

# Don't See Me, Just Edit Me: Toward Secure Cloud-based Video Editing

Anonymous ACM Multimedia Submission

## ABSTRACT

Imagine that you want to store and edit a video in the cloud, but you don't want to let cloud know about the content. Currently there is not good solution to this problem. In this paper we present a system that allows users to encrypt their video files using Shamir's secret sharing scheme, upload them to the cloud where they can perform cropping and bicubic scaling to achieve zooming. This system in its preliminary stage, though has some limitations, provides perfect security and sets up for further research to explore the possibility of supporting other video editing operations over cloud in the encrypted domain.

## Categories and Subject Descriptors

K.6.5 [Security and Protection]

## General Terms

Security

## Keywords

Cloud computing, Video zooming, Shamir's secret sharing, Secure processing

## 1. INTRODUCTION

Cloud computing is rapidly becoming ubiquitous. Users are moving their data to cloud data centers (CDCs) for storage, and businesses are using cloud resources to do their processing. Computationally expensive operations are ideally suited for cloud computing, and as such, off-loading heavy-lifting video editing operations to CDCs is a cost-effective alternative to local processing.

Moving data to the cloud has many benefits but it comes at the cost of security and privacy. A lot of data being stored and processed by cloud services is in plaintext, which can be intercepted by attackers. Several recent data breaches indicate that users data is not always secure when stored

by a third party. Companies could still be the victims of a breach and the attacks could steal or tamper with the data. Additionally, there could be insider attacks at the CDCs.

These insecurities become increasingly significant when the video data in question is of a sensitive nature. With a huge amount of video data, emerging from different applications such as camera surveillance and smart phones, and being outsourced to cloud, it is important that video processing operations performed by cloud services are secure. Our focus in this paper is on video editing operations such as cropping and scaling. While traditional schemes provide computational security, those that lack homomorphic properties forces users to decrypt the video before processing it, thus exposing the data to increased risk.

The challenge then becomes to construct a method that allows users to encrypt their video before uploading it to cloud services where it can be safely manipulated in encrypted form. By utilizing a homomorphic cryptographic scheme we can achieve this goal. Using this scheme also allows us to perform basic arithmetic operations on the ciphertext, and we can still retrieve the desired result. Thus, as long as video editing operations can be performed without comparisons and only using the four fundamental arithmetic operations (addition, subtraction, multiplication and division), we can reliably edit videos in an encrypted domain if the following holds  $E(v_1 \circ v_2) = E(v_1) \circ E(v_2)$ , where  $v_1$  and  $v_2$  represent the two data units in a video (e.g. two frames or two pixels in a frame),  $E$  denotes the encryption function, and  $\circ$  represents the one of the four fundamental operations.

In this paper, we propose a solution to this problem by using Shamir's Secret Sharing (SSS) scheme. For each frame in a video, we process it byte-by-byte and encrypt each byte, thus creating shares, or shadow copies, of the video. Our solution allows for cropping and scaling on the encrypted shadow copies to achieve zooming. To the best of our knowledge, this is the first paper which performs cropping and scaling of a video in encrypted domain. We achieve bicubic scaling by way of the additive and multiplicative properties of homomorphism [2]. Zooming into a frame is done by first cropping the frame and then upscaling it.

## 2. BACKGROUND AND RELATED WORK

### 2.1 SSS scheme

The  $(k, n)$  SSS scheme,  $2 \leq k \leq n$ , was proposed by Adi Shamir in 1979 [8], which divides a secret into  $n$  shares such that: i) any  $k$  or more shares can reconstruct the secret, and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

ii)  $k - 1$  or fewer shares cannot reconstruct the secret.

To share a secret  $a_0$  among  $n$  participants, a polynomial function  $F(x)$  is constructed of degree  $k - 1$  using  $k - 1$  random coefficients  $a_1, a_2 \dots a_{k-1}$  in a finite field  $GF(p)$  where  $p$  is prime number  $> a_i, 0 \leq i \leq k - 1$ .

$$F(x) = \sum_{i=0}^n a_i x^i \mod p \quad (1)$$

Any  $k$  out of  $n$  shares can reconstruct the secret using Lagrange interpolation; the secret can be obtained at  $L(0)$  i.e.  $L(0) = a_0$ , as:

$$L(x) = \sum_{j=1}^k y_j \prod_{i=1, i \neq j}^k \frac{x - x_i}{x_j - x_i} \mod p \quad (2)$$

The SSS scheme has been used to protect different types of data such as text, image and video [1]. There have been efforts to limit the storage requirements using a variation of the SSS scheme, called Ramp Secret Sharing. While these schemes reduce how much space is needed to store the shares, the scheme loses its information theoretic security property, thus making it vulnerable to attacks.

There have been a few attempts to process image data in encrypted domain using SSS's homomorphic property. Examples include: Mohanty et al. [6] for image scaling and cropping, Lathey and Atrey [3]. This work is an extension of [6]. While in [6], authors performed scaling and cropping on encrypted images, this paper goes a step ahead and attempts to perform the same operations on encrypted videos.

## 2.2 Encrypted Domain Video Processing

From the perspective of encrypted domain video processing, there are a few works. For instance, Upmanyu et al. [9] proposed secure solutions for applications such as blind authentication, i.e. blindly authenticating a remote-user using his biometric, object tracking, and face detection over cloud. They presented a secure framework for carrying out visual surveillance on random looking video streams at remote servers. Similarly, Avidan and Butman [7] presented a method to securely perform face detection without leaking vital information about the image. Lu et al. [4] discussed various applications of video processing in the encrypted domain, the challenges and potential techniques, focusing on video search, classification and summarization. Despite these works, video scaling and cropping has not been done earlier, which we do in this paper.

## 2.3 Bicubic Scaling

For scaling of video frames, we used the bicubic interpolation method [5], which takes 16 reference points and interpolate them. This scaling algorithm generally works well for upscaling images while still maintaining decent levels of detail. Mathematically, it is expressed as:

$$P(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j \quad (3)$$

Here,  $x$  and  $y$  denote the position of the interpolated value,  $a_{ij}$  is the value of the  $ij^{th}$  sub pixel in the targeted image, and  $P(x, y)$  is the interpolated pixel value.

## 3. PROPOSED WORK

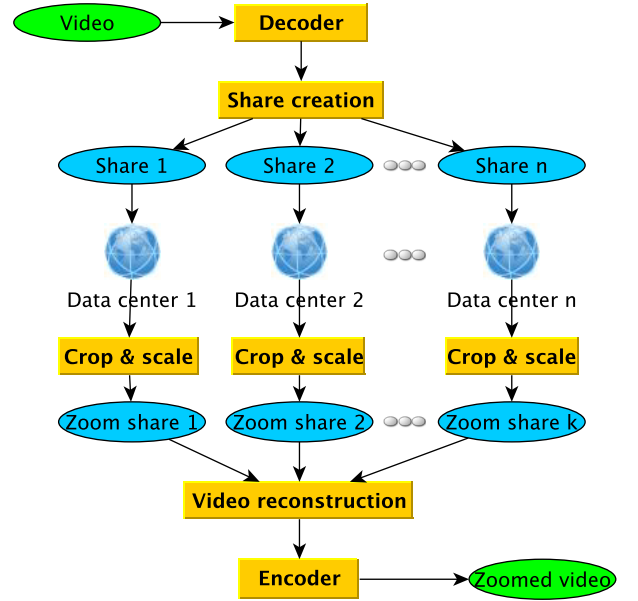


Figure 1: System architecture

## 3.1 Threat Model and Assumptions

Cloud-based video editing give host to a slew of security threats. The following entities are involved in the process: the owner of the video, video transmission channels, CDCs, users of the video. Here transmission channels and CDCs are assumed to be vulnerable to attacks and they are considered untrustworthy, while the owner and the user of the video can have full access to the video, and hence, are regarded as trustworthy.

With these vulnerabilities we need a solution that assumes the data is at risk as soon as it is not in the possession of the user. To achieve the best security possible, all CDCs should be unrelated and any  $(k \leq n)$  must not collude. We assume that both the input and output will be in the mpeg compression format. For convenience and to maintain data integrity, all video editing operations should be nondestructive so not to change the input file. For simplicity we further assume that all operations will be performed on the entire video and not for selected frames.

## 3.2 Architecture

To not expose the secret video to unnecessary risks, all encryption and decryption should happen on the user's end. The CDCs should not possess the capability to do any of these operations. We address this issue with the architecture illustrated in Fig. 1. Here, the CDCs can only crop and scale the shadow copies while the user can create shares and reconstruct the original.

The secret video is inputted into the decoder, which gives us access to all the frames in their presentation order. Decoding is done by the FFmpeg decoder. With each frame, we input the frame, byte-by-byte to the SSS algorithm to create the shares. Once all the frames have been processed, and all shares created, the shadow copies are uploaded to the CDCs and the original secret video is deleted.

With shadow copies stored in CDCs they can now be pro-

cessed. This is where shadow copies will be cropped and scaled. Shares are cropped and scaled individually. To reconstruct the secret video, edited or otherwise,  $k$  shares are downloaded and reconstructed using  $L(0)$  where  $x_i$  is the number of that particular share and  $y_i$  is the value of the share being used in reconstruction. Reconstruction leads to either the original secret being recovered, or the result of cropping and scaling of the shadow copies, but in plaintext.

### 3.3 System Implementation

We utilized FFmpeg’s libav library for decoding and encoding. This also gave us access to the frame data, which we could then encrypt. We originally intended to use libav to re-encode each shadow copy into the same format as the secret video, but this proved difficult to achieve with codecs that use lossy compression techniques. As a result, by re-encoding a share in the same format as the original video we lose information thus making reconstruction impossible. We could not reliably re-encode shares using lossy codecs so we opted to develop out own file format to store each share.

Our file format is reflects our naïve encryption of the video where all I, P and B frames are stored completely, and in presentation order. This is a less than ideal solution and gives cause for future improvements. As a result, we store a lot of redundant information for each frame, and as such the space required for each share is dramatically higher than that of the original video because we do not employ any compression techniques.

Furthermore we chose to store the header information in plaintext, while all the frame data is stored as ciphertext. The headers contain important information that it is needed to properly reconstruct the video. Even with headers in plaintext, the content of the video is still secure. An attacker does not gain any insight into the content of the secret video by knowing the headers. Our headers store timestamps, the resolution of the video, and the size of each frame in bytes.

Our system consists of three components, the video source, the  $n$  shareholders and the user, as illustrated by Fig. 1. A user would first create shares of the secret video before distributing all  $n$  ( $n = 5$  in our case) shadow copies to its shareholders. Then cropping and scaling is done on all  $n$  shares. Any  $k$  ( $k = 3$  in our case) shareholders bring their shadow copies together to reconstruct the video.

After decoding the input video, we process all P, B and I frames in presentation order. All frames contain Y, Cb and Cr components with 4:2:0 sub-sampling. Each component is processed individually and independently to each other. The FFmpeg decoder interleaves the components to display the frame. To create all  $n$  shares, the system computes  $F_b(x)$  for each byte  $b$  in the frame and for  $n$  different values of  $x$ .

When all three components are processed, we will have the fully generated shadow copies for the frame. Each shadow frame is now written to their respective shares, where all frames are stored in presentation order. This simplifies the reconstruction phase.

Once the shadow videos have been created, they are uploaded to different CDCs, and the original video is deleted. The CDCs do not possess the means to reconstruct the video, nor analyze the contents of videos.

To achieve zooming in the video, the user submits the dimensions for the Region of Interest (RoI) along with the starting coordinate of the region. The system removes all data outside the RoI for all frames in the shares. When the

frame is fully cropped, the system scales the cropped frame to the dimensions of the original frame. This is done by calculating  $P(x, y)$  for each sub-pixel in the output frame. Note that all computations are performed under modulo  $p$ . Now that the RoI has been scaled, the frame is written to a new file that stores the zoomed shadow copy.

To reconstruct the original video,  $k$  shareholders input their shares to the system. To find the original secret video, the  $i^{th}$  byte of  $k$  shadow copies are used in Lagrange interpolation to create the polynomial  $L(x)$  where  $x = 0$  and byte  $b_i = y_i$  and  $x_j$  is the number indicating the share’s id.  $L(x)$  is now the reconstructed  $i^{th}$  subpixel. After reconstructing a frame, it is re-encoded using FFmpeg’s encoding functionality. With all frames reconstructed the original video has been retrieved.

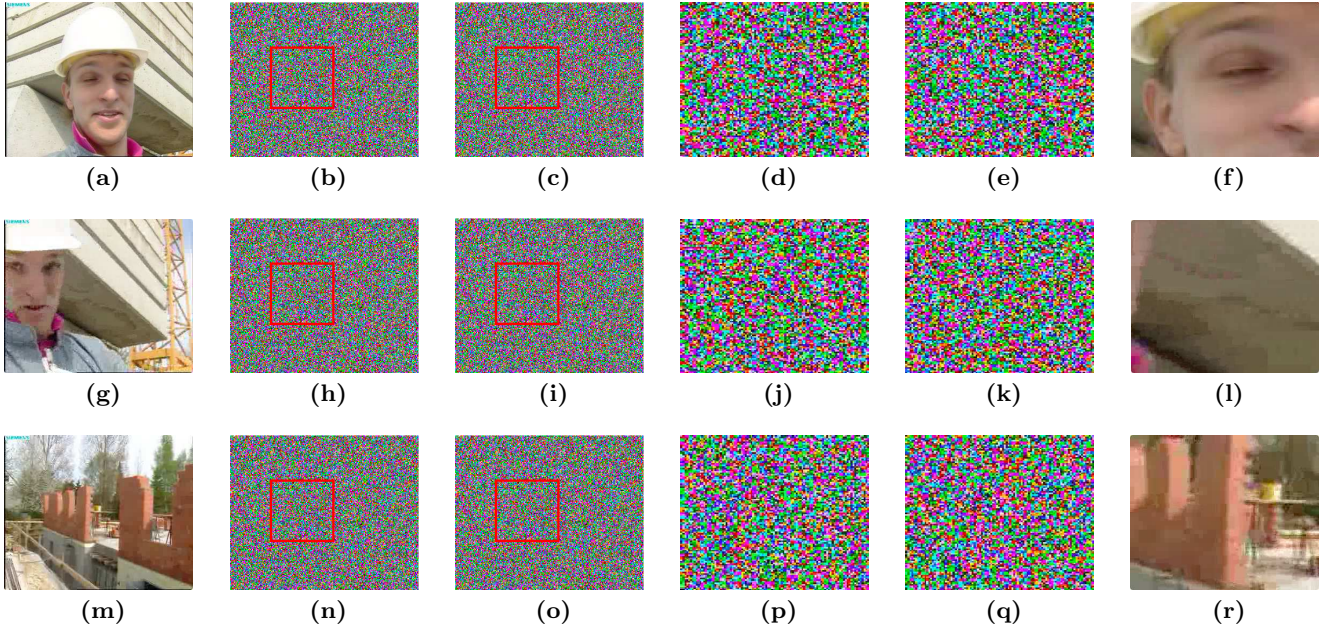
## 4. ANALYSIS

In our testing our secret video was the Foreman video (size: 436 KB, dimensions:  $352 \times 288$ ). As can be seen in Fig. 2, our solution maintains visual security of the video. An attacker cannot infer the content of the video by viewing the shadow copy. Furthermore, because our system uses the SSS scheme, along with naïve video processing, we know that our system is information theoretically secure. Assuming that an adversary does not have access to  $k$  shadow copies, the adversary cannot infer the value of a specific subpixel in a frame. In our system, if an attacker gains access to 2 shares, for any particular value there are  $p$  possible values for the secret byte in the finite field  $\mathbb{Z}_p$ .

An additional benefit of using SSS is that it is resistant to data tampering. SSS provides data integrity by making reconstruction of a secret impossible if the shadow value does not coincide with the original polynomial. Thus if the adversary modifies a share, the polynomial being reconstructed will be incorrect, resulting in an unintelligible video.

The proposed system, though provides perfect security, suffers from some limitations in its design. By storing video shares in their entirety the storage requirements of our system is rather strict. the shadow copies of the Foreman take up 47 MB. The storage requirements of the shadow copies depend on the resolution of the video as well as its length. Thus, long high resolution videos will require large amounts of free space. In our tests, a 01:58 minute video with  $720 \times 480$  resolution require only 90 MB of storage in its compressed form, but approximately 2 GB for each of its shadow copies. The exact compressed-to-share storage ratio is difficult to pinpoint as it is a function of the video resolution and length.

Further limitations come from share generation, cropping, scaling and reconstruction. Because each byte in the video data is considered a secret, any operation must be done on each individual byte. This means that there is a lot of processing overhead, especially with share creation, bicubic interpolation and Lagrange interpolation. If only cropping is requested by the user, the time to reconstruct the secret using Lagrange interpolation can be significantly reduced due to the smaller video resolution. There are fewer secrets in each frame to reconstruct. Table 1 shows the run time of our system using the Foreman video. Share creation is relatively quick and is largely bound by decoding the input video. Reconstruction can take a lot longer because of Lagrange interpolation where we have to calculate  $\frac{x-x_i}{x_j-x_i} \equiv (x-x_i)(x_j-x_i)^{-1} = 1 \pmod{p}$  which can be com-



**Figure 2:** Cropping and scaling of the Foreman where (a) , (g) and (m) are the 1st, 190th and 300th frames, respectively, from the original video. (b), (c), (h), (i), (n) and (o) are the 1st and 2nd unedited shares for their respective frame. The red rectangle in these share frames shows the region chosen for cropping and scaling. (d), (e), (j), (k), (p) and (q) are the 1st and 2nd cropped and scaled shadow copy for their respective frame (f), (l) and (r) show the reconstructed frames.

**Table 1:** Processing time for Foreman video in seconds

	Average	Minimum	Maximum
Encryption	6.86	6.03	7.83
Reconstruction	31.48	5.42	57.53

putationally expensive.

## 5. CONCLUSION AND FUTURE WORK

As previously stated, our implementation of the encrypted videos uses a naïve representation of video data. This leads to increased storage requirements, and as such, a form of compression should be implemented. Video features such as Discrete Cosine Transform coefficients, motion vectors, inter frames and macroblocks would gain significant storage improvements. The benefit would be two-fold: i) reducing storage overhead, and ii) reducing processing overhead.

Implementing macroblocks for encrypted videos allow for additional benefits by making the video better suited to be streamed to a user. Thus the shadow copies can be streamed to a user and cropping and scaling can be done interactively. Also, in future, we would attempt to identify a more efficient way of storing the encrypted data while still maintaining its security.

## 6. REFERENCES

- [1] S. Alharthi, P. K. Atrey, and M. S. Kankanhalli. Secret video sharing. In *Proc. of the APSIPA annual summit and conference*, 2010.
- [2] J. C. Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract).
- [3] In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO’86*, number 263 in Lecture Notes in Computer Science, pages 251–260. Springer Berlin Heidelberg, 1987.
- [4] A. Lathey and P. K. Atrey. Image enhancement in encrypted domain over cloud. *ACM Transactions on Multimedia Computing, Communications and Applications*, 11(3), 2015.
- [5] W. Lu, A. Varna, and M. Wu. Secure video processing: Problems and challenges. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5856–5859, May 2011.
- [6] D. P. Mitchell and A. N. Netravali. Reconstruction Filters in Computer-graphics. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’88*, pages 221–228, New York, NY, USA, 1988. ACM.
- [7] M. Mohanty, W. T. Ooi, and P. Atrey. Scale me, crop me, know me not: Supporting scaling and cropping in secret image sharing. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2013.
- [8] B. Schölkopf, J. Platt, and T. Hofmann. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. MIT Press, 2007.
- [9] A. Shamir. How to share a secret. In: *Communications of the ACM*, 22:612–613, 1979.
- [10] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. V. Jawahar. Efficient privacy preserving video surveillance. In *Proceedings of IEEE 12<sup>th</sup> International Conference on Computer Vision*, pages 1639–1646, 2009.