Cost-Effective Security Support in Real-Time Video Surveillance

Udaya L. N. Puvvadi, Kevin Di Benedetto, Aditya Patil, Kyoung-Don Kang *Member, IEEE*, Youngjoon Park

Abstract—Visual surveillance has numerous applications. It can support, for example, public safety, traffic monitoring, and facility protection to name just a few. Networked digital surveillance devices are revolutionizing the surveillance industry by supporting high quality images, remote monitoring, and advanced image processing. However, they also raise serious privacy/security concerns. To address the issue, the surveillance industry has recently begun to provide basic support for secure communications. In this paper, we present a new protocol to significantly enhance the security and performance compared to the state-of-the-art baseline method widely taken in the video surveillance industry. Through extensive experiments for performance evaluation, our approach is shown to substantially reduce the delay to execute cryptographic mechanisms and increase the supported bit rate compared to the baseline, while providing desirable security features.

Index Terms—Digital Video Surveillance, Data Confidentiality, Integrity, and Freshness, Delay for Cryptographic Processing, Bit Rate

1 Introduction

Visual surveillance systems are increasingly deployed in many places, such as buildings, streets, industrial facilities, schools, shopping centers, airports, and homes, to support, for example, public safety, traffic monitoring, and infrastructure protection. The recent availability of digital HD (High Definition) cameras and network surveillance devices that support the IP (Internet Protocol) greatly enhances the effectiveness of visual surveillance by supporting high quality images, remote monitoring, and advanced image processing. For example, it is envisioned that a suspicious behavior in a crowd can be detected in real-time by analyzing HD images using advanced computer vision techniques before it becomes a problem [1]. In 2012, approximately 8 million security/surveillance devices worldwide were connected to the Internet and the number is expected to grow to 170 million in 2021 [2]. This is a sea change in the surveillance industry that traditionally relied on analog cameras with limited processing capabilities and closed, isolated surveillance networks.

On the other hand, networked visual surveillance systems raise privacy concerns. As more IP cameras and other surveillance/recording devices are deployed and networked, the fear of always being watched and recorded is increasing. An attacker may eavesdrop plaintext images transmitted across computer networks. One can also see or modify stored images, if they already have or get legitimate/illegitimate access to the networked storage device that is required to retain the surveillance images for 30 - 180 days depending on the security requirement of the specific organization being monitored [3]. A fundamental approach to alleviate privacy concerns is encrypting surveillance images using cryptographic techniques. However, the overall performance of real-time visual surveillance can be degraded, because cryptographic techniques are computationally heavy. As a result, an important event, e.g., a crime scene, could be detected late. Although lightweight techniques, e.g., weaker encryption algorithms, selective encryption of image frames, and image scrambling, may decrease the computational resource consumption, they are more vulnerable to certain attacks, e.g., statistical analysis to extract the original unencrypted images [4]–[6].[1]

In addition to passive eavesdropping, it is relatively easy to launch active attacks against video surveillance. Even simple active attacks, which replay previously captured images, could be devastating, because visual surveillance data often have substantial redundancy. For example, the same background image can be captured and transmitted by the cameras for most of the time in a high security area usually empty due to the strong physical security enforcement. Therefore, an attacker can launch a replay attack with relative ease by intercepting a few of those images transmitted across the network, even if he does not have the

• *U. L. N. Puvvadi, K. Di Benedetto, A. Patil, and K. D. Kang are with the Department of Computer Science, State University of New York at Binghamton, U.S.A. Y. Park is with Hanwha Techwin Corp., Korea. (K. D. Kang is the corresponding author.) E-mail: {upuvvad1,kdibene1,apatil10,kang}@binghamton.edu and yj71.park@hanwha.com*

1. Our approach is not tied to any specific encryption algorithm, but generally applicable to support the data confidentiality, integrity, and freshness for video surveillance in a cost-effective manner.

cryptographic key needed to encrypt/decrypt images. He/she can launch a replay attack that will continue for a certain time interval before attempting to enter a high security area under surveillance without detection.

In a video surveillance system, digital IP cameras typically transmit images to an NVR (Network Video Recorder), which can store the transmitted images, forward them to the CMS (Central Management System), and optionally display them (as a video). The CMS stores and displays the images forwarded from the NVR. Recently, visual surveillance systems have begun to support basic confidentiality by instrumenting each digital IP camera to encrypt a surveillance image before transmitting it to an NVR. Most of them support a tunneling protocol using the OpenSSL library [7], similar to SSL/TLS (Secure Socket Layer/Transport Layer Security). In the tunneling protocol, the receiver, e.g., an NVR or CMS, immediately decrypts the received packet. In general, this is acceptable in SSL/TLS underlying HTTPS for secure web transactions. However, it is agnostic to the needs of visual surveillance and subject to serious performance and security disadvantages. First, an NVR has to re-encrypt the decrypted image data to securely store them locally or forward them to the CMS. The decryption and re-encryption of the surveillance images originally encrypted and transmitted by the IP cameras incur considerable computational overheads. At the same time, surveillance images decrypted upon arrival can be inadvertently exposed to attackers or unprivileged users, compromising the confidentiality. Furthermore, relatively little attention has been paid to avoid active attacks, e.g., packet replay, modification, or injection attacks, and to support key management critical for supporting cryptographic security.

To address these problems, we design, implement, and evaluate a new protocol, called SVS (Secure Video Surveillance). A key observation for SVS's design is that it is unnecessary for an NVR to decrypt and then re-encrypt an image received from a digital camera to store locally or forward it to the CMS securely in terms of confidentiality. Instead, it is better for the NVR to simply store the received encrypted image locally or forward it to the CMS *as is*, without decrypting it and then re-encrypting it, in terms of both security and performance. Similarly, in the CMS, the encrypted image received from the NVR is stored withoug being decrypted and re-encrypted. It is only decrypted, if necessary, to be played back in the surveillance display in a physically secured area, such as the security control center in a modern building. (After being displayed, the decrypted images are discarded.) Compared to the tunneling protocols based on OpenSSL, SVS consumes much less computational resources by avoiding unnecessary decryption and re-encryption to securely store or forward real-time surveillance images, while enhancing the data confidentiality by avoiding any inadverdent exposure of surveillance images to attackers or unprivileged users. Therefore, it is more *cost-effective* than the state-of-the-art tunneling approaches for secure real-time video surveillance. Moreover, it supports the other important security features not supported by the tunneling schemes that only consider data confidentiality as follows:

- For each packet, the data integrity as well as the authenticity of the source and destination addresses are supported, via message authentication based on secure one-way hashing, to detect packet modification or injection attacks.
- The verification of each packet's freshness is supported to avoid replay attacks.
- A secure key management scheme is supported.

Notably, in SVS, these desirable features are supported not only for real-time surveillance data transmitted from a camera to another device, e.g., an NVR or CMS, but also for stored and forwarded data. To support these desirable security features in a cost-effective manner, we apply cryptographic encryption and one-way hashing techniques using symmetric keys that are several orders of magnitude faster than alternative methods based on a public key system, e.g., RSA, are [8] as much as possible. Moreover, in SVS, no additional processing of the surveillance data already stored in an NVR or CMS in a secure manner, via the aforementioned techniques, is needed due to a cryptographic key renewal required for enhanced security.

For performance evaluation, we have implemented our protocol and the baseline tunneling approach that represents the current state-of-the-art in the video surveillance industry. In our performance evaluation using a Samsung IP camera (SND-6084R) and two workstations modeling an NVR and CMS, SVS decreases the total delay to process the cryptographic mechanisms by more than 80%, while providing enhanced data confidentiality, integrity, and freshness support. Furthermore, compared to the baseline, our approach decreases the corresponding delay and increases the throughput (bit rate) for an increasing number of video streams, which model real-time surveillance image streams in our computer science department network, by $28\% - 47\%$ and $39.08\% - 89.01\%$, respectively. In addition to the fact that SVS enhances security via the protocol design, these experimental results empirically verify that SVS is considerably more cost-effective compared to tunneling-based protocols.

The rest of the paper is organized as follows. Related work is discussed in Section 2. In Section 3, an overview of the proposed system architecture is given. A formal description of the SVS protocol is given in Section 4. The performance of SVS is compared to the baseline tunneling protocol in Section 5. Finally, Section 6 concludes the paper and discusses future work.

2 Related Work

Smart, automated, and intelligent surveillance systems [9]–[11] are developed to support more intelligent surveillance capabilities, e.g., moving object detection, object classification [12], human motion analysis, activity interpretation [13], traffic control [14], health care, and military data collection, requiring less human involvement. However, the security/privacy concerns are increasingly raised as more surveillance systems are deployed and smarter functionalities are supported. Further, video surveillance is subject to real-time performance constraints [11], [15]. To support secure surveillance efficiently, our protocol, SVS, improves upon the baseline approach to secure communication that is a de facto standard in the surveillance industry, in terms of the time taken to process cryptographic mechanisms by utilizing them only when needed, simultaneously enhancing the support for data confidentiality, (source/destination) authenticity, integrity, and freshness. By doing this, SVS alleviates security/privacy concerns in a cost-effective manner.

As more digital IP cameras are deployed, fundamental security support, such as encryption, is adopted in surveillance networks. However, other crucial security requirements are often overlooked. For example, most OpenSSL-based approaches commonly used in state-of-the-art surveillance systems do not consider data integrity and freshness issues. Also, they are subject to redundant decryption and re-encryption, incurring performance penalties. As another example, Wang et. al. [16] utilize the Diffie-Hellman key exchange algorithm that suffers from man-in-the-middle attacks, which may largely compromise the confidentiality and integrity. SVS circumvents this issue by supporting secure key exchanges based on RSA that is free of man-in-the-middle attacks [8], while decreasing the frequency of shared key renewals and supporting the data integrity and freshness.

It is known that a field-programmable gate array (FPGA) [17] can be utilized to accelerate encryption. In this paper, we focus on enhancing the security and performance of real-time visual surveillance by designing a new protocol, without using any additional special-purpose hardware accelerator that may increase the monetary cost of a surveillance system. In the future, SVS can be combined with an FPGA to further improve performance. Thus, it is complimentary to [17].

3 An Overview of SVS

In this paper, we assume that appropriate user authentication and physical security are enforced. Also, we assume that guaranteed and ordered packet deliveries between two communicating parties are supported by the underlying network stack. Under the assumptions, an overview of SVS is given in this section.

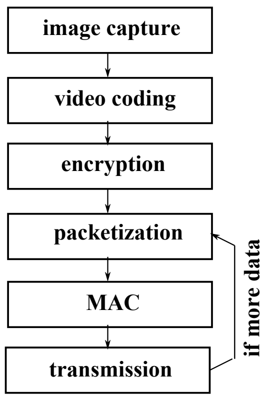

Fig. 1. Control Flow in a Network Camera

3.1 Security Support in a Data Source

In SVS, a digital IP camera is a source or producer of surveillance image data. It periodically performs the following steps illustrated in Figure 1 to support secure real-time surveillance:

1) Periodically capture an image where the period is the inverse of the specified frame rate, e.g., 60 fps (frames per second).
2) Compress the captured image using a codec, such as H.264.
3) Encrypt the compressed image to support data confidentiality using a well established encryption algorithm, e.g., the AES (Advanced Encryption Standard) algorithm.
4) Packetize the encrypted data and compute the MAC (Message Authentication Code) upon the source/destination addresses, payload, and counter incremented for each successful packet transmission. Append the computed MAC to the packet.
5) Transmit the packet to the destination, e.g., an NVR or CMS.
6) If the image needs to be delivered using multiple packets, repeat Steps 4 and 5 until the entire image is transmitted to the destination.

3.2 Security Support in a Destination

In a visual surveillance system, monitoring images are usually transmitted to an NVR or CMS that records, analyzes, and displays them. Received images can also be forwarded to the other specified surveillance devices. Therefore, it is important to support security in not only the source but also the destination.

In Figure 2, the control flow diagram for security support in the destination, e.g., an NVR or CMS, is

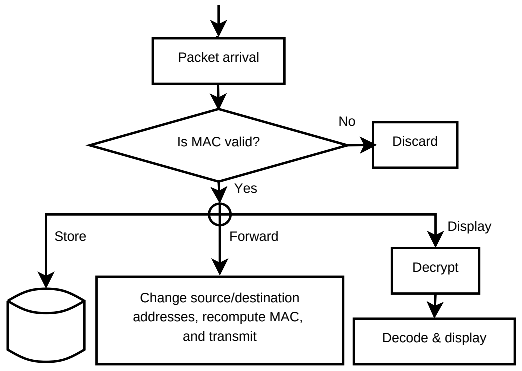

Fig. 2. Control Flow in an NVR or CMS

depicted.[2] As shown in Figure 2, the destination, e.g., an NVR or CMS, in SVS verifies the MAC of each received packet. If the verification is unsuccessful, the packet is discarded. Conversely, if it is successful, 1) the authenticity of the source and destination are verified; that is, the source actually sent this packet to the destination, 2) the packet is not modified in transit, and 3) the packet is fresh (not replayed), because it is computationally infeasible for an attacker, which does not have the cryptographic key used to compute the MAC, to modify the message or replace it with a different message without detection [8]. Subsequently, the destination node performs one of the following operations:

- If the required operation is 'store', simply store the encrypted payload together with the source/destination addresses, counter, and MAC to support the confidentiality, authenticity, and integrity of the stored data. Thus, unlike tunneling protocols broadly adopted in the video surveillance industry, SVS does not decrypt and re-encrypt the received data to store them, avoiding potential data leaks and computational overheads as discussed before.
- If the operation is 'forward', create a new message that consists of the new source and destination addresses, the original encrypted data received from the source IP camera, and the newly computed MAC. Transmit the constructed message to the destination, e.g., a CMS. Note that, unlike tunneling protocols, SVS only needs to change the source and destination addresses and recompute the MAC without doing any decryption and re-encryption of the image data for message forwarding. Although recomputing the MAC is clearly not free in terms of computation, it is unavoidable to verify the entire message's integrity and the authenticity of the source and destination addresses. Hence, we claim the overhead of SVS is minimal and much smaller than that of a tunneling alternative that requires to decrypt and re-encrypt all received messages regardless of application needs.
- If the operation is 'display', decrypt and display the image. In SVS, this is the only case where the received encrypted data should be decrypted.

2. A single diagram is used to show the control flow in both the NVR and CMS, as their functionalities are similar.

In SVS, data are decrypted only if necessary and, therefore, unnecessary decryption and re-encryption for secure storage and forwarding of surveillance images are eliminated. This is different from the state-of-the-art tunneling protocol for secure visual surveillance that first decrypts all incoming images and re-encrypts them for secure storage or forwarding, incurring considerable overheads and potential data leak as discussed before.

In a video surveillance system, users may want to display or forward stored data later. In such a case, SVS first verifies the authenticity and integrity based on the stored source/destination addresses, encrypted data, and MAC. After a successful verification, the retrieved data is securely forwarded to another device or decrypted and displayed in SVS. Thus, in the rest of this paper, we focus on secure transmission and storage of real-time surveillance data.

4 SVS: Secure Video Surveillance Protocol

In this section, a more formal description of SVS is given. The symbols used in this section are summarized in Table 1.

Symbol	Meaning
A, B, C	Data producer, intermediary, and consumer
$K_{U,A}$, $K_{U,B}$, $K_{U,C}$	Public keys of A, B, and C
$K_{A,B}$	Symmetric encryption key shared between A and B
$K'_{A,B}$	Message authentication key shared between A and B
$\mathcal{E}(K_{A,B}, data)$	Data encrypted using $K_{A,B}$
$MAC(K'_{A,B}, data)$	Secure 1-way hash of the data computed using $K'_{A,B}$

TABLE 1
Symbols used to describe SVS

4.1 Key Management

4.1.1 *Verification of Public Keys*

In SVS, a public key system (PKS), e.g., RSA [8], is applied to initially establish a secure connection between two networked surveillance devices. Each device has a pair of a private key and a public key digitally signed by a trusted third party. To initiate secure communications, an arbitrary pair of a producer and a consumer, A and B, check each other's certificate to verify the validity of the opponent's public key:

$$A \text{ verifies } K_{U,B} \text{ and } B \text{ verifies } K_{U,A} \tag{1}$$

where $K_{U,A}$ and $K_{U,B}$ are the public keys of A and B, respectively. If the mutual verification fails in A or B, all the other steps are aborted.

4.1.2 Shared Symmetric Key Exchanges

If the verification in Step 1 is successful, A generates a pair of random symmetric keys to encrypt and authenticate each message for video surveillance, $\{K_{AB}, K'_{AB}\}$, encrypts them using B's public key $K_{U,B}$, and transmits them to B:

$$A \rightarrow B : \mathcal{E}(K_{U,B}, K_{AB}||K'_{AB}) \tag{2}$$

where $||$ is a concatenation.[3] Note that only B can decrypt this message using its private key to extract shared symmetric keys, $K_{A,B}$ and K'_{AB}, used to encrypt and authenticate messages between A and B. In SVS, symmetric keys are used to actually encrypt and authenticate messages, because a cryptographic algorithm using the shared key, e.g., AES, is orders of magnitude faster than an asymmetric algorithm using a pair of a public and private key, e.g., RSA [8]. Thus, a shared symmetric key system is more efficient to encrypt and authenticate real-time surveillance data.

4.1.3 Key Management for Stored Data

Usually, surveillance data have to be stored for an extended period of time [3]. As the encrypted data and MAC are stored together with the data, the secret encryption and MAC keys, e.g., K_{AB} and K'_{AB}, should be securely stored to decrypt and verify the integrity of the stored data in B. In SVS, K_{AB} and K'_{AB} are encrypted using B's public key. When the stored data need to be decrypted and authenticated, the encrypted K_{AB} and K'_{AB} are decrypted using B's private key. Thus, in SVS, the confidentiality and integrity of the transmitted and stored data are supported as long as B's private key is kept securely. For example, a private key can be encrypted by a master key and saved in a separate read-only storage medium such as CD-ROM. In SVS, a similar approach is taken when data is securely forwarded from B to C that may store, forward, or (decrypt and) display the data.

4.1.4 A Renewal of a Public-Private Key Pair

In a PKS, a certificate digitally signed by a trusted third party includes the expiration date [18]. A public-private key pair has to be renewed only when the expiration period is over, unless there is a relatively rare security incident involving the certificate authority before the certificate expiration. A device performs the following tasks to efficiently maintain the confidentiality and integrity of the stored surveillance data even after a renewal of the public-private key pair:

1) Using the previous private key, decrypt the symmetric keys, e.g., K_{AB} and K'_{AB}, shared between A and B to encrypt and authenticate surveillance images.
2) Re-encrypt the symmetric encryption and MAC keys using the new public key.
3) Declare the expiration of the previous public key to the other nodes on the surveillance network.
4) Destroy the old public-private key pair.
5) Distribute the new public key to the nodes in the surveillance network.

Notably, this approach requires the minimal computation to only decrypt and re-encrypt the symmetric keys used for encryption and message authentication. It requires no further processing of the already stored surveillance data, such as decryption and re-encryption of the data stored in an NVR or CMS, when the public-private key pair is renewed. Thus, the renewal procedure of SVS is cost-effective in terms of security and computational efficiency. Moreover, it is executed only when a public-private key pair is actually renewed, which is relatively infrequent (e.g., once a year).

4.2 Secure Data Transmission and Storage

Generally, in a video surveillance network, an IP camera is a data producer and an NVR or CMS is a consumer. Furthermore, a consumer can work as an intermediary that securely forwards the surveillance images to another consumer. Thus, a node can be a consumer and a producer at the same time, if it forwards surveillance images received from a producer to another consumer as an intermediary. In this section, we first describe how a pair of a producer and consumer in SVS communicate and later discuss how a consumer works as an intermediary to securely relay data between a producer and consumer that is not directly communicating with the producer.

4.2.1 Data Confidentiality, Authenticity, Integrity, and Freshness Support

In SVS, a producer A periodically sends a message to a consumer/intermediary B that consists of the encrypted data followed by the corresponding MAC:

$$A \rightarrow B : \mathcal{E}(K_{AB}, D)||MAC(K'_{AB}, A, B, \mathcal{E}(K_{AB}, D), ctr_A) \tag{3}$$

where data D is encrypted using the shared symmetric key K_{AB} to support the data **confidentiality**.

In SVS, to support the data **authenticity**, **integrity**, and **freshness**, the MAC is computed using the symmetric key, K'_{AB}, and a cryptographic one-way hash function, e.g., MD5, SHA-1, or SHA-2 [8], based on the producer's and consumer's IP addresses (A and

3. Alternatively, B can generate a pair of symmetric keys and share them with A.

B), encrypted image data, and counter value ctr_A incremented by A after successfully sending a message to B as described in Step 3 above. When B receives a packet, it recomputes the MAC based on the received packet's source/destination addresses, encrypted data, and counter:

$$B \text{ recomputes } MAC'(K'_{AB}, A, B, \mathcal{E}(K_{AB}, D), ctr_A)) \tag{4}$$

B compares the MAC received from A (Step 3) and MAC' (Step 4). If $MAC = MAC'$, the message is not modified in transit and it is actually sent by A to B. As the message authenticity and integrity are confirmed formally, B sends an ACK (acknowledgment) to A. If A receives an ACK from B, it increments its counter, ctr_A, which is initially 0. Otherwise, it resends B the packet formed in Step 3. (If the transmission fails for more than a prespecified re-transmission threshold, the packet is dropped.)

In this paper, we mainly consider the TCP/IP as the underlying communication protocol to reliably transmit surveillance images. However, it is possible to extend SVS to support the UDP. To do this, after A transmits a packet, it can simply increment the counter without waiting for an ACK. However, the UDP is a connectionless protocol; therefore it may result in many packet losses, reducing the quality of visual surveillance. A thorough investigation of the trade-off between the less bandwidth usage and quality degradation that could be provided by the UDP is reserved for future work.

Data freshness is supported by including the counter to compute and verify the MAC in Steps 3 and 4. To support real-time video surveillance, A generates a message as described in Step 3 and transmits it to B at every transmission period $P_A = 1/r_A$ where r_A is the prespecified frame rate supported by A. For example, suppose that an IP camera A sends an empty background image to B at time t and transmits the same image to B taken at time $t + P_A$. Even in such a case, the MAC computed in Step 3 at time $t + P_A$ will be different from the previous one computed at time t, since the counter value is incremented for the second message at $t + P_A$. Without the key, K'_{AB}, used to compute the MAC, it is infeasible for an attacker to compute the correct MAC, even if he has the same data, the exact counter value, and the secure 1-way hash function, which may be known to the public. Thus, an adversary cannot launch a replay attack without detection.

When the counter rolls over, a new pair of symmetric encryption and MAC keys needs to be generated and exchanged between the producer and consumer to avoid cryptanalysis and packet replay/injection attacks. In both RTP (Real-Time Transport Protocol) and SRTP (Secure Real-Time Transport Protocol) used for video streaming, a 16 bit sequence number is used [19]. However, this is not sufficient to avoid replay attacks in video surveillance. If one image (e.g., an H.264 frame) fits into a single packet, the 16 bit counter rolls over approximately every 36 minutes when images are streamed at 30 fps. An eavesdropping adversary can simply record, for example, the initial 10 minute-worth packets and begin to replay them when the 16 bit counter rolls over. To launch such an attack, an adversary does not have to know any secret, such as the cryptographic keys for encryption and message authentication. One way of avoiding it is renewing the key used to compute the MAC; however, it requires frequent key renewals and exchanges between the communicating parties. To address this issue, in SVS, a 48 bit counter is used to compute the MAC in Step 3. The key renewal period for the 48 bit counter is longer than the lifetime of surveillance devices by orders of magnitude. Thus, K_{AB} and K'_{AB} can be renewed only when there is a security incident/concern or periodically, e.g., every six month, as a precaution. When K_{AB} and K'_{AB} need to be renewed in SVS, A and B verify each other's public key, A derives new symmetric keys for encryption and message authentication, and A shares the new keys with B as described in Subsections 4.1.1 and 4.1.2.

4.2.2 Securely Storing and Forwarding Received Data

After a successful verification of the MAC, B (e.g., an NVR) stores, forwards, or displays the received data. Depending on the specific surveillance requirements, B needs to perform all or a subset of the three operations concurrently. In SVS, B **stores** the encrypted data and MAC received from source A as follows:

$$B \text{ stores } \mathcal{E}(K_{AB}, D)||MAC(K'_{AB}, A, B, \mathcal{E}(K_{AB}, D), ctr_A) \tag{5}$$

When it retrieves the stored data later, B recomputes the MAC, similar to Step 4, and verifies whether the stored MAC matches the recomputed MAC or not. If the verification succeeds, it returns the retrieved data. Otherwise, it drops it. In this way, the confidentiality and integrity of the stored data are supported in SVS too.

If B works as an intermediary between the producer A and another consumer C that is not directly connected to A, B **forwards** the data received from A to C. To do this, B and C need to share K_{AB} and K'_{AB}. In SVS, this approach is taken rather than using a different pair of keys for encryption and message authentication between B and C, because it requires B to decrypt and re-encrypt every message, further complicating key management.

Initially, B and C need to verify their public keys with each other. If the verification is successful, B encrypts K_{AB} and K'_{AB} using C's public key, $K_{U,C}$, and transmits the encrypted keys to C as follows.

$$B \rightarrow C : \mathcal{E}(K_{U,C}, K_{AB}||K'_{AB}) \tag{6}$$

Only C can decrypt this message to extract K_{AB} and K'_{AB} using its private key.

After sharing the encryption and MAC keys with C, B periodically forwards the surveillance data it receives from A to C:

$$B \rightarrow C : \mathcal{E}(K_{AB}, D)||MAC(K'_{AB}, B, C, \mathcal{E}(K_{AB}, D), ctr_A) \quad (7)$$

Note that, in Step 7, B only has to recompute the MAC using the new source and destination addresses, i.e., B and C, whereas it simply forwards the encrypted data $\mathcal{E}(K_{AB}, D)$ without first decrypting and subsequently re-encrypting it. When C receives the message, C verifies the MAC, similar to Step 4:

$$C \text{ recomputes } MAC'(K'_{AB}, B, C, \mathcal{E}(K_{AB}, D), ctr_A) \quad (8)$$

If $MAC = MAC'$, C stores the encrypted data and MAC received in Step 7, forwards them to another node using the technique similar to Steps 6 and 7, and/or displays the received data according to the requirements of the specific surveillance application. Thus, in SVS, the confidentiality, integrity, and freshness are supported for forwarded data too.

5 Performance Evaluation

For performance evaluation purposes, we have compared SVS to OpenSSL, which is the baseline tunneling protocol commonly used in the surveillance industry. The OpenSSL library [7] and TCP/IP protocol were use to ensure the reliable, ordered delivery, of surveillance data. The baseline always decrypts and re-encrypts a received message, to store it locally or forward it to another networked surveillance device as discussed before. On the other hand, both the baseline and SVS support the fundamental integrity and data freshness features described in Section 4 to avoid data modification, false data injection, or replay attacks.

We have done extensive performance evaluation consisting of two sets of experiments. In the first set, we have used a Samsung SND-6084R digital IP camera with the Cortex A8 800 MHz CPU and 256 MB DDR3 memory. A single port is used to supply power to the camera and wire it to the Ethernet based on the PoE (Power over the Ethernet) technology. Also, two workstations are used to model an NVR and a CMS, respectively. (For brevity, they are referred to as the NVR and CMS in the rest of this paper.) Each workstation has an Intel Core i7-4790 quad core CPU running at 3.6 GHz, 16 GB memory, and a 1 TB hard disk. Also, every machine runs Ubuntu 14.04 LTS.

In the second set, we use three workstations. One of them is used to generate an increasing number of simulated surveillance image streams for performance evaluation, while the other two are used as the NVR and CMS. All four cores and eight hardware threads of the CPU are used in each machine. In all our experiments, the 1 Gbps Ethernet available in the Department of Computer Science at the Binghamton University is used to transmit data.

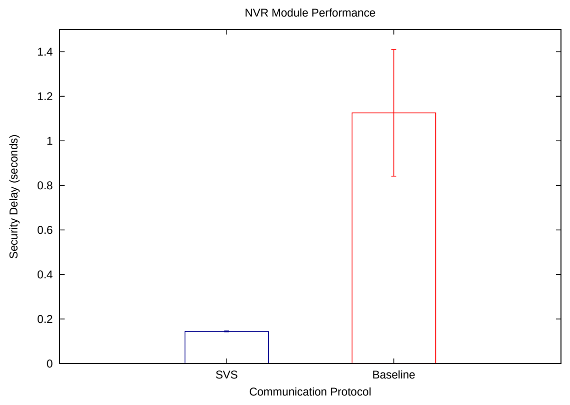

Fig. 3. Security Delay in the NVR

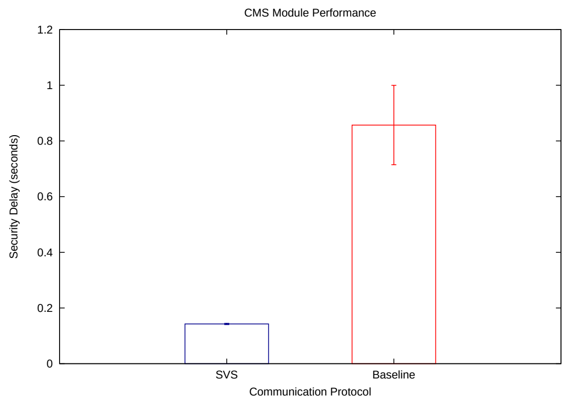

Fig. 4. Security Delay in the CMS

5.1 Experimental Set 1

In this set of the experiments, 1,000 images are transmitted at 30 fps from the Samsung IP camera to the NVR. It locally stores and forwards the received encrypted images to another workstation used to model a CMS that locally stores encrypted images. In addition, the received images are decrypted and displayed in the CMS. This experiment is run 10 times to report the average performance with 95% confidence intervals.

In order to illustrate the speedup achieved with SVS when compared to the state-of-the-art baseline, we define a term, *security delay*, which refers to the total delay to process the cryptographic mechanisms. In the NVR, as shown in Figure 3, the baseline and SVS yield the average security delay of $1.125 \pm 0.284s$ and $0.144 \pm 0.002s$, respectively. Thus, SVS decreases the average security delay by approximately 87%. Also, the confidence interval of SVS is significantly smaller than that of the baseline, indicating that SVS has much smaller security delay variations.

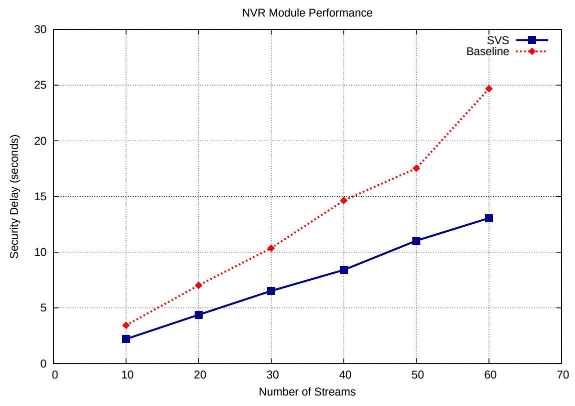

Fig. 5. Security Delay in the NVR

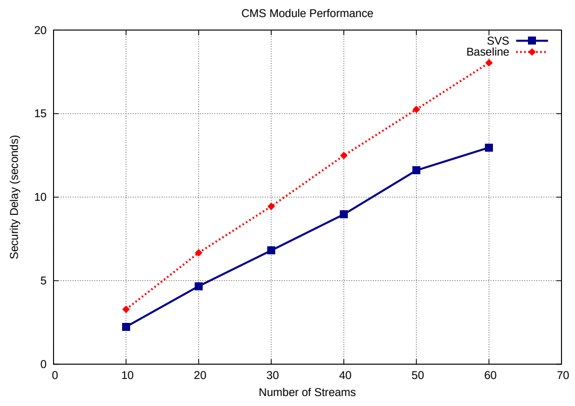

Fig. 6. Security Delay in the CMS

In the CMS, as plotted in Figure 4, the baseline and SVS show the security delay of $0.857 \pm 0.142s$ and $0.143 \pm 0.002s$, respectively. Thus, SVS decreases the security delay by approximately 83% in the CMS. In this experiment, the average security delay of the baseline is slightly decreased, because the CMS does not have to forward the received data. Therefore, the baseline has no need to decrypt and re-encrypt the received encrypted data for forwarding. The average security delay of SVS is similar to the value observed in the NVR, because unnecessary decryption and re-encryption for data storage and forwarding subject to the performance penalty and confidentiality vulnerability is eliminated in the SVS design phase. In summary, by forgoing unnecessary decryption and re-encryption, SVS decreases the security delay by 87% in the NVR, and by 83% in the CMS, when compared with the baseline approach.

5.2 Experimental Set 2

In this set of experiments, one workstation is used to generate $10 - 60$ image streams, increased by 10, to evaluate the performance of the baseline and SVS for increasingly intense workloads. We take this approach to generate realistic workloads, since only one IP camera is provided to us by Samsung. Neither are we allowed access to any actual visual surveillance network for security/privacy reasons. More specifically, we consider a scenario of one terminal of a highly secured airport equipped with up to 60 cameras for surveillance. To implement this scenario, we mimic the behavior of one camera by one thread; each thread transmits a stream of 1,000 images extracted from a video of a flight of an airplane with a considerable amount of motions [20] to model a relatively high, real-world workload to stress the system at 60 fps, which is the maximum frame rate supported by most of the security cameras on the market. For 60 video streams, with each transmitting 60 of 4 KB frames per second, the total bitrate is 115.2 Mbps. The maximum bitrate supported by our experimental machines is 125 Mbps. Beyond 60 streams, the entire system stops working, since the upper bound of the Linux buffer space is exceeded given a large amount of video data streamed in real-time.[4]

In the NVR, as shown in Figure 5, the baseline and SVS support the security delays of $3.433s$ and $2.199s$ for 10 streams, respectively. For 60 streams, the baseline and SVS provide the security delay of $24.683s$ and $13.059s$, respectively. As depicted in Figure 5, the difference between the baseline and SVS increases for the increasing number of the streams. Thus, SVS is more scalable than the baseline is as the number of the streams increases. SVS decreases the security delay by $35\% - 47\%$ as shown in Figure 5.

In the CMS, as shown in Figure 6, the baseline and SVS support the security delays of $3.284s$ and $2.24s$ for 10 streams, respectively. For 60 streams, the baseline and SVS provide the security delays of $18.041s$ and $12.971s$, respectively. Thus, SVS decreases the security delay by $28.1\% - 31.7\%$. As shown in the figure, the difference between the baseline and SVS increases for the increasing number of the streams, similar to the case of the NVR. Hence, we observe that SVS is more scalable. In the CMS, the speedup of SVS against the baseline is smaller than it is in the NVR, similar to the results in Section 5.1. This is because the CMS is not required to forward the received data to any other surveillance device.

Figure 7 plots the throughput (bit rate) of SVS normalized to that of the baseline in the NVR. As shown in the figure, SVS increases the throughput by $56.08\% - 89.01\%$ as the number of streams is increased from 10 to 60. Specifically, SVS supports 46.8 Mbps and 47.3 Mbps in the NVR for 10 and 60 streams, respectively. Under the baseline approach, 29.9 Mbps and 25 Mbps are supported in the NVR for 10 and 60 streams, respectively.

4. More data can be transmitted in real-time, if the buffering limitations of the systems used for our experiments can be mitigated. In such settings, the magnitude of SVS's performance improvement in comparisons to the tunneling baseline might increase, since SVS can eliminate unnecessary encryption and re-encryption for more data. A thorough investigation is reserved for future work.

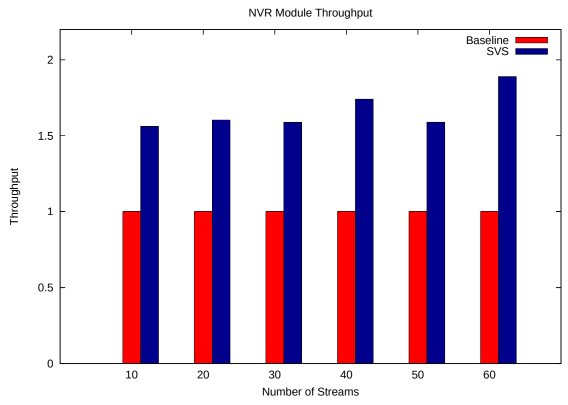

Fig. 7. Normalized Throughput in the NVR

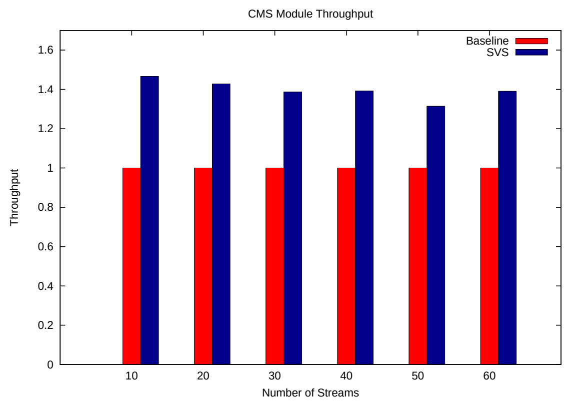

Fig. 8. Normalized Throughput in the CMS

In the CMS, as shown in Figure 8, the throughput of SVS is higher than that of the baseline by $39.08\% - 46.59\%$. More specifically, SVS supports 45.9 Mbps and 47.6 Mbps for 10 and 60 streams, respectively. The baseline supports only 31.3 Mbps and 34.2 Mbps for 10 and 60 streams, respectively.

Overall, our experimental results demonstrate that our protocol reduces the security delay by $28\% - 87\%$ (depending on the system settings and workloads) and increases the throughput by $39.08\% - 89.01\%$ compared to the baseline. At the same time, data confidentiality is enhanced by avoiding unnecessary decryption and re-encryption subject to unintended data leaks and computational overheads. Further, data integrity, data freshness, and source/destination authenticity are supported to avoid packet modification, replay, and injection attacks that could lead to devastating results in video surveillance. Thus, it is significantly more cost-effective than the state-of-the-art baseline relying on tunneling.

6 Conclusions and Future Work

Networked surveillance devices are revolutionizing the surveillance industry by supporting high quality images, remote monitoring, and advanced image processing. However, they raise serious security/privacy issues too. To address the issues, we present a new protocol, called SVS (Secure Video Surveillance), to securely transmit and store surveillance images, improving not only the performance but also the security in terms of the data confidentiality, integrity, and freshness to alleviate eavesdropping, packet modification, injection, and replay threats. For performance evaluation, we have implemented our protocol to compare its performance to a state-of-the-art baseline widely adopted in the surveillance industry. Our experimental results demonstrate that SVS substantially decreases the total delay to process the cryptographic mechanisms and increases the throughput. Thus, the results empirically verify the cost-effectiveness of SVS.

Being that an increasing number of wireless and hand-held surveillance systems are becoming available, novel research opportunities accompany them. As a result of this growth, privacy/security concerns could be further escalated. Our approach can be applied to mitigate these concerns, while enhancing the performance in terms of the security delay and throughput. A thorough investigation is reserved as a future work where we intend to investigate more cost-effective approaches to further enhance the performance and security of surveillance.

Acknowledgment

This work was supported, in part, by Samsung Techwin (Agreement #: 13052059).

References

[1] J. Wang and Z. Xu, “Crowd Anomaly Detection for Automated Video Surveillance,” in *International Conference on Imaging for Crime Detection and Prevention*, 2015.

[2] “What to expect from security and surveillance monitoring solutions in an IoT world.” [Online]. Available: http://blog.bosch-si.com/categories/technology/2013/07/what-to-expect-from-security-and-surveillance-monitoring-solutions-in-an-iot-world/ [Last visited on 06/12/2015]

[3] N. Goud, “How long a surveillance video should be stored?” [Online]. Available: http://blog.dnfcorp.com/?p=2078 [Last visited on 06/12/2015]

[4] Z. Li, X. Wang, Y. Lin, and C. Cheng, “RDEA: A Novel Video Encryption Algorithm,” *Advanced Multimedia and Ubiquitous Engineering, Lecture Notes in Electrical Engineering*, vol. 352, pp. 183–189, 2015.

[5] S. Li, C. Li, K.-T. Lo, and G. Chen, “Cryptanalysis of an Image Scrambling Scheme Without Bandwidth Expansion,” *IEEE Trans. Circuits Syst. Video Techn.*, vol. 18, no. 3, pp. 338–349, 2008.

[6] B. Furht and D. Kirovski, Eds., *Multimedia Security Handbook*. CRC Press, 2014.

[7] “OpenSSL: The Open Source Toolkit for SSL/TLS.” [Online]. Available: https://www.openssl.org/ [Last visited on 06/12/2015]

[8] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 6th ed. Pearson, 2013.

[9] W. Gao, Y. Tian, T. Huang, S. Ma, and X. Zhang, “The IEEE 1857 Standard: Empowering Smart Video Surveillance Systems,” *IEEE Intelligent Systems*, vol. 29, Sept 2014.

[10] L. Meinel, M. Findeisen, M. Hes, A. Apitzsch, and G. Hirtz, “Automated Real-Time Surveillance for Ambient Assisted Living Using an Omnidirectional Camera,” in *IEEE International Conference on Consumer Electronics*, Jan 2014.

[11] H. Liu, S. Chen, and N. Kubota, "Intelligent Video Systems and Analytics: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1222–1233, Aug 2013.

[12] T. Zhang, S. Liu, C. Xu, and H. Lu, "Mining Semantic Context Information for Intelligent Video Surveillance of Traffic Scenes," *IEEE Transactions on Industrial Informatics*, Feb 2013.

[13] D. Bruckner, C. Picus, R. Velik, W. Herzner, and G. Zucker, "Hierarchical Semantic Processing Architecture for Smart Sensors in Surveillance Networks," *IEEE Transactions on Industrial Informatics*, May 2012.

[14] A. Mahendran, S. Smith, M. Hebert, , and X.-F. Xie, "Bus Detection for Adaptive Traffic Signal Control," CMU-Penn T-SET – A U.S. DOT University Transportation Center, Tech. Rep., 2014.

[15] C. Poppe, G. Martens, P. De Potter, and R. Van de Walle, "Semantic web technologies for video surveillance metadata," *Multimedia Tools and Applications*, vol. 56, no. 3, pp. 439–467, 2012.

[16] T.-C. Wang, C.-H. Wang, R.-I. Chang, and J.-M. Ho, "Ubiquitous video surveillance service with secure forwarding agents," in *Asia-Pacific Conference on Communications*, 2008.

[17] Y. Akman and T. Yerlikaya, "Encryption Time Comparison of AES on FPGA and Computer," in *Advances in Computational Science, Engineering and Information Technology*, 2013, vol. 225, pp. 317–324.

[18] E. Barker, M. Smid, D. Branstad, and S. Chockhani, "A Framework for Designing Cryptographic Key Management Systems," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-130, 2013.

[19] "RTP: A Transport Protocol for Real-Time Applications." [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt [Last visited on 08/05/2015]

[20] "Surreal Flight Over Binghamton, NY." [Online]. Available: http://www.youtube.com/watch?v=P9MqHfXFuLk [Last visited on 08/26/2015]

Udaya L. N. Puvvadi received his MS degree in Computer Science from the State University of New York at Binghamton, and is currently pursuing a Ph.D. degree in the same department. His research interests include energy efficient systems and real-time video streaming.

Kevin Di Benedetto received both his BS degree and MS degree in Computer Science from the State University of New York at Binghamton. His research interests include real-time data services, m-Health, and medical cyber physical systems.

Aditya Patil received his MS degree in Computer Science from the State University of New York at Binghamton. He is a software engineer at Intel Corp. His research interests include computer architecture and real-time video streaming.

Kyoung-Don Kang is an Associate Professor in the Department of Computer Science at the State University of New York at Binghamton. He received his Ph.D. degree in Computer Science from the University of Virginia in 2003. His research areas include real-time data services, cyber physical systems, and Internet of Things.

Youngjoon Park is a principal research engineer in the Security Solution Division at Hanwha Techwin Corp., Korea. He graduated as Master of Science in the field of automation design engineering from KAIST (Korea Advanced Institute of Science and Technology). He has developed expertise in video security and storage products in the domain of surveillance and delivered well known world class IP Security products and solutions on behalf of Techwin.