AuthCropper: Authenticated Image Cropper for Privacy Preserving Surveillance Systems

Jihye Kim[1], Jiwon Lee[2], Hankyung Ko[2], Donghwan Oh[2], Semin Han[2], Kwonho Jeong[2], and Hyunok Oh[2(✉)]

[1] Kookmin University, Seoul, Korea,
{jihyek}@kookmin.ac.kr
[2] Hanyang University, Seoul, Korea,
{jiwonlee,hankyungko,donghwanoh,saemin2700,jkho1229,hoh}@hanyang.ac.kr

Abstract. As surveillance systems are popular, the privacy of the recorded video becomes more important. On the other hand, the authenticity of video images should be guaranteed when used as evidence in court. It is challenging to satisfy both (personal) privacy and authenticity of a video simultaneously, since the privacy requires modifications (e.g., partial deletions) of an original video image while the authenticity does not allow any modifications of the original image. This paper proposes a novel method to convert an encryption scheme to support partial decryption with a constant number of keys and construct a privacy-aware authentication scheme by combining with a signature scheme. The security of our proposed scheme is implied by the security of the underlying encryption and signature schemes. Experimental results show that the proposed scheme can handle the UHD video stream with more than 17 fps on a real embedded system, which validates the practicality of the proposed scheme.

Keywords: Privacy, Authentication, Forward-secure signature, Video

1 Introduction

A video from surveillance systems plays an important role in investigating incidences of crime and is often adopted as evidence in courts. However, it is getting harder to assure that the video frame submitted as evidence is equivalent to the original image as image editing skills and tools are more sophisticated; it is pretty simple to add or remove a person or a thing by editing the frame. For its purpose as evidence, the authentication procedure of a video is prerequisite. Another desirable requirement for video authentication is forward security. In general, malicious video manipulation is attempted *after* an undesirable event happens. Everyone who is trying to hide his/her unsuitable past behavior is in range of possible attackers. In the situations where even authorities try to modify the past CCTV video for their advantage with their authority[3], forward security is essential to fundamentally block a forgery against any past event.

[3] In July 2010, London police were accused of editing the original CCTV videos to use it in the court[Cut10]

A (personal) privacy protection is an orthogonal issue to authentication but required for the proper use of video evidence. It is because original raw video images collected by surveillance devices may contain personal sensitive data of a witness, victim, or unrelated third party. Image masking is one of the most common ways to efficiently hide some portions of an image. Many existing works[Mou01,RD02] propose watermark or masking based image hiding techniques, to gain the personal privacy within a video.

Authentication vs Privacy. Masking and authentication, however, are contradictory concepts in terms of their properties; while the authentication verifies that the image is *not modified*, masking *modifies* a portion of the original image. When the masking technique is applied to the image, the image is no longer original, and the digital signature is not valid anymore. Currently in the real world, the public is obligated to trust the image-masking authority completely, but it is desirable to let even the authority at least provide a proof (i.e. valid signature) that the image from the past is not modified maliciously.

PASS scheme (privacy preserving surveillance system) in [KLY+17] overcomes this issue by designing a new signature scheme with an authorized deletion function. PASS detects each object as a deletion unit using a deep learning algorithm and limits its size overhead to the number of objects. Still, the signature size in PASS increases by the number of objects, and its deletion availability is bound to the object detection. Therefore it is desirable to design a partially deletable authenticated video scheme with constant signature/key size overhead without defining deletable objects in advance.

Encrypt-and-Sign. To satisfy both privacy and authenticity requirements, we consider a method to encrypt every pixel using different symmetric keys in a video frame and generate a signature on the encrypted frame, i.e., encrypt-and-sign. It achieves authenticity and provide controllable privacy; the signature verifies the image authenticity, while the pixel level encryption allows to decrypt only required image portion given the corresponding keys. In this solution, the number of keys grows in proportion to the number of pixels; for example, a 1080p video frame requires 1920×1080 keys to encrypt each pixel, which is a heavy burden to be deployed in practice.

If we utilize a hash chain, the size overhead can be reduced into a constant. An initial seed key generates the rest of keys of each pixel in a sequential order through a hash chain. The limitation of this approach is that it is difficult to give a secret key so that only the specified portion of images is decrypted, because one key generates every keys of its descendants in the sequential order. We overcome this issue by applying hash chains independently in four directions: left, right, up, and down. Each pixel is encrypted/decrypted with the four keys. It is possible to decrypt a particular rectangular area in which the four chain keys can be generated corresponding to every pixel. Refer to section 4.2 for the details.

AuthCropper. In this paper, we propose a crop supporting authenticated surveillance system called AuthCropper (Authenticated Image Cropper for Surveillance Systems). It is built by sequentially composing encryption and signature

schemes as an authenticated encryption scheme. Since the default image remains as a ciphertext, we can guarantee the personal privacy within the video frame. Instead of deleting the unrelated part of images, it *discloses* the related part of images. It supports the partial decryption, i.e., a crop by providing the corresponding key. By applying the hash chain in a novel way, the proposed AuthCropper requires the constant number of keys even without defining deletable object in advance by additional fancy systems like AI as in [KLY+17]. Encryption units can range from pixel to full image, controlling the trade-off relationship between encryption performance and privacy quality; A smaller unit takes long encryption time to generate more keys, while a larger unit opens images in a coarser-grained rectangle. AuthCropper ensures authorized disclosure and privacy protection through decryption key generation. According to our experiment, the proposed scheme can encrypt and sign more than 17 fps on a real embedded system, which validates the practicality of the proposed scheme. The proposed scheme is secure under the security of the underlying encryption and signature schemes.

We begin by discussing the related work in Section 2. In Section 3, we describe preliminaries of the AuthCropper scheme. Section 4 introduces the overall workflow of the proposed scheme briefly, constructs the proposed scheme, and shows the security proof. Experimental results are shown in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

We overview authentication techniques that support some set of permissible image processing.

Semi-fragile watermarks[SSY02],[LC] withstand the normal transformation such as compression. In [SSY02], a semi-fragile watermark was embedded in coefficients of SVD decomposition of image blocks in a way that tolerates JPEG compression. In [LC], an authentication string embedded in the image DCT coefficients using two JPEG compression invariants, makes the watermark robust to JPEG compression.

Content based authentication schemes[VKJM00,LL03] which encode unique features from the source image to form digital signatures provide the authenticity of an image, while allowing content preserving transformations. Venkatesan et al.[VKJM00] develop an image hash based on an image statistics vector that stays invariant under a large class of content-preserving modifications to the image. A structural signature scheme was proposed in [LL03] by identifying the stable relationship between a parent-child pair of coefficients in the wavelet domain. These schemes not based on the cryptographic hash functions are known to be vulnerable to the content changing attack as demonstrated in [LWDD15].

Johnson et al.[JWL11] propose a scheme for image authentication based on a merkle hash tree, while allowing a public deletion of granulated blocks. A merkle hash tree is constructed assuming each granulated blocks as leaf nodes.

By keeping the hash value, a particular block of an image can be deleted by anybody.

Photoproof [NT16] implements SNARK[BCCT12] proof to digital images. SNARK (Succinct Non-interactive ARguments of Knowledge) is a cryptographic concept of verifying the outsourced computation. The Photoproof allows authenticated users to transform the image, while providing a succinct proof that can prove the transformation that has been executed. In this approach public verifiers can verify the image authenticity with some modifications allowed. However, it lacks practicality due to the SNARK complexity and large key size. For instance, it takes 55700 seconds to generate a proof in the transformation with 460GB proving key for a 100x100 size image.

PASS[KLY$^+$17] is proposed to ensure privacy and authenticity of the image simultaneously. PASS detects objects in a video frame using deep learning image processing, and generates signatures such that partial deletions are allowed in an authorized way. It implements chameleon hash scheme[ACDMT05] which finds a hash collision for different message with a secret key. Thus, chameleon hash based signatures allows a portion of areas to be deleted in a valid way. The PASS allows an authorized deleter to distort the objects that are unrelated to the occasion, to keep the privacy. However, this approach has some limitations; its privacy controllability is affected by the object detection algorithm; if an object is not detected by the deep learning algorithm, the privacy of the object cannot be preserved. Its time/storage complexities due to the signature algorithm increases linearly with the number of objects, which becomes burdens to be efficiently deployed in practice.

3 Preliminaries

In this section, we overview a forward secure signature scheme [IR01,KO17] that our proposed scheme is based on. And then, we describe the definition of AuthCropper (Authenticated Image Cropper for Surveillance Systems) and its security notions.

3.1 Forward Secure Signature Scheme

A forward secure signature scheme is a key evolving digital signature scheme, which ensures that even if a signing key is leaked at the current period, the authenticity is preserved for the previous periods. A forward secure signature scheme consists of four algorithms Σ_{FS}=(KeyGen, Update, Sign, Verify) such that:

KeyGen(T, λ) : It receives a security parameter λ and a total time period T, and outputs a pair of key (pk, sk).

Update(sk) : It receives the secret key for the current time period, and outputs sk' for the next time period.

$\mathsf{Sign}(sk, m)$: It outputs a signature σ on the message m using the secret key at the current period.

$\mathsf{Verify}(pk, m, \sigma)$: It outputs 1 if a signature σ on message m is valid with the public key pk and 0 otherwise.

In the forward secure signature scheme, although secret key sk is leaked at time b, it is impossible to forge a signature with past time period b' where $b' < b$.

3.2 Model

We define AuthCropper and its security notions similarly with the scheme[KLY+17]. In the system, a video in a surveillance device consists of multiple frames. To guarantee controllable privacy and authenticity of the video, AuthCropper encrypts each frame using a symmetric key encryption scheme and generates a signature on the encrypted frame AND its decryption key tag. We note that AuthCropper authenticates the image decrypted from the authenticated ciphertext and its decryption key matching the authenticated key tag. (Notice that the decryption key tag verifies only whether a specific decryption key is valid.)

When the recorded frames are used as evidence, it is required to reveal a part of the image frames. The authorized manager determines a rectangle to be revealed in each encrypted image frame. And then the manager generates a decryption key ck which can decrypt the chosen rectangle in the frame. If the validity of a given decryption key ck is verified from the key tag then the decryption is performed correctly; otherwise the given encrypted frame and the decryption key are rejected as evidence and the decryption process is aborted.

The AuthCropper consists of five algorithms $\Pi_{AC} = (\mathsf{Setup}, \mathsf{Update}, \mathsf{AuthEnc}, \mathsf{ExtKey}, \mathsf{AuthDec})$ such that:

$\mathsf{Setup}(T, \lambda)$ takes as inputs a maximum time period T and a security parameter λ. It outputs a master key msk, a public verification key pk, and a forward secure signing key sk initialized for period 0.

$\mathsf{Update}(sk)$ takes as input the signing key sk for the current period and outputs an updated signing key sk' for the next time period.

$\mathsf{AuthEnc}(msk, sk, M, i)$ takes as inputs a master key msk, a signing key sk, a frame image M, and its frame index i. It outputs a ciphertext and its signature (C, σ) where a ciphertext includes an encrypted frame and its decryption key tag or $C = (E, t)$.

$\mathsf{ExtKey}(msk, i, R)$ takes as inputs a master key msk, a frame index i, and an area $R = (x_1, y_1, x_2, y_2)$ which denotes a rectangle area between top left point (x_1, y_1) and bottom right point (x_2, y_2), and outputs key ck to decrypt the corresponding area R in an encrypted frame of which frame index is i.

$\mathsf{AuthDec}(pk, ck, (C, \sigma))$ takes as inputs a verification key pk, a decryption key ck, a ciphertext $C(= (E, t))$ and its signature σ. It verifies σ on C using pk and checks the validity of ck using t; if the verification fails, it outputs $\perp$. Otherwise, it decrypts a rectangle R region using ck and E correctly.

An AuthCropper scheme must satisfy the following properties:

Correctness: AuthDec with decryption key ck from ExtKey must decrypt a corresponding encrypted image produced from AuthEnc and reveal area R in the image.

$$(C, \sigma) \leftarrow \mathsf{AuthEnc}(msk, sk, M, j); ck \leftarrow \mathsf{ExtKey}(msk, i, R);$$
$$\mathsf{AuthDec}(pk, ck, (C, \sigma)) = \big[\, [M]_p \qquad \text{if} \quad p \in R \wedge i = j \,\big]$$

where $[M]_p$ represents a pixel value at position p in image M.

Forward Secure Immutability: The adversary should not manipulate any past image even with the access to the master key and the current signing key. Since the pair of a ciphertext and a key is mapped to a plaintext uniquely by decryption, there are two scenarios to manipulate the plaintext image. The adversary wins if he forges a valid signature for a modified ciphertext or a manipulated decryption key. This attack should not allowed even if the adversary has access to the master encryption key and the current signing key.

Formally, Π_{AC} is forward secure immutable, if for all PPT adversaries $\mathcal{A}$ there is a negligible function ϵ such that:

$$Pr\left[\begin{array}{l}(msk, sk, pk) \leftarrow \mathsf{Setup}(T, \lambda);\\ b \leftarrow \mathcal{A}^{\mathsf{AuthEnc}(\cdot),\mathsf{Update}(\cdot),\mathsf{Break\text{-}in}(\cdot)}(pk, msk);\\ (b', ck^*, C^*, \sigma^*) \leftarrow \mathcal{A}(sk_b) : \mathsf{AuthDec}(pk, ck^*, (C^*, \sigma^*)) \neq \perp\end{array}\right] \leq \epsilon$$

where sk_i is the signing key at period i and b is the time period at which Break-in was issued. $(b', ck^*, C^*, \sigma^*)$ are generated by the attacker, where ck^* is a decryption key, C^* is a ciphertext, and σ^* is a signature of C^* at period b'. If $b' < b$, $\exists(ck^*, C^*, \sigma^*) : (C^*, \sigma^*) \neq (C, \sigma)$ for any AuthEnc query output (C, σ) at period b' and σ^* is the valid signature of C^*, or $ck^* \notin \{ck : \forall R, ck \leftarrow \mathsf{ExtKey}(msk, b', R)\}$, successfully passing AuthDec, $\mathcal{A}$ wins.

Authorized Disclosure: Π_{AC} discloses in an authorized way, if for all PPT adversaries $\mathcal{A}$ there is a negligible function ϵ such that:

$$\left| Pr\left[\begin{array}{l}(msk, sk, pk) \leftarrow \mathsf{Setup}(T, \lambda); i \leftarrow \mathcal{A}(pk, sk);\\ M_0, M_1 \leftarrow \mathcal{A}^{\mathsf{ExtKey}(\cdot,j,\cdot)}(pk, sk)\ s.t.\ j \neq i;\\ b \leftarrow \{0,1\}; (C, \sigma) \leftarrow \mathsf{AuthEnc}(msk, sk, M_b, i);\\ b' \leftarrow \mathcal{A}^{\mathsf{ExtKey}(\cdot,j,\cdot)}(pk, sk, C, \sigma) \quad : b = b'\end{array}\right] - \frac{1}{2} \right| \leq \epsilon$$

where adversary $\mathcal{A}$ can query $\mathsf{ExtKey}(\cdot, j, \cdot)$ such that all $j \neq i$. This definition informs that if msk and ck for frame i are not provided then no information about the frame i is leaked. Note that $\mathsf{ExtKey}(\cdot, j, R)$ provides a (encryption/decryption) key for frame j when R includes a whole frame area and $\mathsf{AuthEnc}(\cdot)$ can be computed using the key in $\mathcal{A}$.

Area Privacy: The definition is extended from the above authorized disclosure. All areas except the revealed rectangle area must be private. If the two different

frames with the same size have an identical revealing area R, they should be indistinguishable. Formally Π_{AC} is area private, if for all PPT adversaries $\mathcal{A}$ there is a negligible function ϵ such that:

$$\left| Pr \left[\begin{array}{l} (msk, sk, pk) \leftarrow \mathsf{Setup}(T, \lambda); i, R \leftarrow \mathcal{A}(pk, sk); \\ M_0, M_1 \leftarrow \mathcal{A}^{\mathsf{ExtKey}(\cdot,j,\cdot),\mathsf{ExtKey}(\cdot,i,R')}(pk, sk) \\ \ where\ \forall p \in R, [M_0]_p = [M_1]_p; \\ b \leftarrow \{0,1\}; (C, \sigma) \leftarrow \mathsf{AuthEnc}(msk, sk, M_b, i); \\ b' \leftarrow \mathcal{A}^{\mathsf{ExtKey}(\cdot,j,\cdot),\mathsf{ExtKey}(\cdot,i,R')}(pk, sk, C, \sigma) \\ : b = b' \end{array} \right] - \frac{1}{2} \right| \leq \epsilon$$

where adversary $\mathcal{A}$ can query $\mathsf{ExtKey}(\cdot, j, \cdot)$ such that all $j \neq i$, and $\mathsf{ExtKey}(\cdot, i, R')$ such that $R' \subset R$. Similarly to Authorized Disclosure, $\mathsf{ExtKey}(\cdot, j, R')$ simulates $\mathsf{AuthEnc}(\cdot)$ when R' represents a whole frame area.

4 Authenticated Image Cropper for Surveillance Systems

4.1 Overview

The main objective of our AuthCropper is to authenticate the given video frame (i.e. authentication), and publicly disclose issuable area only (i.e. privacy). To ensure both authenticity and controllable privacy, we adopt the concept of forward secure signature, and partially decryptable encryption. With the assistance of two schemes, the workflow of AuthCropper consists of two phases: *ordinary phase* and *submission phase*, as in figure 1.

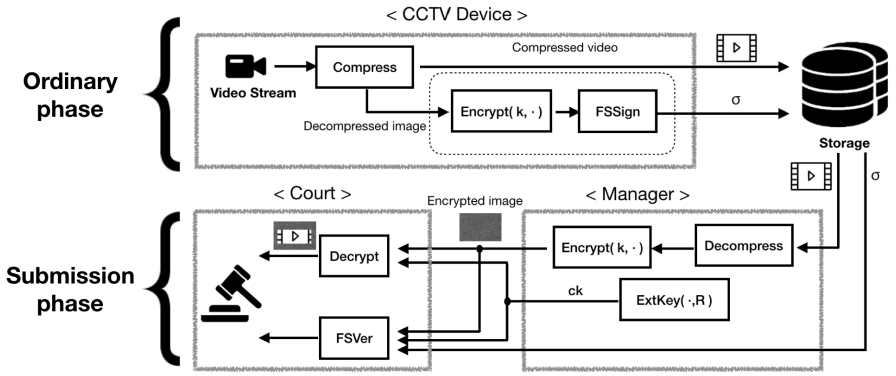


Fig. 1: Authenticated Image Cropper for Surveillance Systems

Ordinary Phase. The ordinary phase indicates an original CCTV streaming phase, where the CCTV records the on-going video frames. When an ordi-

nary CCTV device records the video, it compresses the video stream with an internal reconstruction of decompressed images. With leveraging this original decompressed image, we add an encrypt-and-sign method simultaneously; we first encrypt the decompressed image and then sign the encrypted image[4] with a forward secure signature scheme. We store these digital signatures (for each frame) along with the original compressed video, so that the encrypted format can be verified even when reconstructed for the submission to the court. Namely, the AuthCropper CCTV storage stores the original compressed video same as traditional devices, and additional signatures for each frame.

Submission Phase. The submission phase indicates a post-processing phase, where the image is submitted to the court as a formal evidence. When the captured images are requested from the court, they are provided by an authorized manager who manages the privacy of the captured image. The authorized manager receives compressed video data and digital signatures which are generated by the CCTV device, in the ordinary phase. The manager decompresses the image, and encrypts the decompressed image to generate an encrypted message E. Since the device signed the encrypted image in the ordinary phase, it is valid for the reconstructed message E. Note that the CCTV device and the manager share the same encryption key msk while a signing key sk is kept only in the CCTV device. The encrypted image E preserves privacy for the video frame if no decryption key is provided. The manager determines an area to be revealed by examining the frame, and then generates a decryption key ck which can only decrypt the selected area in the encrypted image.

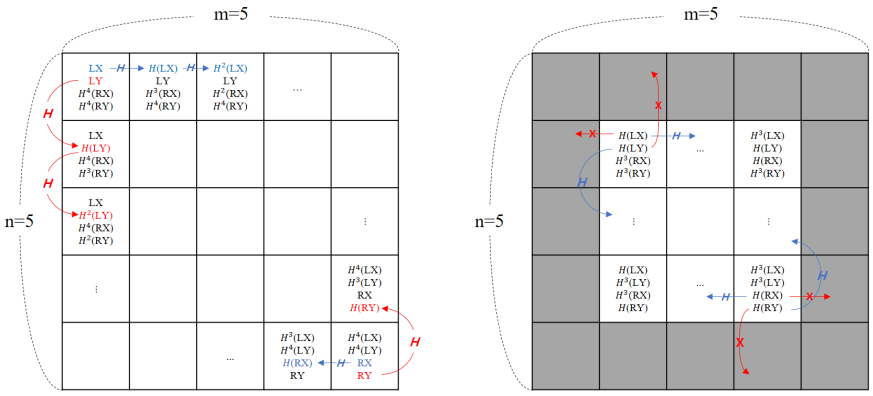
The encrypted image, the signature, and the decryption key become a legal evidence to prove the authenticity while preserving the privacy of the unrevealed area. In court, it is checked if the given signature σ is valid for the ciphertext and the decryption key matches the key information in the ciphertext. If the signature is invalid or the decryption key is mismatched then the data is rejected and the video cannot be accepted as a legal evidence in court. Otherwise, the encrypted image E is decrypted using key ck. The output image preserves privacy, because it is disallowed to decrypt the area which is not covered by key ck. Consequently, the video maintaining privacy is finally used as a legal digital evidence.

4.2 Idea for Short Keys

Figure 2 shows an image example with 5×5 pixel blocks in the proposed AuthCropper. As in figure 2a, each image frame of the video is divided into a grid matrix of blocks called pixel block (we can control the block granularity). Each pixel block (x, y) contains a set of four keys (LX_x, LY_y, RX_x, RY_y). The keys are generated by using hash chains from (LX_1, LY_1, RX_m, RY_n) where the frame image is composed of $m \times n$ pixel blocks ($m = 5, n = 5$ in this example) by applying a one-way hash function H repeatedly. More specifically, the intention is that the hash chain of LX starts from left to right, LY from top to bottom,

[4] We sign the corresponding encryption key (anchor key) along with the image, to also authenticate the key chain.

RX from right to left, and RY from bottom to top. In this way, each block has a unique key set, since at least one of the four components would be different from the other blocks. Note that $H^n(x)$ denotes that H is applied n times. LX_1, LY_1, RX_5, and RY_5 are keys for left, top, right, and bottom blocks, respectively. LX and RX, and LY and RY evolve along the x and y axes respectively, in different direction with applying H.



(a) Encryption (whole image) (b) Decryption (open area)

Fig. 2: The main idea of encryption and decryption (5×5 example)

For encryption and decryption, the AuthCropper adopts a stream cipher which is an XOR based encryption scheme; it utilizes any pseudorandom generator (PRG) to generate a stream cipher key, and performs an XOR operation of the input and the key for both encryption and decryption. For each pixel block (x, y), a stream cipher key K is generated by providing a set of four keys (LX_x, LY_y, RX_x, RY_y) to PRG. Then the encrypted image block is computed with $e = M \oplus K$, where M is an original image block and K is a stream cipher key.

The decryption key generation for the area of $(x_L, y_L) \sim (x_R, y_R)$ is managed by an authorized entity, by providing the $(LX_{x_L}, LY_{y_L}, RX_{x_R}, RY_{y_R})$. To check the validity of the decryption key, an anchor key is computed using the given decryption key. If the computed anchor key is equal to the anchor key in the ciphertext then the decryption process proceeds. With the decryption key, every stream cipher key can be generated in the specified area while one of LX, LY, RX, and RY cannot be computed in the outside of the area. Figure 2b illustrates how to generate a stream cipher key for each pixel block. Assume that the manager chooses the area of block $(2, 2) \sim (4, 4)$ to be revealed. The manager first computes a key of $(LX_2, LY_2, LX_4, LY_4) = (H(LX_1), H(LY_1), H(RX_5), H(RY_5))$ for

an image composed of 5×5 pixel blocks. With the key of (LX_2, LY_2, LX_4, LY_4), every stream cipher key in the selected area can be generated. For instance, a stream cipher key for a block $(3, 4)$ is $(LX_3, LY_4, RX_3, RY_4) = (H(LX_2), H(H(LY_2)), H(RX_4), RY_4)$. For any block outside of the area, one of keys is not computable at least. For example, consider a block $(1, 3)$. The key becomes $(LX_1, LY_3, RX_1, RY_3) = (LX_1, H(LY_2), H(H(H(RX_4))), H(RY_4))$. However, LX_1 is not computable from LX_2.

Anchor Key. Since an original image block is uniquely determined by an encrypted image block and a stream cipher key, a signature should be generated for not only the encrypted image block but also the stream cipher key. To validate the key chain, the AuthCropper devises an anchor key which is a hash value of the last keys in the key chains or $AK = H(LX_{m+1}||LY_{n+1}||RX_0||RY_0)$. A ciphertext in fact also includes the anchor key along with the encrypted image blocks. In decryption example of figure 2b, an anchor key for the frame is computed using the given decryption key or $AK = H(LX_6||\ LY_6||\ RX_0||\ RY_0) = H(H^4(LX_2)||\ H^4(LY_2)||\ H^4(RX_4)||\ H^4(RY_4))$.

4.3 Construction

Let Σ_{FS} = (FSKeyGen, FSUpdate, FSSign, FSVerify) be a standard forward-secure signature scheme. Also let PRG be a pseudorandom generator and H be a one-way hash function. We assume that a pixel block size parameter is given as bs and construct the AuthCropper by utilizing Σ_{FS}, PRG, and H. The AuthCropper Π_{AC} = (Setup, Update, AuthEnc, ExtKey, AuthDec) is constructed as follows:

Setup(T, λ): The Setup algorithm outputs a master key msk by choosing a random value and a secret key sk by utilizing Σ_{FS}:

$msk \xleftarrow{\$} \{0,1\}^\lambda$; $(pk, sk) \leftarrow$ FSKeyGen(T, λ)
return (msk, pk, sk)

Update(sk_t): It updates the forward secure signature secret key sk_t to sk_{t+1}:

$sk_{t+1} \leftarrow$ FSUpdate(sk_t)
return sk_{t+1}

AuthEnc(msk, sk, M, i): We divide image M into multiple pixel blocks where each pixel block is $bs \times bs$ square. The image size of M is $(bs \times m) \times (bs \times n)$, and there are $m \times n$ pixel blocks in a frame. For each pixel block, a stream cipher is applied for encryption. To ensure the authenticity of the key chains, the hash of the final chain keys called an anchor key is included in a ciphertext and a signature is generated for the encrypted image and the key. Note that no area can be decrypted with an anchor key. The resulting algorithm is as follows:

$LX_1 \leftarrow H(msk, i, 0)$; $LY_1 \leftarrow H(msk, i, 1)$
$RX_m \leftarrow H(msk, i, 2)$; $RY_n \leftarrow H(msk, i, 3)$
for all $1 \le x \le m, 1 \le y \le n$ **do**
  $LX_x \leftarrow H^{x-1}(LX_1)$; $LY_y \leftarrow H^{y-1}(LY_1)$

$RX_x \leftarrow H^{m-x}(RX_m)$; $RY_y \leftarrow H^{n-y}(RY_n)$
$XK_{x,y} \leftarrow PRG(LX_x, LY_y, RX_x, RY_y)$
$[E]_{x,y} \leftarrow [M]_{x,y} \oplus XK_{x,y}$
end for
$AK \leftarrow H(LX_{m+1}||LY_{n+1}||RX_0||RY_0)$
$C \leftarrow E||AK$
$\sigma \leftarrow \mathsf{FSSign}(sk, C)$
return (C, σ)

where $[M]_{x,y}$ (or $[E]_{x,y}$) presents a pixel block at position (x, y) in M (or E), and $||$ denotes concatenation.

$\mathsf{ExtKey}(msk, i, R)$: An area R is compiled as $R = (bs \times x_L, bs \times y_L, bs \times x_R, bs \times y_R)$. Decryption key ck for R is computed by applying hash function repeatedly:

$LX_1 \leftarrow H(msk, i, 0)$; $LY_1 \leftarrow H(msk, i, 1)$
$RX_m \leftarrow H(msk, i, 2)$; $RY_n \leftarrow H(msk, i, 3)$
$LX_{x_L} \leftarrow H^{x_L - 1}(LX_1)$; $LY_{y_L} \leftarrow H^{y_L - 1}(LY_1)$
$RX_{x_R} \leftarrow H^{m-x_R}(RX_m)$; $RY_{y_R} \leftarrow H^{n-y_R}(RY_n)$
return $ck = (LX_{x_L}, LY_{y_L}, RX_{x_R}, RY_{y_R})$

$\mathsf{AuthDec}(pk, ck, (C, \sigma))$: takes inputs a verification key pk, a decryption key ck, a ciphertext $C(= E||AK)$, and an authentication tag σ. First it verifies the signature to C using a forward secure signature algorithm and checks whether ck is a valid key against anchor key AK by applying H. If it passes, it decrypts E using key ck. Note that decryption key $ck = (LX_{x_L}, LY_{y_L}, RX_{x_R}, RY_{y_R})$ is used to decrypt an encrypted frame for an area $(bs \times x_L, bs \times y_L, bs \times x_R, bs \times y_R)$.

Parse $C = (E, AK)$
If $\mathsf{FSVerify}(pk, C, \sigma) = 0$ **return** $\perp$
If $AK \neq H(H^{m+1-x_L}(LX_{x_L}) \,||\, H^{n+1-y_L}(LY_{y_L}) \,||\, H^{x_R}(RX_{x_R}) \,||\, H^{y_R}(RY_{y_R}))$
return $\perp$
for all $x_L \leq x \leq x_R, y_L \leq y \leq y_R$ **do**
$LX_x \leftarrow H^{x-x_L}(LX_{x_L})$; $LY_y \leftarrow H^{y-y_L}(LY_{y_L})$
$RX_x \leftarrow H^{x_R-x}(RX_{x_R})$; $RY_y \leftarrow H^{y_R-y}(RY_{y_R})$
$XK_{x,y} \leftarrow PRG(LX_x, LY_y, RX_x, RY_y)$
$[M]_{x,y} \leftarrow [E]_{x,y} \oplus XK_{x,y}$
end for
return M

4.4 Security Proof

In this section, we provide security proofs for the proposed AuthCropper, which are defined in section 3.

Theorem 1. *Assuming that Σ_{FS} is forward secure with advantage Adv_{FS} and H is a second pre-image resistance hash function with advantage Adv_H in t_H, the proposed scheme Π_{AC} satisfies forward secure immutability with advantage $Adv_{AC} \leq Adv_{FS} + 2(m+n) \times Adv_H$ in $t_{AC} \leq q_s(t_{Enc} + 2(m+n)t_H + t_{FSSign}) +$*

$q_u t_{FSUpdate}$, *where* q_s *and* q_u *denote the number of sign and update queries, respectively.* t_{sign} *and* t_{update} *represent the time of* Sign *and* Update *in* Σ_{FS} *and* t_{Enc} *indicates the encryption time of an image in* AuthEnc.

Proof. Assume that there is PPT adversary $\mathcal{A}$ to break the forward secure immutability with advantage Adv_{AC} in the proposed scheme Π_{AC}. We show there exists PPT adversary $\mathcal{B}$ that breaks the forward security of Σ_{FS} or the second pre-image resistance of H.

There are two scenarios for attacker $\mathcal{A}$ to generate a valid forward secure signature for a past image in period b'.

First, assuming that $\mathcal{A}$ finds (C^*, σ^*) that is not queried at period b' where σ^* is the valid signature of C^* with probability Adv_{FS}, we construct attacker $\mathcal{B}$ against Σ_{FS}. Given pk, $\mathcal{B}$ selects msk as in Π_{AC} and provides (msk, pk) for $\mathcal{A}$. To simulate $\mathsf{AuthEnc}(msk, sk, M, i)$, $\mathcal{B}$ computes $C = (E, t)$ following the protocol Σ_{FS} and obtains σ by querying FSSign on C. For the Update query, $\mathcal{B}$ requests the $\mathsf{FSUpdate}$ query. When $\mathcal{A}$ breaks in at period b, $\mathcal{B}$ also breaks in at period b and responds with sk_b. Finally, if $\mathcal{A}$ outputs $(b', ck^*, C^*, \sigma^*)$ verified by $\mathsf{AuthDec}$ then $\mathcal{B}$ outputs (C^*, σ^*).

Second, assuming that $\mathcal{A}$ finds ck^* which is not generated from msk while ck^* passes the anchor key check step in $\mathsf{AuthDec}$, it occurs due to the second pre-image attack. Let Adv_H denote the second pre-image resistance attack probability. Since the second pre-image collision attack can happen for each of $2(m+n)$ hash values, the collision probability is $2(m+n) \times Adv_H$.

Theorem 2. *Assuming that PRG is a pseudorandom generator and H is a pre-image collision resistant,* Π_{AC} *satisfies authorized disclosure and area privacy.*

Proof. Since the area privacy implies the authorized disclosure, the authorized disclosure is automatically proven when the area privacy holds. The area privacy is defined by the game where an adversary tries to distinguish C_0 a ciphertext of one area from C_1 a ciphertext of the other area except area $R = (x_1, y_1, x_2, y_2)$. For area R, assume that decryption key ck_R is generated as (LX_{x_1}, LY_{y_1}, RX_{x_2}, RY_{y_2}). Note that from the decryption key ck_R, a valid anchor key is generated by applying hash function H multiple times.

We describe a series of hybrid experiments G_0 - G_5, where the first game corresponds to the view of an adversary when M_0 is used as input for AuthEnc, while the final game corresponds to the view of an adversary when M_1 is used. The experiment G_0 is identical to the real area privacy experiment and the remaining G_1 - G_5 are progressively modified in such a way that each consecutive pair is proven to be indistinguishable.

Game G_0: This is the area privacy experiment described in Section 3.2 with the adversary getting output of $AuthEnc(msk, sk, M_0, i)$.

Game G_1: This is same as G_0 except that $PRG(K)$ is replaced by $PRG(K')$ where a component outside of R is chosen randomly in K'. Without loss of generality, assume that $x < x_1$. Then $K' = (LX'_x, LX_y, RX_x, RY_y)$ when K

$= (LX_x, LX_y,\ RX_x, RY_y)$ where LX'_x is randomly chosen. To distinguish two games, the adversary needs to succeed the pre-image attack and check if $H^{x_1-x}(LX'_x)$ is equal to LX_{x_1}. Since H is applied at most m or n, the adversary succeeds with probability of $(m+n) \times Adv_H$ where the probability to find the pre-image of hash function is Adv_H.

G_1 is computationally indistinguishable from G_0, i.e.,

$$|\Pr[\mathcal{A}(\mathsf{G}_0) = 1] - \Pr[\mathcal{A}(\mathsf{G}_1) = 1]| \leq (m+n) \times Adv_H \tag{1}$$

Game G_2: This is same as G_1 except that $PRG(K')$ is replaced by a truly random function $\mathcal{R}$. By the security of the PRG, G_2 is computationally indistinguishable from G_1, i.e.,

$$\Pr[\mathcal{A}(\mathsf{G}_1) = 1] - \Pr[\mathcal{A}(\mathsf{G}_2) = 1]| \leq Adv_{PRG} \tag{2}$$

where PRG and $\mathcal{R}$ is distinguishable with probability of Adv_{PRG}.

Game G_3: This is same as G_2 except that C is computed as an encryption of M_1 rather than M_0. Since the encryption is a XOR operation, it has perfect security. Therefore, G_3 is totally indistinguishable from G_2.

Game G_4: This is same as G_3 except that use $PRG(K')$ instead of $\mathcal{R}$. G_4 is computationally indistinguishable from G_3, i.e.,

$$|\Pr[\mathcal{A}(\mathsf{G}_3) = 1] - \Pr[\mathcal{A}(\mathsf{G}_4) = 1]| \leq Adv_{PRG} \tag{3}$$

Game G_5: This is same as G_4 except that K is used rather than K'. G_5 is computationally indistinguishable from G_4, i.e.,

$$|\Pr[\mathcal{A}(\mathsf{G}_4) = 1] - \Pr[\mathcal{A}(\mathsf{G}_5) = 1]| \leq (m+n) \times Adv_H \tag{4}$$

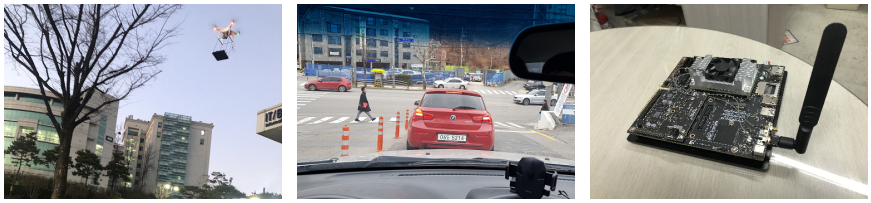
Finally, G_0 is computationally distinguishable from G_5, i.e.,

$$|\Pr[\mathcal{A}(\mathsf{G}_0) = 1] - \Pr[\mathcal{A}(\mathsf{G}_5) = 1]| \leq 2(m+n) \times Adv_H + 2Adv_{PRG} \tag{5}$$

Since PRG is a pseudorandom generator and H is a one-way function, Adv_{PRG} and Adv_H are both negligible. Thus, Π_{AC} satisfies area privacy.

5 Experiment

In this section, we show the results of our AuthCropper implementation on the real PC server and the embedded system with a camera - Jetson TX2 quad-core ARM Cortex-A57. Figures 3a and 3b show the captured images the experiment environment, with installing the Jetson TX2 board (figure 3c) on a drone cam and a black box camera. We implement the proposed AuthCropper with utilizing the OpenCV 3.4 library. In experiment, we consider the three implementations



(a) Drone cam (b) Vehicle black box (c) Applied board

Fig. 3: The actual capture of the AuthCropper experiment environment (embedded systems)

on a single core, with pre-computation of stream cipher keys, and on multi-core with the pre-computation.

Pre-Computation. In the pre-computation method, it is assumed that all stream cipher key $XK_{x,y}$ sets for each frame are constructed in advance. In the real-time recording, the device fetches the pre-computed XK sets and do the encryption (i.e. XOR operation) directly along with the forward-secure sign. In practice, by utilizing dedicated hardware hash accelerators such as Intel SHA extension and ARM cryptography engine, the pre-computation approach can be realized.

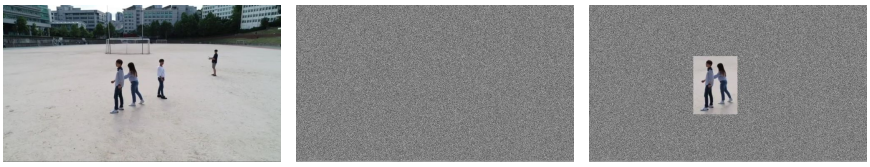
Parallel Processing. While there may be a control dependency among frames in video compression, there is no dependency in AuthEnc. Hence AuthEnc can be parallelized on multiple cores. The parallel processing optimization can improve the performance of AuthEnc by starting to execute AuthEnc for the next frame before AuthEnc for the current frame is completed.

We implement and compare the proposed AuthCropper without and with the optimization methods of pre-computation and parallel processing. For the building block, we utilize an existing signer-friendly scheme [KO17] for the forward secure signature, and SHA-256 hash for both pseudorandom generator and one-way hash function.

Recall that the workflow of our AuthCropper consists of two phases: *ordinary phase* and *submission phase* (Section 4.1, figure 1). The ordinary phase indicates the original CCTV streaming phase in real-time, while the submission phase is a post-processing phase that happens when the image (or video) is submitted to the court.

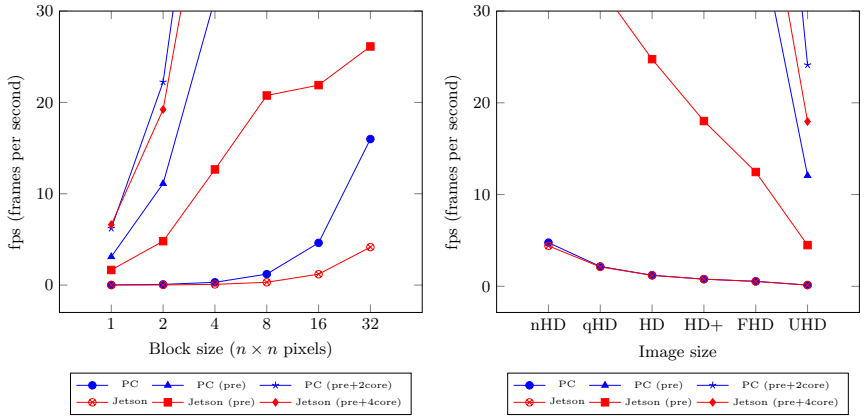
Since submission phase is not sensitive to the execution time, we mainly focus on the ordinary phase - specifically fps (frames per second) which is governed by the AuthEnc time. First we analyze the FPS of the video by varying the image size and the pixel block size, to observe if the AuthCropper can be practically applied to the current surveillance systems of at most 30 FPS. Then for the justification of submission phase, we briefly represent the execution time of ExtKey and AuthDec by varying the open area size. The experiments are performed on a PC server with Intel i5-4670 (3.40GHz dual-core), and a Jetson

TX2 embedded system (with an attached camera) with quad-core ARM Cortex-A57 (2GHz quad-core). Figure 4 represents a real implementation result of the AuthCropper, where an area in an image is decrypted from the encrypted image.



(a) Original image (b) Encrypted image (c) Partially revealed image

Fig. 4: The evidence generation in AuthCropper



(a) Block size (HD image) (b) Image size (16×16 pixels of granularity)

Fig. 5: FPS of the video by varying the block size and image size

Figure 5 shows the average fps results for the recorded video (ordinary phase) when varying the block size and the image size. PC, PC (pre), and PC (pre+2core) represent performance results on a PC server with no optimization, applying the pre-computation method, and applying both pre-computation and parallel processing on dual-core, respectively. Similarly, Jetson, Jetson (pre), and Jetson (pre+4core) indicate results on a Jetson board with no optimization, applying the pre-computation method, and applying both pre-computation and parallel processing on quad-core, respectively.

In figure 5a, x-axis indicates the size of a pixel block which has $n \times n$ pixels (e.g. 32 represents a pixel block of 32×32 pixels), and the y-axis illustrates the average FPS. Note that the image size is HD or (1280×720). The performance increases as the pixel block size increases since the number of pixel blocks decreases and the numbers of hash operations and memory accesses decrease. Unless the optimizations are applied, large size pixel blocks are required to satisfy 30 fps. However, if pre-computation and multi-core optimizations are utilized then the 30 fps can be easily reached even for 4×4 size pixel blocks.

Figure 5b shows the average fps results for a pixel block with 16×16 pixels by varying the frame image size among nHD (640×360), qHD (960×540), HD (1280×720), HD+ (1600×900), FHD (1920×1080), and UHD (3840×2160). As the frame image size increases, the fps results decrease since the number of pixel blocks increases and encryption time increases. With the full optimizations, the performance of 24 fps and 17 fps is achievable on PC and Jetson for UHD images.

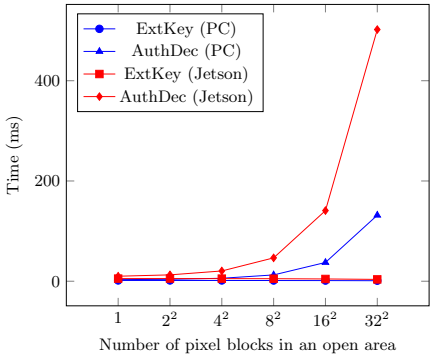


Fig. 6: ExtKey and AuthDec time by varying the open area size in UHD (16×16 pixels of granularity)

In the submission phase, the authorized manager determines the area to be opened in the image. Figure 6 shows the execution times of ExtKey and AuthDec algorithms for an ultra-HD (UHD) image depending by varying the size of the open area where a pixel block includes 16×16 pixels. Note that the performance of the submission phase is independent of the optimizations. ExtKey (PC) and AuthDec (PC) represent the results on the PC server, and ExtKey (Jetson) and AuthDec (Jetson) indicate the results on the Jetson board, respectively. The x-axis indicates the open area size of $n \times n$ *blocks*.In the ExtKey, the only required computation is hash chains for four sets of keys: two for top left (LX, LY) and two for bottom right (RX, RY). Thus the ExtKey remains constant. The execution time of the AuthDec algorithm is proportional to the open area size. As the open area size become large, the number of blocks to be

decrypted increases. The results show that the execution time is fast enough to be used in practice.

6 Conclusion

This paper presents a new Authenticated Image Cropper for Surveillance Systems. While a signature guarantees that an image frame is not altered, hiding of objects or masking is required in the frame for privacy. Since privacy and integrity are contradictory, it is a hard problem to solve the privacy preserving authentication problem.

The proposed scheme deals with the privacy preserving secure signature problem. Using the one-wayness of the hash function, our scheme allows to disclose only a selected part of video. Only an authorized manager examines the original video and determines an area that is relevant to the event. A forward signature is used in our proposed scheme, which provides forward secure immutability. That is, no past signature can be generated even after the current secret sign key is leaked.

Experimental results show that the proposed scheme is supposed to be utilized practically in a real-time video surveillance system due to its high performance of symmetric encryption and a far smaller signature size overhead compared to the previous works.

References

[ACDMT05] Giuseppe Ateniese, Daniel H. Chou, Breno De Medeiros, and Gene Tsudik. Sanitizable signatures. In *European Symposium on Research in Computer Security*, pages 159–177. Springer, 2005.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349. ACM, 2012.

[Cut10] Nicola Cutcher. CCTV and police abuse of power. `https://www.theguardian.com/commentisfree/2010/jul/19/gaza-protests-inquiry-police-cctv`, 2010.

[IR01] Gene Itkis and Leonid Reyzin. Forward-Secure signatures with optimal signing and verifying. In *Advances in Cryptology - CRYPTO 2001, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 332–354, 2001.

[JWL11] Rob Johnson, Leif Walsh, and Michael Lamb. Homomorphic signatures for digital photographs. In *Financial Cryptography*, pages 141–157. Springer, 2011.

[KLY+17] Jihye Kim, Seunghwa Lee, Jungjun Yoon, Hankyung Ko, Seungri Kim, and Hyunok Oh. Pass: Privacy aware secure signature scheme for surveillance systems. In *Advanced Video and Signal-based Surveillance (AVSS), 2017 IEEE Symposium on*. IEEE, 2017.

[KO17] Jihye Kim and Hyunok Oh. Forward-secure digital signature schemes with optimal computation and storage of signers. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 523–537. Springer, 2017.

[LC] C.-Y. Lin and S.-F. Chang. Semifragile watermarking for authenticating jpeg visual content. In *Electronic Imaging. International Society for Optics and Photonics, pages=140—151, year=2000,.*

[LL03] Chun-Shien Lu and H-Ym Liao. Structural digital signature for image authentication: an incidental distortion resistant scheme. *IEEE Transactions on Multimedia*, 5(2):161–173, 2003.

[LWDD15] Swee-Won Lo, Zhuo Wei, Robert H. Deng, and Xuhua Ding. On security of Content-Based video stream authentication. In *ESORICS 2015*, pages 366–383, 2015.

[Mou01] Pierre Moulin. The role of information theory in watermarking and its application to image watermarking. *Signal Processing*, 81(6):1121–1139, 2001.

[NT16] Assa Naveh and Eran Tromer. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 255–271. IEEE, 2016.

[RD02] Christian Rey and Jean-Luc Dugelay. A survey of watermarking algorithms for image authentication. *EURASIP Journal on Advances in Signal Processing*, 2002(6):218932, 2002.

[SSY02] R. Sun, H. Sun, and T. Yao. A svd-and quantization based semi-fragile watermarking technique for image authentication. In *International Conference on Signal Processing*, pages 1592—-1595. IEEE, 2002.

[VKJM00] Ramarathnam Venkatesan, S. M. Koon, Mariusz H. Jakubowski, and Pierre Moulin. Robust image hashing. In *Image Processing, 2000*, volume 3, pages 664–666. IEEE, 2000.