AN EXPERIMENTAL STUDY ON ENERGY CONSUMPTION OF VIDEO ENCRYPTION FOR MOBILE HANDHELD DEVICES[*]

Kyoungwoo Lee, Nikil Dutt, and Nalini Venkatasubramanian

School of Information and Computer Science
University of California at Irvine
{kyoungwl,dutt,nalini}@ics.uci.edu

Abstract Secure video communication on mobile handheld devices is challenging mainly due to (a) the significant computational needs of both video coding and encryption algorithms and (b) the limited battery capacity of handheld devices. In this paper, we evaluate several video encryption schemes from the perspective of energy consumption both analytically and experimentally. Specifically, we implement video encryption schemes on mobile handhelds to support a H.263 based secure video application, and extensively measure the energy consumption due to encoding and encryption for several classes of video clips. Contrary to popular belief, our experiments show that energy overhead of full video encryption is insignificant compared to the energy consumed for video encoding (between 2% and 4% of total energy cost) in most cases.

1. MOTIVATION

Interactive multimedia applications on mobile handhelds are gaining popularity in a variety of domains. Video applications executing on battery-operated handheld devices consume significant amounts of energy due to the high complexity of video processing (encoding and decoding). Recent work in power-aware mobile multimedia focuses on addressing the communication computation tradeoffs involved in preserving video QoS while increasing device lifetimes. Ubiquitous wireless networks supporting video communication are vulnerable to a host of security attacks that can compromise both the quality and integrity of the communication. Indeed, for many critical applications – military services, financial transactions, and crisis communications to name a few – specific bitstreams must be encoded and protected using cryptographic techniques.

Over the past decade, several efforts [1, 2, 3, 4] have developed video encryption schemes for secure transfer of multimedia information. Many of these approaches exploit the inherent characteristics of video data and the steps taken to compress and encode video information. The typical steps performed in video compression (e.g., MPEG and H.263) include motion estimation, DCT transform, quantization, and entropy encoding [5]. Most video encryption techniques are applied to encoded bitstreams after video encoding, specifically after the step of entropy encoding, and video decryption is performed right before video decoding. Some algorithms [4] have also proposed encryption between the quantization step and entropy encoding in a video encoder. Figure 1 depicts the conventional steps in a secure video application. Qiao and Nahrstedt [2] compare video encryption schemes in terms of security, speed, and size.

Energy optimization is an additional concern when one of the endpoints of the secure video communication is a mobile handheld device such as a PDA or a cellular phone. Since both video encryption and video encoding are expensive operations in terms of computation, optimizing energy while ensuring security of data transfer and guaranteeing video QoS is an important issue. To the best of our knowledge, the impact of energy limitations simultaneously on both security and QoS has been neither widely recognized nor adequately addressed in many of prior works. This paper attempts to bridge that gap by investigating the energy consumption characteristics of several video encryption algorithms both analytically and experimentally. We implement a popular video conferencing standard, H.263, on a mobile handheld in order to profile power usage of several video encryption (and decryption) algorithms. The objective of this study was to develop techniques to prolong battery life for secure mobile multimedia applications without sacrificing user experience.

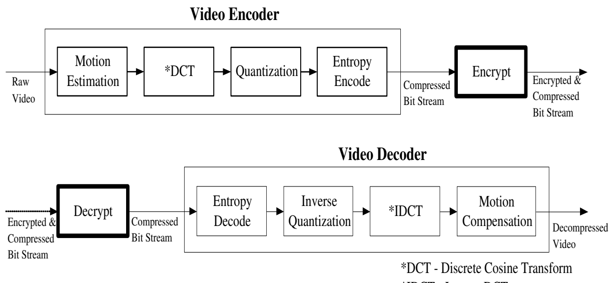


Fig. 1. Secure Video Application Steps

This paper is organized as follows: Section 2 analyzes the computational needs of existing video encryption techniques - in theory, this has a one-to-one correspondence

*This work was supported in part by NSF award ACI-0204028

with the energy consumption characteristics. Section 3 describes our experimental approach to evaluating the energy overheads for video encryption in a practical setting and Section 4 presents experimental results. We conclude in Section 5 with a discussion on the relevance of this work.

2. ENERGY ANALYSIS

In this section, we present a simple but intuitive analysis of the energy consumption of various video encryption techniques. Before analyzing the energy consumption characteristics of the proposed video encryption techniques, we summarize the notations:

S_T (in Bytes)	size of total encoded bit streams
S_I, S_P (in Bytes)	size of *Intra*-frames and *Inter*-frames, resp.
S_{ib} (in Bytes)	size of Intra-blocks in S_P
e_{DES} (in Joules)	amount of energy to encrypt 1-byte data using DES
E_X (in Joules)	amount of energy to encrypt an encoded video using an $algorithmX$
E_{encd} (in Joules)	amount of energy to encode a raw video file

Since energy consumption of symmetric ciphers such as DES [6] increases proportionally with the data size [7], we can compute the total energy consumption to encrypt a video clip based on the specifics of the video encryption scheme, the size of data, and e_{DES} if DES is applied for encryption.

The Naive Algorithm encrypts the entire video stream by treating it like a text stream. We can directly compute energy consumption for encrypting video data as follows:

$$E_{Naive} = e_{DES} * S_T$$

The Selective Algorithm exploits features of layered structures in encoded video - in particular, the frame structure. The analysis below assumes a more rigorous form of the Selective Encryption Approach. Since encrypting only Intra-coded frames (I-frames) is less secure [1], we assume encryption of Intra-coded blocks in Inter-frames as well as encryption of Intra-frames and compute E_{Sel_I+ib} as follows:

$$E_{Sel\_I+ib} = e_{DES} * (S_I + S_{ib}) + e_{overhead}$$

where $e_{overhead}$ is the amount of energy consumption to detect Intra-blocks after encoding. Since a typically encoded video consists of 30% to 60% Intra-frames of the total and S_{ib} takes up between 10% and 40% of S_P, we can select the median values to compare E_{Sel_I+ib} with E_{Naive}.

$$\begin{aligned} E_{Sel\_I+ib} &= e_{DES} * (0.45 + 0.25 * 0.55) * S_T + e_{overhead} \\ &= 0.59 * E_{Naive} + e_{overhead} \end{aligned}$$

The Zig-Zag Permutation Algorithm [4] shuffles coefficients at random without the use of cryptography. Thus there is no energy overhead for encryption and the analysis is shown below:

$$E_{Zig-Zag} = 0 * E_{Naive} + e_{overhead}$$

where $e_{overhead}$ is the amount of energy consumption for split and permutation. E_{encd}, however, increases since shuffling coefficients breaks the efficiency of encoding, which in turn increases the size of encoded video stream. Further the Zig-Zag algorithm violates security [2]. The key idea behind the Video Encryption Algorithm referred to as VEA is based on the statistical property of MPEG encoded video streams. Since the values of each byte in MPEG streams are uniformly distributed, we do not have to encrypt the entire video stream [2]. Instead, VEA attempts to encrypt half of the total stream after XORing two halves and E_{VEA} follows:

$$\begin{aligned} E_{VEA} &= e_{DES} * (0.5 * S_T) + e_{XOR} \\ &= 0.5 * E_{Naive} + e_{XOR} \end{aligned}$$

where e_{XOR} is the amount of energy consumption for XOR operations. While the uniform distribution characteristic may hold for video delivery applications with MPEG streams, we noticed that H.263 encoded streams for a video conferencing applications demonstrated uneven distributions, rendering VEA less useful for video streams encoded with H.263.

This paper extends the previous comparison [2] to include energy analysis on the video encryption algorithms as summarized in Table 1.

Table 1. Comparison of Video Encryption Algorithms

Algorithm	Security	Speed	Size	Relative Energy
Naive	High	Slow	No Change	100%*
Selective	Moderate	Fast	No Change	59%*
Zig-Zag	Very Low	Very Fast	Big Increase	<1%*
VEA	High	Fast	No Change	50%*

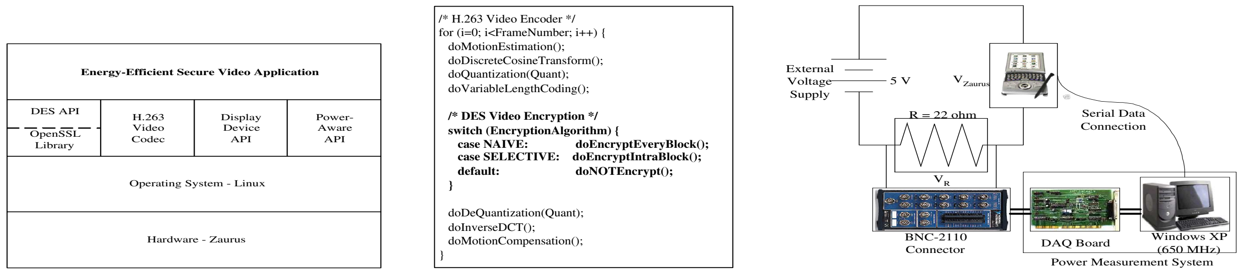
* Analytical Expectation

The Naive Algorithm is the most secure of the above approaches but is comparatively slower since it requires encryption of the whole data. The Selective Algorithm is faster than the Naive Algorithm since it encrypts the partial data instead of the total video clip. However the Selective Algorithm is not as secure as the Naive Algorithm. These two algorithms preserve the file size of the encoded video data after encryption. In terms of energy, the Selective Algorithm consumes about 59% of the Naive Algorithm excluding the overhead of detecting Intra-blocks. Our experiment will show a similar result that concurs with this analysis. The Zig-Zag Permutation Algorithm has a big drawback since it increases the file size (unsuitable for video communication); furthermore, the odd distribution in H.263 is a major obstacle for the use of VEA in mobile environments. Therefore in the rest of this paper, we focus our discussion and results on the Naive Algorithm and the Selective Algorithm for video encryption.

3. EVALUATION AND IMPLEMENTATION

3.1. Quality and Security Levels

In order to evaluate the tradeoffs between energy consumption and encryption strength as well as video quality, we define Quality and Security **(QnS)** levels classifying secure video application with respect to energy consumption. We suggest four classes of video quality according to quantization scale values such as 1, 4, 10 and 31, since the quantization scale dominates video quality in terms of PSNR (Peak Signal-to-Noise Ratio), and 10 is default, 1 is for the highest quality, 31 for the lowest quality, and the quality changes dramatically at 4. With respect to security,



```
/* H.263 Video Encoder */
for (i=0; i<FrameNumber; i++) {
  doMotionEstimation();
  doDiscreteCosineTransform();
  doQuantization(Quant);
  doVariableLengthCoding();

  /* DES Video Encryption */
  switch (EncryptionAlgorithm) {
    case NAIVE:         doEncryptEveryBlock();
    case SELECTIVE:     doEncryptIntraBlock();
    default:            doNOTEncrypt();
  }

  doDeQuantization(Quant);
  doInverseDCT();
  doMotionCompensation();
}
```

(a) System Architecture (b) Pseudo Code (c) Setup for Power-Profiling

Fig. 2. Power-Aware Secure Video Encoder

two classes of video encryption – high and low – are defined at each quality level. The high security encrypts all the encoded video file using symmetric cipher like DES but low security class encrypts only Intra-blocks in all encoded video blocks. Thus we define QnS levels ranging from level 1 to level 8 as shown in Table 2. For example, QnS level 1 represents the high video quality and high security so this level video is encoded where quantization scale is 1 and all the encoded bitstreams are encrypted, i.e., the Naive Algorithm. A cursory analysis shows that energy consumption for encoding with encryption or without encryption decreases with an increase of QnS level. Our detailed experimental results will confirm this observation.

3.2. Implementation

We use H.263 standard version [8] as a video codec generated by PeaCE [9] and DES library as an encryption algorithm from OpenSSL [10]. Thus we implement the secure video encoder and decoder for mobile handheld devices based on system architecture and algorithm as shown in Figure 2(a) and Figure 2(b), respectively.

Figure 2(c) shows our experimental setup. The testbed mobile handheld is a Zaurus (model:Sharp SL-5600) running Linux. It has a 400 MHz Intel XScale with 64 MB of flash memory and 32 MB of SDRAM. In order to obtain accurate power measurements, we remove the battery from the Zaurus during the experiment and place a resistor in series with the power supply. We use a National Instruments PCI DAQ (Data AcQuisition) board to sample the voltage drops across the resistor to calculate the current at 1000 samples per second. We calculate the instantaneous power consumption corresponding to each sample and profile each power consumption using the following formula:

$$P_{Zaurus} = \frac{V_R}{R} V_{Zaurus}$$

4. EXPERIMENTAL RESULTS

The first experiment evaluates energy consumption for each coder and crypto. We measure the AVERAGE power consumption and the Total Execution Time for each application and then compute the Expected Total Energy from multiplying them. In case of Figure 3(a) [1], the input file is FOREMAN.qcif with 300 frames and the H.263 encoder uses default parameters where quantization value is 10, IP ratio is 1:10, and the resolution is 176*144. Figure 3(a) examines the overheads of energy consumption for encryption compared with video encoding and decoding. For example, the Expected Total Energy for H.263 Encoder is around 75 Joules and 11 Joules for Decoder. However, DES consumes just 1.5 Joules for encrypting the whole encoded video or decrypting it, which is negligible since it is 2% of the energy used for encoding and 13% for decoding. On the other hand, H.263 decoder shows the biggest Energy Consumption per Byte from dividing the Expected Total Energy by the size of input file since the input file (233 KB) for DES and Decoder is much smaller than that (11 MB) for Encoder.

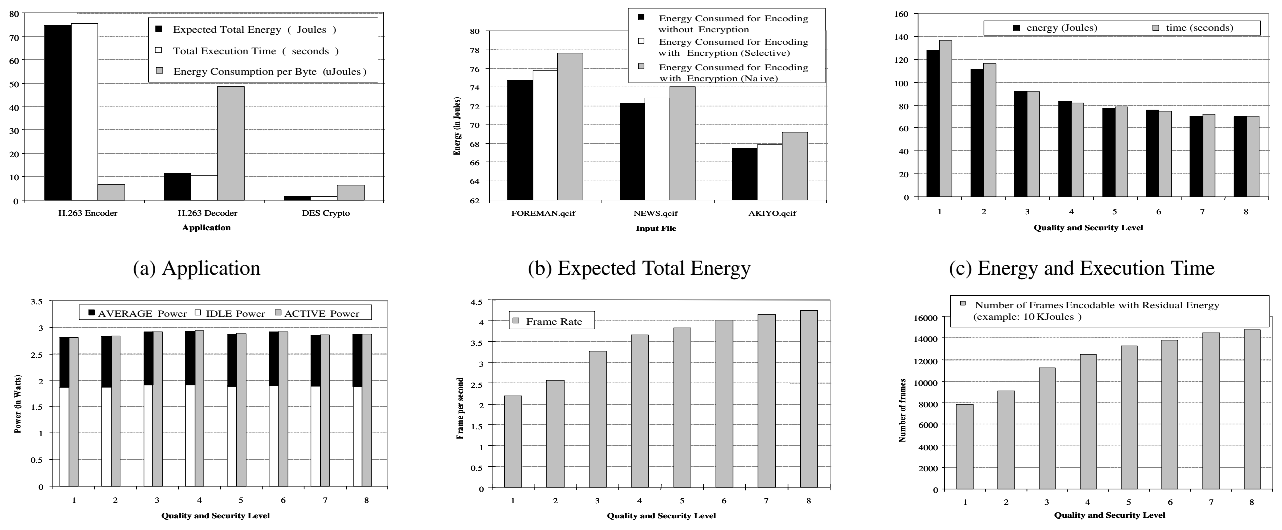
[1]This result can depend on encoding parameters and various input files.

The next experiment integrates the video encoder with the cipher. The results show that the Expected Total Energy for H.263 Encoding with the Naive Algorithm is about 78 Joules and for H.263 encoding without encryption is about 75 Joules, thus the energy overhead of encryption to encoder is only 3.6%. To confirm this result, we experiment on other video clips with smoother sequences and less activity like AKIYO.qcif and NEWS.qcif with the same number of frames as FOREMAN.qcif. Figure 3(b) displays that they all have the negligible energy overhead of full encryption on video encoding such as 2.5% at AKIYO.qcif and 2.4% at NEWS.qcif, respectively. And we compare the Naive Algorithm with the Selective Algorithm with respect to energy consumption. Apparently energy consumption for the Selective Algorithm is less than that for the Naive Algorithm but almost equals that for just encoding without encryption as shown in Figure 3(b). The energy overhead of the fully encrypted video encoder to the partially encrypted video encoder is at most 2.4% in case of FOREMAN.qcif.

Table 2. Tradeoffs between Energy Consumption and QnS levels

QnS	Quality	Quanti-zation	Security	Encryption	Energy (in Joules)
1	High	1	High	Naive	128.2
2	High	1	Low	Selective	111.0
3	Mid-High	4	High	Naive	92.05
4	Mid-High	4	Low	Selective	83.56
5	Mid-Low	10	High	Naive	77.62
6	Mid-Low	10	Low	Selective	75.78
7	Low	31	High	Naive	70.44
8	Low	31	Low	Selective	69.89

Finally, we profile the energy consumption in each secure video encoder with each QnS level. Table 2 exposes these experimental results that higher QnS in order from 8 to 1 requires higher energy consumption. For example, the encoder with QnS level 1 consumes 128 Joules but 111 Joules in QnS level 2 so the overhead is around 13%. How-



(a) Application (b) Expected Total Energy (c) Energy and Execution Time

(d) Power Consumption (e) Frame Rate (f) Number of Encodable Frames

Fig. 3. Experimental Results of Energy Consumption and QnS Levels

ever, they are inadequate in mobile environments due to the high requirement of the wide bandwidth. On the other hand, the AVERAGE power usage computed by subtracting IDLE power from ACTIVE power for each QnS level fluctuates around 0.99 Watts as presented in Figure 3(d), which validates that the execution time is a main cause of energy usage as Figure 3(c) depicts the energy consumption is linearly proportional to the execution time. According to QnS levels, we also measure performance metrics such as frame rate in fps (frame per second) and the number of frames to be encoded in the given residual energy - 10 KJoules. As shown in Figure 3(e), an encoder with the higher QnS level provides lower performance, i.e., lower frame rate. For instance, frame rate for QnS level 1 is about 2.2 fps but it increases to 4.2 fps in QnS level 8. The number of encodable frames in Figure 3(f) implies that the encoder with each QnS level can provide more or less encoded frames when switching to higher or lower QnS level. But the difference, for example, between the number of frames encodable with QnS level 1 and that with QnS level 2 within the given energy is 1,290 frames capable of providing the decoder with the video by just additional 43 seconds at 30 fps.

Through these experiments, we figure out that the video application with the high QnS demands the high energy consumption. However the energy usage for video encryption is definitely negligible compared to the video encoder.

5. CONCLUSION AND FUTURE WORK

This paper investigates the energy consumption characteristics for secure video applications on mobile handheld devices. To express the tradeoffs between quality and security, we classify video communications into various QnS levels with respect to two parameters - (a) quantization and (b) security level. Our basic experimental study arrives at the conclusion that while an encoding with encryption for better quality and higher security requires significant energy consumption, the energy overhead consumed for video encryption is negligible in comparison to that consumed for encoding. Future work will expand the notion of video quality to parameters other than quantization - these factors include video resolution, IP ratio, and other search algorithms for motion estimation. We also intend to explore the implications of lower layer networking components such as routing protocols, authentication methods, and key management algorithms for secure and power-aware mobile multimedia applications. Understanding the cross-layer interactions in mobile multimedia applications and exploiting these interactions to design efficient adaptation strategies that can balance the functional and non-functional needs of mobile applications is an important step in realizing pervasive multimedia services of the future.

6. REFERENCES

[1] L. Agi and L. Gong, "An Empirical Study of MPEG Video Transmissions," in *NDSS '96*, Nov 1996, pp. 137–144.

[2] L. Qiao and K. Nahrstedt, "Comparison of MPEG Encryption Algorithms," in *International Journal of Computers and Graphics, special issue: Data Security in Image Communication and Network*, Jan 1998, vol. 22, pp. 437–448.

[3] C. Shi and B. Bhargava, "An Efficient MPEG Video Encryption Algorithm," in *SRDS '98*, Oct 1998, pp. 381–386.

[4] L. Tang, "Method for Encrypting and Decrypting MPEG Video Data Efficiently," in *ACM Multimedia '96*, Nov 1996, pp. 219–230.

[5] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*, Prentice Hall, 1995, ISBN 0-13-324435-0.

[6] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, Oct 1996, ISBN 0-8493-8523-7.

[7] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the Energy Consumption of Security Protocols," in *ISLPED '03*, 2003.

[8] ITU-T, "H.263 Draft: Video Coding for Low Bitrate Communication," Draft ITU-T Recommendation H.263, ITU-T, May 1996.

[9] Seoul National University, "PeaCE: Ptolemy extension as Codesign Environment," Tech. Rep., SNU, Nov. 2003.

[10] The OpenSSL Project, "OpenSSL," in `http://www.openssl.org/`.