

Now You See Me, Now You Don't: Secure Cloud-based Video Editing

Anonymous ACM Multimedia 2014 Submission

ABSTRACT

Personality prediction has been widely studied in various fields including psychology and social media. The emergence of online social networks provides novel opportunities to obtain information about user behaviors, demographics and administer standardized surveys for studying such social and psychological phenomena in a data-driven manner. Using the data generated from a personality survey app on Facebook used by 228,343 users, this work undertakes a statistical validation of a frequently posited hypothesis: *does a person's personality have any relation with their demographic information such as gender and date of birth (DOB)*. Also, vast literature on astrology - though not yet scientifically proven - suggests that the one's personality is related to their sun sign which is derived from DOB. Based on DOB we have categorized all users into 12 groups referred as 12 sun sign and examined the relationship between users' big five personality traits and their sun sign through chi-square (χ^2) test. Subsequently, the ordinal regression model is used to model this relationship. Our results suggest that there are few personality traits which can relate to DOB, and it can be a very quick indicator of one's personality where there exists a relationship.

Categories and Subject Descriptors

J.4 [Social and Behavioral Sciences]: Miscellaneous

General Terms

Human Factor

Keywords

Personality prediction, Date of birth, Sun sign, Social media

1. INTRODUCTION

Cloud computing is rapidly becoming ubiquitous. Users are moving their data to cloud data centers 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 cloud data centers 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 as the data is being transported to and from cloud data centers. Several recent data breaches indicate that users data is not always secure even 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.

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 editing operations performed by cloud services are secure. While traditional schemes provide computationally secure data, 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. 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 correct 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.

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

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.

ACM Multimedia '14 Orlando, FL, USA

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

[?]. Zooming into a frame is done by first cropping the frame and then upscaling it.

The rest of this paper is outlined as follows: Section 2 presents prior work on this topic. Section 3 describes or proposed work along with its assumptions, with Section ?? presenting our security analysis and our results. Future improvements are discussed in Section ?? and our conclusion in Section ??

2. RELATED WORK

Khiem et al. [?] proposed two different methods of streaming video data that allowed for dynamic zooming. By dividing a frame into macroblocks of 16x16 pixels, the macroblocks needed to display the Region of Interest — RoI — can be requested and the server will only decode the necessary macroblocks and thus achieve zooming. Tiled streaming was achieved by pre-selecting grid tiles that are encoded separately, which would be requested as the RoI changes. Then the excess data would be removed during decoding to result in a zoomed frame. Monolithic streams on the other hand use dependency graphs to calculate the necessary macroblocks to decode the requested RoI.

Shamir’s Secret Sharing algorithm creates polynomials of degree $n - 1$ such we need n points to reconstruct the polynomial $F(x)$.

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

where a_0 is the secret we want to obscure, and any other a is a random number such that $a_i < p$. x is the number of the share being created, much like an identifier, such that $x_i < p$ and p is a prime number.

To reconstruct the secret value we use Lagrange Interpolation, $L(x)$, to find the points in $L(x)$ that can recreate the polynomial $F(x)$.

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

$L(x)$ where $x = 0$, x_i is the share value from $F(x)$ and y_j is the number of the applied shares, their identifier, will reveal the secret when the polynomial is of degree $n = k$.

Mohanty et al. [?] demonstrated how cropping and scaling can be performed on encrypted image data using Ramp Secret Sharing. This was done by creating shadow copies of the image and then performing the cropping and scaling operations on the shares. Reconstructing these shares then resulted in an image that had been zoomed in at the specified RoI.

There have been many applications of Secret Sharing. Examples including encryption of images etc. There have been efforts to limit the storage requirements of Shamir’s Secret Sharing, i.e. Ramp Secret Sharing. While these schemes reduce how much space is needed to store the shares, the scheme loses its information theoretically secure property, thus making it vulnerable to attacks. Alharthy et al. [?] showed how Shamir’s Secret Sharing scheme could be applied to video while attaining perfect secrecy. The proposed work is an extension of Mohanty and Alharthy work.

Lu et al. [?] discussed various applications of video processing in the encrypted domain, the challenges and potential techniques, focusing on video search, classification and

Figure 1: System architecture

summarization. Brown et al. demonstrated how to automatically detect different events in a video stream. Saini showed that image forgery is more reliably implemented in hardware, not in software. Similarly, Saini et al [?] found that image enhancement can be implemented better using VHDL for in-chip image processing. Bitouk et al. [?] proposed a system to replace faces in photographs. Avidan and Butman [?] presented a method to securely perform face detection without leaking vital information about the image. Chen et al. [?] proposed a method to securely calculate linear equations in cloud data centers. Newton et al. [?] presented an algorithm, k-same, to de-identify faces in images, thus rendering face detection impossible. Defaux and Ebrahimi [?] showed how a region of interest could be scrambled to hide private information.

3. PROPOSED WORK

3.1 Threat Model

Cloud-based video editing give host to a slew of security threats. If the video is being sent in clear text, it can be intercepted and collected by an adversary during the upload. An adversary can break into the cloud data center and steal the data or modify it. Disgruntled system administrators pose additional security risks. Encrypting the video prevents an attacker from stealing or modifying it, but in order to facilitate editing operations the cloud data center must either have access to a secret key, or alternatively the operations can be carried out at the cost of increased computation, which often proves unfeasible in real applications. Of course, providing cloud data centers with keys still means that unreliable system administrators are still a threat. Devising a method to efficiently perform operations on encrypted video data would combat all of the issues mentioned above.

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. As such we assume that both data transmission and cloud data centers are vulnerable to attacks. To achieve the best security possible, all n data centers should be unrelated. 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 non-destructive 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

I show the architecture of our system. The secret video is inputted into the decoder, which gives us access to all the frames in presentation order. Decoding is done by the FFmpeg decoder. With each frame, we input the frame, byte-by-byte to the Secret Sharing algorithm to create our