *ITU-R BT.601: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*.

[21] ISO/IEC JTC1/SC29/WG11. ISO/IEC 14496 – Coding of audio-visual objects. `http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm`.

[22] M. T. Ivrlač, R. L.-U. Choi, E. G. Steinbach, and J. A. Nossek. Models and analysis of streaming video transmission over wireless fading channels. *Signal Proessing: Image Communication*, 24(8):651–665, Sept. 2009.

[23] D. Li and J. Pan. Performance evaluation of video streaming over multi-hop wireless local area networks. *IEEE Trans. Wireless Commun.*, 9(1):338–347, Jan. 2010.

[24] T. Lookabaugh and D. C. Sicker. Selective encryption for consumer applications. *IEEE Commun. Mag.*, 42(5):124–129, May 2004.

[25] M. F. Neuts. A versatile Markovian point process. *Journal of Applied Probability*, 16(4):764–779, Dec. 1979.

[26] G. Papageorgiou, J. Gasparis, S. V. Krishnamurthy, R. Govindan, and T. L. Porta. Securing mobile video uploads from eavesdroppers with minimum performance penalties: Tech report. `http://www.cs.ucr.edu/~gpapag/tech-report-encryption.pdf`, June 2013.

[27] G. Papageorgiou, S. Singh, S. V. Krishnamurthy, R. Govindan, and T. L. Porta. Distortion-resilient routing for video flows in wireless multi-hop networks. In *IEEE ICNP*, 2012.

[28] P. Prasithsangaree and P. Krishnamurthy. Analysis of energy consumption of RC4 and AES algorithms in wireless LANs. In *IEEE GLOBECOM*, 2003.

[29] M.-R. Ra, R. Govindan, and A. Ortega. P3: Toward privacy-preserving photo sharing. In *USENIX NSDI*, 2013.

[30] V. Ramaswami. The N/G/1 queue and its detailed analysis. *Advances in Applied Probability*, 12(1):222–261, Mar. 1980.

[31] United States National Institute of Standards and Technology (NIST). *Announcing the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197.*, Nov. 2001.

[32] W. Wang, M. Hempel, D. Peng, H. Wang, H. Sharif, and H.-H. Chen. On energy efficient encryption for video streaming in wireless sensor networks. *IEEE Trans. Multimedia*, 12(5):417–426, Aug. 2010.

[33] Y. Wang, M. Claypool, and R. Kinicki. Impact of reference distance for motion compensation prediction on video quality. In *Proceedings of ACM/SPIE Multimedia Computing and Networking (MMCN 2007)*, 2007.

[34] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):560–576, July 2003.