Enabling Encrypted Cloud Media Center with Secure Deduplication

Yifeng Zheng[†,*], Xingliang Yuan[†,*], Xinyu Wang[†], Jinghua Jiang[†‡], Cong Wang[†], Xiaolin Gui[‡]
[†]Department of Computer Science, City University of Hong Kong, Hong Kong
[‡]Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China
{yifezheng2, xinywang, congwang}@cityu.edu.hk,{xinglyuan3-c, jjinghua2-c}@my.cityu.edu.hk
xlgui@mail.xjtu.edu.cn

ABSTRACT

Multimedia contents, especially videos, are being exponentially generated today. Due to the limited local storage, people are willing to store the videos at the remote cloud media center for its low cost and scalable storage. However, videos may have to be encrypted before outsourcing for privacy concerns. For practical purposes, the cloud media center should also provide the deduplication functionality to eliminate the storage and bandwidth redundancy, and adaptively disseminate videos to heterogeneous networks and different devices to ensure the quality of service. In light of the observations, we present a secure architecture enabling the encrypted cloud media center. It builds on top of latest advancements on secure deduplication and video coding techniques, with fully functional system implementations on encrypted video deduplication and adaptive video dissemination services. Specifically, to support efficient adaptive dissemination, we utilize the scalable video coding (SVC) techniques and propose a tailored layer-level secure deduplication strategy to be compatible with the internal structure of SVC. Accordingly, we adopt a structure-compatible encryption mechanism and optimize the way how encrypted SVC videos are stored for fast retrieval and efficient dissemination. We thoroughly analyze the security strength of our system design with strong video protection. Furthermore, we give a prototype implementation with encrypted end-to-end deployment on Amazon cloud platform. Extensive experiments demonstrate the practicality of our system.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Information Storage and Retrieval—*Online Information Services*; H.5.1 [**Information Systems**]: Information Interfaces and Representation—*Multimedia Information Systems*; E.3 [**Data**]: Data Encryption

*The first two authors contributed equally to this work.

ASIA CCS'15, April 14–17, 2015, Singapore, Singapore.

http://dx.doi.org/10.1145/2714576.2714628.

Keywords

Cloud Media Center; Secure Deduplication; Scalable Video Coding; Layer-level Deduplication

1. INTRODUCTION

With the explosive growth of multimedia technology and mobile devices with high-definition cameras, multimedia contents, especially videos, have already dominated the network traffic and demanded a great amount of hardware storage [8]. To handle such a rapidly growing trend, many existing and emerging applications based on videos are deployed at public clouds for its well-known advantages, e.g., availability, scalability, and economy [26]. However, the user privacy could be violated if content-sensitive videos are not protected properly in such an outsourcing environment [23]. In fact, current cloud-based data hosting services are shown to be vulnerable to security breaches. Data disclosure occurs frequently in recent years [15]. Therefore, addressing the privacy concerns becomes significant for building a cloud media center.

A plausible approach is to require each user to encrypt the videos using her secret key before sending them to public clouds. As long as the user's secret key is protected, video confidentiality can be guaranteed. But this approach prevents the cloud media center from supporting deduplication, a crucial function that can greatly save the network bandwidth and eliminate the storage redundancy in cloud services [14]. Identical videos encrypted by different users' secret keys would lead to different ciphertexts, making deduplication infeasible.

To support secure deduplication over encrypted videos, one might consider utilizing convergent encryption (CE) [10], which encrypts data with a key deterministically derived from the data itself (e.g., the hash value), so as to ensure the identical data will map to identical ciphertexts. However, CE is vulnerable to off-line brute-force attacks when the target message space is small or the data is predictable [2]. Besides, CE is under a weak security model, which does not consider the bounded leakage setting, where a certain amount of deterministically and efficiently extractable information of the data could be leaked [25]. Although some recent designs have improved the security of CE [2,14,17,25], it remains to be fully explored whether they can directly enable an encrypted cloud media center, which should simultaneously ensure the strong video confidentiality, provide immediate compatibility with the video structure, and deliver practical video dissemination services.

In practice, an encrypted cloud media center should be able to adaptively disseminate videos to heterogeneous networks and devices such as PCs, Mobile phones, Tablets and SmartTVs [12]. However, such adaptive dissemination might be disabled after encryption. Alternatively, one can ask each user to first generate different versions of videos from the same source, and then send all encrypted versions to the cloud. Yet, this method will incur a great amount of bandwidth and storage overhead, and thus increase the capital of using cloud services. Therefore, how to enable efficient adaptive video dissemination while supporting secure deduplication over encrypted videos, is the challenge we aim to tackle in this paper, for which we bring together techniques from both cryptography and video processing.

We first investigate the applicability of secure deduplication in the encrypted cloud media center and propose a non-trivial design to strongly protect video confidentiality. It takes into consideration the defenses against the adversaries in the bounded leakage setting, and the adversaries launching brute-force attacks over predictable videos, respectively. To support efficient adaptive dissemination, we exploit the scalable video coding (SVC) techniques [19]. SVC utilizes the concept of layers and enables multiple versions of the same video content to be embedded in a single video file, which can significantly improve the storage efficiency and dissemination scalability [24]. To make secure deduplication compatible with the structure of SVC, we propose a layer-level deduplication strategy tailored for SVC videos, enabling suitable deduplication for encrypted SVC videos.

Besides, we investigate the encryption of SVC videos [20, 22], and adopt a structure-compatible encryption mechanism and optimize the way encrypted SVC videos are stored for fast retrieval and efficient dissemination. Our thorough security analysis shows that video confidentiality is strongly protected in the proposed system design. We give a full implementation of an encrypted end-to-end cloud media center. which is deployed at Amazon EC2 instances and leverages Amazon DynamoDB as the video storage backend. Extensive experiments are conducted to justify the practicality of our system design.

Contributions. Our main contributions are listed below:

- We formulate a secure system framework for the encrypted cloud media center. It supports secure deduplication while protecting data confidentiality against malicious users and untrusted cloud. Building on top of latest literature on secure deduplication [2, 25], our framework supports the secure video deduplication in the bounded leakage setting, and defends the off-line brute-force attacks over predictable videos, respectively.

- We leverage the inherent characteristics of SVC videos in our design. We utilize an encryption mechanism which is compatible with the underlying SVC structure, and optimize the way how encrypted SVC videos are stored for fast retrieval. Meanwhile, the security of our system is thoroughly analyzed.

- We implement an end-to-end cloud media center system with roughly 17,000 lines of codes, which is fully deployed at Amazon AWS. We evaluate our system via various performance metrics on SVC videos with sizes up to 500MB. The results demonstrate the practicality of our system.

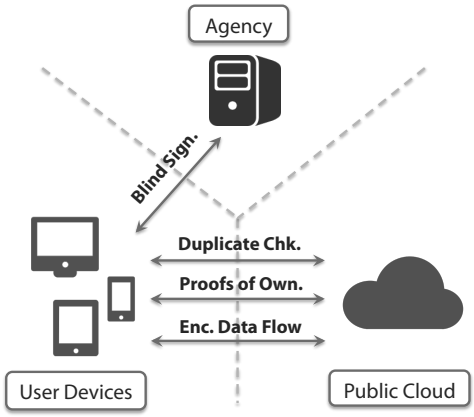

Figure 1: The proposed system framework.

The rest of this paper is organized as follows. Section 2 presents our problem formulation. Section 3 gives some preliminaries. Section 4 formulates the general secure deduplication framework. Section 5 gives the detailed construction for deduplication over encrypted SVC videos under the proposed framework. Section 6 presents the security analysis. Section 7 gives the implementation and performance evaluation. Section 8 describes the related works. Section 9 concludes the whole paper.

2. PROBLEM STATEMENT

2.1 System Model

Our system involves three entities: the cloud media center, the user, and the agency, as illustrated in Figure 1. Their roles are described below:

- The cloud media center (abbr. cloud) provides a video hosting platform, which stores users' encrypted videos and adaptively distributes them to cater for heterogeneous user devices and network bandwidth. It enforces secure client-side deduplication, i.e., duplicate check is performed before users upload videos.

- The user outsources her encrypted videos to the cloud, and possibly deletes the original ones at local. Later, the user may access her own videos. Video sharing is not the focus of this paper, although it can be achieved along with our design through techniques such as the attribute-based access control [23].

- The agency, hosted by a third party (e.g., a video service provider), facilitates our system to safeguard the confidentiality of user videos against off-line brute-force attacks. It assists users and cloud to perform the duplicate check in a controllable fashion, and enables users to perform the encryption that supports deduplication.

2.2 Threat Models

Our goal is to protect the confidentiality of users' videos. We will consider a strong security model for secure deduplication, i.e., the bounded leakage setting as in [25], where a certain amount of deterministically and efficiently extractable information about videos could be leaked. We will also be

concerned with protection for both predictable and unpredictable videos.

Two types of adversaries are considered in our system: (1) *Malicious outside adversary.* The outside adversary may refer to a user who might obtain some knowledge (e.g., a hash value) of a video and try to earn the ownership of the target video from cloud. We assume that she will not upload a fake video to compromise the integrity of other users' videos. (2) *Honest-but-curious inside adversary.* The inside adversary may refer to cloud or the agency. On the one hand, cloud faithfully follows the designated deduplication scheme, yet intends to infer users' encrypted video contents. It might also manipulate one user or a number of users to harvest target video contents. On the other hand, the third-party agency dutifully executes the assigned functions, but also tries to extract useful information about users' videos. In this paper, we do not consider that cloud modifies or deletes users' videos. And we assume that there is no collusion between cloud and the agency.

3. PRELIMINARIES

3.1 Oblivious Pseudorandom Function

An oblivious pseudorandom function (OPRF) protocol enables two parties, say sender S and receiver R, to jointly and securely compute a pseudo-random function (PRF) $f_{sk}(x)$, where sk is the secret key of S and the input x is contributed by R. The OPRF protocol enforces that R only learns the output value $f_{sk}(x)$ and S learns nothing from the interactive process [11]. Verifiable OPRF schemes allow R to further verify whether the result $f_{sk}(\cdot)$ is correctly computed under S's secret key sk, via utilizing S's corresponding public key pk. Deterministic blind signatures can be used to build verifiable OPRF schemes [7]. In a blind signature scheme, a user is able to obtain and verify the digital signature of a message from a signer without revealing any data information. Meanwhile, the user cannot learn the signer's secret key. In this paper, we use the RSA-OPRF scheme built on RSA blind signatures [4] as a building block for our proposed system.

3.2 Scalable Video Coding

The SVC technique utilizes the concept of layers and enables multiple versions of the source video content to be embedded in a single file, i.e., a SVC video [19]. Intuitively, a SVC video is composed of a base layer, which represents the basic visual experience, and enhancement layers, which can improve the video by supplementing the base layer in different scalability dimensions (i.e., time, quality, resolution). In our system, we are particularly interested in the resolution scalability as the first instantiation. Through employing various inter-layer prediction methods, SVC removes the redundancy between different representations of the same video content [18]. It is noted that at the decoder side, a lower layer must be present if a higher layer exists but not the other way around. In other words, if the layers of a SVC video are discarded from the highest layer, the rest of layers are still decodable. Thus, a user can adaptively enjoy different versions of the same video content through a single SVC video.

Therefore, from the perspective of storage efficiency and dissemination scalability, it is advantageous to store SVC videos in the cloud media center to cater for heterogeneous network bandwidth and different devices. Without loss of generality, a SVC video SV with n layers can be denoted as $SV = (m_1, m_2, \ldots, m_n)$, where m_1 is the base layer and m_i is the $(i-1)th$ enhancement layer for $i \in [2, n]$ [23].

4. PROPOSED SECURE DEDUPLICATION FRAMEWORK

In this section, we present our secure deduplication framework that can protect users' videos in the bounded leakage setting, and defend off-line brute-force attacks over predictable videos, respectively. We start with describing our design intuition to address the threats mentioned in Section 2.2, and then elaborate on the detailed construction. We note that the framework is suitable for generic data, e.g., textual files and images.

4.1 Design Intuition

Our system targets secure client-side deduplication over encrypted videos. First, we consider secure client-side deduplication in a strong security model, i.e., the bounded leakage setting first proposed by Xu et al. in [25], where a certain amount of deterministically and efficiently extractable information of the plaintext data could be leaked. Under this setting, CE is not appropriate for use in our system since its data encryption key is not leakage resilient. Specifically, the private key in CE is derived from the data plaintext in a deterministic way and could be leaked *before* the encryption process. For similar reasons, using the plaintext hash as a proxy for the ownership of data can also be insecure under the leakage setting.

To address the threat posed by the bounded leakage setting, the following treatment inspired by [25] could be adopted. Firstly, deduplication is achieved by using the hash value $H(V)$ of the video V for duplicate check. Secondly, the video encryption key τ is selected randomly by the initial uploader, who also creates a one-time message-derived mask via a keyed hash function, i.e., $F_s(\cdot)$, to hide it, where s is a random string. Note that because τ is randomly generated, even the hash value of V is possibly leaked, the video is still protected as long as τ is hidden from cloud. Moreover, this type of masking can enable all the subsequent users who own the same video copies to obtain the random key τ and further prove to cloud that they indeed own the videos, by running a proofs-of-ownership (PoW) protocol with cloud.

Although the above treatment is resistant to the bounded leakage setting, it is not directly suitable for use in our system to provide strong protection for video confidentiality. In particular, the above treatment is vulnerable to off-line brute-force attacks over predictable videos. This vulnerability originates from the fact that the hash value is directly exposed to cloud for duplicate check and the random encryption key τ is only protected by V, which will be analyzed in detail shortly. In order to get rid of this vulnerability while still maintaining the security strength of the above treatment in the bounded leakage setting, we resort to an agency for assistance in our system, inspired by [2]. Specifically, we leverage the agency to securely produce message-derived tag α and label β via a RSA-OPRF protocol similar to [2]. The message-derived tag α is used for secure duplicate check and prevents cloud from directly accessing the video hash, while the message-derived label β is embedded in the mask during the initial upload and assists the recovery of the ran-

dom key τ during the subsequent upload. We note that through this careful enhancement design, our system can also effectively prevent cloud from mounting off-line brute-force attacks over predictable videos and thus provide strong protection for video confidentiality, which will be analyzed in detail in Section 6.

4.2 Secure Deduplication Framework

We are now ready to present the detailed construction of the proposed secure deduplication framework. The work flow is illustrated in Figure 2. As shown, before uploading a video V, the user first needs to engage in the RSA-OPRF protocol with the agency to derive the message-derived tag α and label β of V, i.e., $\alpha = \mathsf{OPRF}_{k_1}^{RSA}(H(V))$ and $\beta = \mathsf{OPRF}_{k_2}^{RSA}(H(V))$, where k_1 and k_2 are two different secret keys of the agency for signing (we refer readers to Appendix for the details of the RSA-OPRF protocol). Then α is submitted to cloud for duplicate check. If it does not exist at the cloud side, the user is considered as the initial uploader of V; otherwise, the user is considered as the subsequent uploader.

Suppose user u is the initial one who uploads the video V after duplicate check. User u first encrypts V with a random key τ and produces the video ciphertext C_V. Then u generates a masked key r via using a keyed hash function $F(s, V \parallel \beta)$ to protect τ, where s is a random string, which will enable all subsequent users who indeed owns V to recover τ for correct decryption. Besides, τ is encrypted under u's private key sk to produce the ciphertext C_τ. Finally, u sends $\{C_V, s, r, C_\tau\}$ to cloud, and $H(C_V)$ is computed by cloud for the later use of PoW protocol. Suppose u' is a subsequent user who tries to upload V after the duplicate check. User u' has to run a PoW protocol with cloud to prove that she indeed owns V. Specifically, u' will request (r, s) from cloud, and utilizes V and β to recover the correct encryption key τ. Then u' encrypts V with τ to produce the ciphertext C_V and also computes $H(C_V)$. After that, $H(C_V)$ is sent to cloud for equality checking. After verification, u' is admitted as the owner of C_V. Finally, u' uses her private key sk' to encrypt the recovered τ and stores it at the cloud side. During the retrieval process, the user first downloads C_V and C_τ from cloud, and then uses its private key sk to decipher C_τ and further decrypt C_V with the recovered τ.

Differences compared with prior works. In the proposed deduplication framework, the message-derived tag α is used for duplicate check, and the random key τ is protected by a mask derived from the video V and the message-derived label β. Note that in [25], the hash value of data is directly exposed to cloud for duplicate check and the random key is only protected by the data. Therefore, if this approach is adopted for secure client-side deduplication, cloud can launch off-line brute-force attacks over predictable videos. Given the ciphertext C_V and knowing that its underlying video V is drawn from a dictionary $D_V = \{V_1, V_2, \ldots, V_n\}$, cloud can launch the following two types of off-line brute-force attacks to recover V. In the first attack, cloud can simply compare the received (stored) hash $H(V)$ with the computed hash of each V_i ($i \in [1, n]$) in D_V, and then recover the target video V when two hashes are found equal. In the second attack, cloud may first use each V_i in D_V to decrypt the masked random key and get a key candidate set $\mu_{set} = \{\mu_1, \mu_2, \ldots, \mu_n\}$. Then for each $i \in [1, n]$, cloud can use μ_i to decrypt the target video ciphertext C_V and compare the decryption result with V_i in the dictionary D_V to find the matched target video.

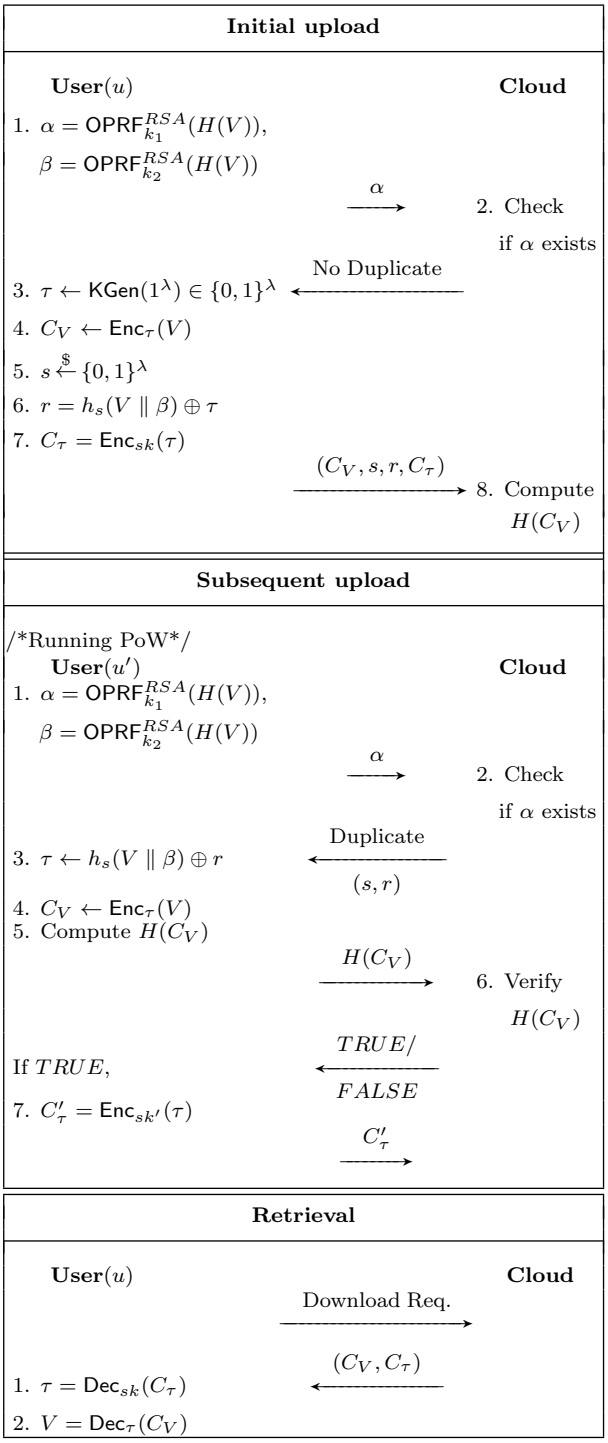

Figure 2: Overview of the system work flow. Here $\mathsf{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a deterministic symmetric encryption scheme with λ bits long key length and $h_K : \{0,1\}^* \to \{0,1\}^\lambda$ is a key-ed hash function. Note that it is omitted that a user runs KGen to derive the private key sk at the system setup.

To securely use *server-side* deduplication and defend offline brute-force attacks over predictable data, Bellare et al. [2] propose an improved design based on CE. They resort to a semi-trusted party named key server for obliviously embedding a secret in the data hash used for encryption and do not consider any data leakage setting. Thus, if directly applied for secure deduplication in the bounded leakage setting, the data encryption key could be leaked as long as the hash is leaked [25]. Different from [2], our system targets client-side deduplication which saves both storage and bandwidth. Meanwhile, our design randomly selects keys for video encryption and protects them by masks derived from both users' videos and the blind signed hashes. Thus, the encryption key is well protected in the bounded leakage setting.

5. STRUCTURE-AWARE DEDUPLICATION OVER ENCRYPTED SVC VIDEOS

In this section, we exploit the internal structure of SVC videos to enforce layer-level deduplication under the proposed secure deduplication framework. We first motivate the importance of layer-level deduplication for SVC videos and then present the detailed construction.

5.1 Layer-Level Deduplication

The traditional file-level deduplication and block-level deduplication are not suitable for SVC videos. This misfit can be demonstrated by an intuitive and practical scenario. Suppose that a user, say u_a, owns the base layer and an enhancement layer for a source content, while another user, say u_b, only owns the base layer for the same source content. And u_a and u_b both used some cloud storage service with deduplication. Then, by file-level deduplication, the two different SVC videos of u_a and u_b can not be deduplicated since they have different contents at the file level. On the other hand, directly splitting a SVC video into blocks and performing block-level deduplication is not a desirable choice since the layers in a SVC video are formatted in a special structure [19,21] and applying block-level deduplication may destroy the structure of SVC video. To address the above challenges, we propose to exploit the layered nature of SVC videos to enforce layer-level deduplication, which treats each layer of a SVC video as a unit for deduplication.

5.2 Construction

Before presenting our construction of secure layer-level deduplication over encrypted SVC videos, we give two important observations that facilitate our design. First, we note that the base layer is the foundation of a SVC video and it also serves as the reference basis for higher enhancement layers [19]. This vital observation indicates that two SVC videos can hardly have duplicate layers if they do not have the same base layer, which inspires us to utilize the base layer for the duplicate check for a given SVC video. Second, users having the same base layer may possess different numbers of enhancement layers for the same source content under their heterogeneous devices and network environments.

Based on these important observations, the main idea for enforcing secure layer-level deduplication over encrypted SVC videos is introduced as follows: Before uploading a SVC video, the message-derived tag α_1 of the base layer and the number of layers are sent to cloud for duplicate check. If a match for α_1 is not found, it is considered that the SVC video does not contain any duplicate layers and thus all layers should be uploaded by the user. Otherwise, a SVC version with the same base layer has already been stored in cloud and thus the user's SVC video may contain a certain number of duplicate layers. In this case, if the user's SVC video has fewer layers than the SVC version in cloud, the PoW protocol needs to be run over all layers of the SVC video. Otherwise, the user only needs to run the PoW protocol over the duplicate layers contained in her SVC video and also uploads the additional layers. In our proposed secure deduplication framework, the construction with layer-level deduplication includes three phases which are described as follows:

Interaction with the agency. Before uploading a SVC video $SV = \{m_1, m_2, \ldots, m_n\}$ to cloud, the user first needs to engage in the RSA-OPRF protocol with the agency to derive the message-derived tag α_1 of the base layer and a message-derived label set $\{\beta_i\}_{1\leq i\leq n}$ for all layers, where $\alpha_1 = \mathsf{OPRF}_{k_1}^{RSA}(H(m_1))$ and $\beta_i = \mathsf{OPRF}_{k_2}^{RSA}(H(m_i))$.

Upload phase. After the interaction with the agency, the user sends α_1 and the number n of layers of SV to cloud for duplicate check. If a match for α_1 is not found, it is considered as the initial upload of SV. Suppose the initial uploader u has L layers (i.e., $n = L$). Then u performs the dedupable encryption as specified in the secure deduplication framework over each layer. For each layer m_i, u generates a layer ciphertext C_i, a masked layer key r_i along with a random string s_i, and a layer key ciphertext C_{τ_i}, where τ_i is the layer key. Then u sends $\{C_i, (s_i, r_i), C_{\tau_i}\}_{1\leq i\leq L}$ to cloud. Cloud will compute $H(C_i)$ over each layer ciphertext C_i for the later use of PoW protocol.

If a match for α_1 is found, it is indicated that a SVC version with the same base layer is already uploaded by some user. Thus, the upload request is considered as the subsequent upload. Suppose the subsequent uploader u' has L' layers (i.e., $n = L'$) and the already stored SVC version in cloud has L_c layers. The subsequent upload proceeds as follows. If $L' \leq L_c$, u' runs the PoW protocol over all her layers to earn the ownership from cloud. Recall that the layer keys $\{\tau_i\}_{1\leq i\leq L'}$ are recovered during the PoW process. Then u' encrypts each τ_i using the private key and stores them in cloud. If $L' > L_c$, for the L_c preceding layers (duplicates) $\{m_i\}_{1\leq i\leq L_c}$ of the SVC video, u' needs to run the PoW protocol over each of them to earn the ownership from cloud, and produces the ciphertexts $\{C'_{\tau_i}\}_{1\leq i\leq L_c}$ of the recovered layer keys $\{\tau_i\}_{1\leq i\leq L_c}$; for each additional layer m_i, where $L_c < i \leq L'$, u' produces a layer ciphertext C_i, a masked layer key r_i along with a random string s_i, and a layer key ciphertext C'_{τ_i}, where τ_i is the layer key. Then u' stores $\{C_i, (s_i, r_i)\}_{L_c<i\leq L'}$ along with $\{C'_{\tau_i}\}_{1\leq i\leq L'}$ in cloud. Note that for the case that $L' > L_c$, cloud updates L_c as L' when u' passes the PoW protocol enforced over all duplicate layers. Besides, if the number of layers actually owned by u' is less than the submitted one, cloud would not update L_c and only marks u' as the owner of the duplicate layers over which she passes the PoW protocol successfully. Therefore, regardless of the number of layers a user submits during a subsequent upload process, a user is marked as the owner of a duplicate layer only when she passes the PoW protocol over it.

Retrieval phase. During the retrieval process, the user u requests the number of layers she wants for a certain SVC video owned by her to cloud. Then the corresponding layer ciphertexts along with the layer key ciphertexts are returned to u. After that, u can use her secret key to recover each layer key and further recovers each layer.

6. SECURITY ANALYSIS

In this section, we give the security analysis of our system design in detail. We will demonstrate that our system addresses the threats from *Malicious outside adversary* and *Honest-but-curious inside adversary* respectively, as mentioned in Section 2.2, and meets the security goals.

Firstly, we consider the security against *Malicious outside adversary*. The outside adversary wants to use the hash $H(V)$ to gain the ownership of the video that does not belong to her from cloud. Note that the adversary with $H(V)$ can generate the message-derived tag α and label β through the agency. Recall that our PoW approach requires each subsequent user to compute the hash of encrypted video. Thus, the adversary should obtain the correct ciphertext, which means that she needs the plaintext V and the correct encryption key τ. Note that τ is protected via $F(s, V||\beta)$. Without V, the adversary cannot derive τ, not to mention the correct ciphertext. Hence, the PoW will fail and cloud will not be fooled to mark the outside adversary as a legitimate duplicate owner.

Secondly, we consider the security against *Honest-but-curious inside adversary*. As stated, the agency could be semi-trusted, which faithfully produces the message-derived tag and label via its private keys, but is interested in $H(V)$. Accordingly, our system adopts the blind signature (RSA-OPRF), so the input $H(V)$ is oblivious to the agency. On the other hand, we assume cloud is also curious but does not collude with the agency. Cloud stores the encrypted video C_V, the tag α for duplicate check, and the masked key $\tau \oplus F(s, V||\beta)$. We note that cloud is not able to obtain the plaintext V if the video is unpredictable. As mentioned, τ is well protected if cloud does not know V or β. Without τ, cloud cannot decrypt C_V.

Finally, our system can effectively prevent cloud from mounting the off-line brute-force attacks over predictable videos. As mentioned in Section 4.2, if cloud knows a relatively small message space (or a dictionary) underlying an encrypted video, it can compute all the message hashes in this dictionary. We note that leveraging the agency can defend off-line brute-force attacks. Firstly, our system resorts to an agency to assist the generation of message-derived tag α for duplicate check rather than directly exposing the hash $H(V)$ to cloud. Without the private key of the agency, cloud cannot produce the α for each candidate in the dictionary. In our system, it is required to interact with the agency to get α for each trial. Secondly, since cloud has the encrypted video C_V and the masked key $\tau \oplus F(s, V||\beta)$ (along with a random string s), it can try to recover the corresponding key and obtain the video plaintext. Likewise, cloud cannot produce the message-derived label β without requesting the agency. For each trial, cloud needs to interact with the agency to get β and unmask the key via xoring a mask derived from β and V. In a word, for predictable videos, our system can prevent the off-line brute-force attacks in a controllable fashion and turn it into online trials against cloud, which can be significantly slowed down by enforcing several proper rate limiting strategies [2].

Note that the proposed construction of secure deduplication over encrypted SVC videos is built under the secure deduplication framework. And the PoW scheme is enforced over each duplicate layer. Only the user who indeed owns the duplicate layers can earn the ownership.

7. EXPERIMENTS

7.1 Implementations

We implement our system prototype with roughly 7,000 lines of c++ code and 10,000 lines of java code. We use Thrift to create network services between entities with cross-language Remote Process Call (RPC)[1]. We use the GMP library [2] and the Openssl library [3] to perform our cryptography, i.e, blind signature, symmetric encryption (AES/CBC-256), and full-domain hash function (SHA-256). We collect videos from two benchmarks: VIRAT [16] and DASH [13]. We encode those videos into SVC videos (totally around 100GB) with the JSVM software[4], and decode them with the Open SVC decoder [6]. Meanwhile, we integrate the Open SVC decoder library to a open source video player MPlayer[5] and re-compile it to play the SVC videos. The implementations of each entity are described as follows:

- User client: it is developed in c++ and can process user's requests and call corresponding services. In the process of generating the message-derived tag α and label β, the client communicates with the agency. We also implement a basic access control mechanism including user login and register operations.
- Agency: it is also developed in c++. Once it receives the blinded input from client, the agency will compute the signature and response it to the client.
- Application server: it is implemented with java and has three functions. Firstly, it handles the user's request of duplicate check and replies with the result of querying on the storage server. Secondly, it verifies the ownership of videos by processing PoW. Lastly, after getting the download request, it verifies the access permission and replies the corresponding SVC videos.
- Storage server: it stores the encrypted SVC videos with tags for duplicate check and ciphertext hashes for PoW, and the user profiles with encrypted private keys and video ownerships. Here, we choose DynamoDB[6] as our storage backend. Note that our application server is running on Amazon EC2, which can efficiently communicate with DynamoDB.

SVC encryption. At a high level, each frame of a SVC video consists of a base layer and several enhancement layers [19]. To encrypt a SVC video, one straightforward method

[1]Apache Thrift: http://thrift.apache.org
[2]The CNU Multiple Precision Arithmetic Library: https://gmplib.org
[3]OpenSSL Project: http://www.openssl.org
[4]Joint Video Team: SVC reference software(jsvm software), 2011.
[5]MPlayer: http://www.mplayerhq.hu/design7/dload.html
[6]Amazon Web Service: http://aws.amazon.com/

(a) 144P (b) 240P (c) 1080P

Figure 3: Different qualities for a given SVC video.

is to perform symmetric encryption for each layer per frame. However, this method is not directly compatible with the underlying SVC structure [20, 22]. From the perspective of the underlying structure, a SVC video bitstream is divided into network abstract layer units (NALUs). Each layer (both base layer and enhancement layers) has its own NALUs, and the NALUs can be identified to the corresponding layers from the attached header information [19]. If the whole layer is encrypted, a user client that requests a video cannot perform timely decryption until the whole ciphertext of one layer is downloaded, which in turn affects the timely decoding for playback, and degrades the user experience and service quality.

Instead of performing encryption at the layer level, we adopt NALU-level encryption inspired from [22], i.e., the payload of each NALU is encrypted individually and the related header information is left in cleartext. As a result, as long as one encrypted NALU is received, the client can perform decryption and then decoding. The processing of each layer can be performed in a pipelined fashion at the user client. For the implementation, the client extracts the layer's NALUs in the same layer for each frame, encrypts them one by one and uploads them to the storage backend.

Storage optimization. The NALUs within the same layer are grouped as a layer block by the storage server in our system. When the client wants to download the video, the storage server needs to fast locate the corresponding layer block and its NALUs. In the SVC standard [19], the NALUs are encoded together by adding the start code prefix, i.e., 0x00000001, between each NALU. Regarding our NALU-based encryption, using the start code to fetch NALUs may not meet the performance requirements for fast retrieval.

To optimize the storage, we use Key-Length-Value (KLV) encoding standard[7] to package the NALU. As a result, the storage server can efficiently distinguish each NALU. Explicitly, NALUs are encoded into Key-Length-Value triplets, where Key identifies the NALU (its frame ID), Length specifies the NALU's length, and Value is the NALU itself. Upon the retrieval of the SVC videos, the storage server can read the NALUs by the help of lengths, and combine them with the same frame ID.

7.2 Evaluation

7.2.1 Visual Experience

Figure 3 shows the different qualities (i.e., resolutions) for a given SVC video after the decryption and decoding at the user client. As shown, our security design does not affect the visual scalability of SVC. The more layers the video has, the higher quality the video is.

[7]BT.1563: Data encoding protocol using key-length-value: http://www.itu.int/rec/R-REC-BT.1563-1-201103-I/en

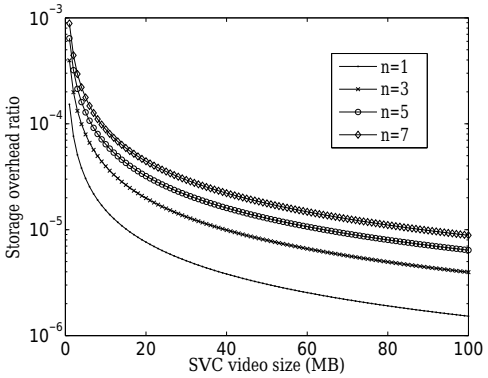

Figure 4: Ratio of the storage overhead to the SVC video size, under different numbers of layers, where n is the number of layers of a SVC video.

7.2.2 Performance

We report our system performance on the aspects of storage overhead, computation costs of different tasks for initial upload and time savings via deduplication. All measurement represents the mean of 10 trials.

Storage overhead. Our design incurs little storage overhead at cloud to support secure deduplication. As we adopt the layer-level deduplication, storing a SVC video with n layers, the storage overhead contains the tag α_1 (32 bytes) for duplicate check, n masked encryption key r with seed s (64 bytes per one) for the access of subsequent users, n owner key ciphertexts (32 bytes per one) and n hashes of the layer ciphertexts (32 bytes per one). Recall that we utilize only the message-derived tag of the base layer for duplicate check, since two SVC videos can hardly have duplicate layers if they do not have the same base layer [21].

In total, the storage overhead for a SVC video with n layers are equal to $32 + 64n + 32n + 32n = 32 + 128n$ bytes, which is roughly in linear to the number of layers and independent of the video size. The relation between the storage overhead and the sizes of SVC videos with different number of layers is displayed in Figure 4. As shown, the storage overhead is very low, compared with the video sizes under different number of layers. For example, given a SVC video with 1MB size and 7 layers, the storage overhead is only 928 bytes, just 0.09% of the video size. And for a fixed number of layers, the ratio quickly diminishes and becomes negligible as the video size gets larger.

Computation costs. We measure the time consumed by different computation tasks during the initial upload. Recall that the computation tasks consist of the following four components: 1. generation of message-derived tag α_1 via RSA-OPRF; 2. SVC video encryption; 3. generation of the masked key r (message-derived label β via RSA-OPRF included); 4. key τ encryption. We note that the time of key encryption is negligible compared to other tasks, so we focus on the first three ones, which are denoted as *vEnc*, *tGen*, and *rGen*, respectively. Besides, *vEnc* is regarded as the necessary operation to safeguard the data confidentiality. Thus, the computation overhead for secure deduplication lies in the other two components, i.e., *tGen* and *rGen*.

Figure 5 depicts the computation costs of the three components when different numbers of layers of a 95MB SVC

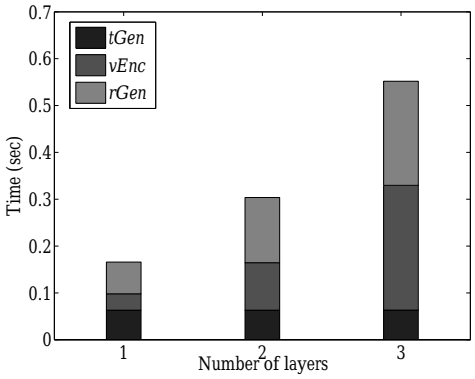

Figure 5: Time costs of different computation tasks for initially uploading different numbers of layers of a SVC video. There are totally three layers and the size is 95MB.

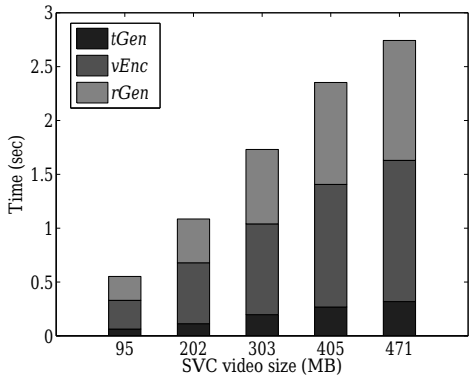

Figure 6: Time costs of different computation tasks for initially uploading SVC videos with the same number of layers but with different sizes. Here each SVC video has three layers.

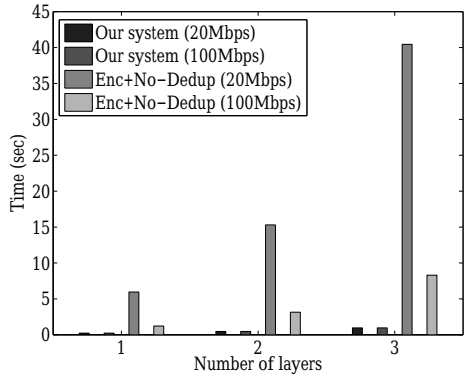

Figure 7: Running time comparisons between our system and $\mathsf{Enc} + \mathsf{No} - \mathsf{Dedup}$ setting, over different numbers of duplicate layers for a given 95MB SVC video, which totally has three layers.

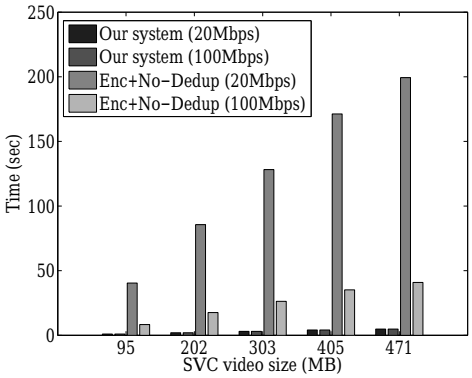

Figure 8: Running time comparisons between our system and $\mathsf{Enc} + \mathsf{No} - \mathsf{Dedup}$ setting, over SVC videos in different sizes but with the same number (three) of duplicate layers.

video is uploaded (totally three layers). The network delays in the RSA-OPRF protocol are not considered. It is observed that the time spent on *vEnc* and *rGen* increases linearly with the number of layers, since the related operations should be conducted for each layer. In contrast, *tGen* remains constant since it is only related to the base layer. Figure 6 shows the computation costs for SVC videos with different sizes but with the same number of layers. When the video sizes increase, the layer sizes increase as well, so each computation task will take more time to finish. Even though, our system is quite practical, e.g., the overall computation cost for a 471MB SVC video is less than 3 seconds.

Time savings. We measure the time savings for the user by using our secure deduplication system, compared with an Enc+No-Dedup policy that always sends the encrypted duplicate SVC videos (layers) to cloud. Two network settings are considered, including a fast network (20Mbps) and an extremely fast network (100Mbps) [1].

Figure 7 compares the running time between our deduplication design and the $\mathsf{Ennc} + \mathsf{No} - \mathsf{Dedup}$ policy, over different numbers of duplicate layers of a 95MB SVC video. Note that the running time in the compared policy includes the layer encryption time and the network transfer time of the layer ciphertexts, while our system includes all the operations specified above for a user to earn the ownership of duplicate layers from cloud. For both two network settings, our design always consumes much less time than the compared policy. The reason is that client-side deduplication is enforced in our system and duplicate layers would not be transferred through the network. Besides, it is noted that although some mechanisms such as the PoW protocol are involved in our system, the advantages in time savings rendered by deduplication remain unaffected. Figure 8 further compares the running time between our deduplication design and the policy over SVC videos in different sizes ranging from 95MB to 471MB with the same number of duplicate layers. As shown, for the 95MB SVC video in the 20 Mbps network setting, the compared policy takes 40.4484 seconds, while the time for our system is only 0.9681 seconds; in the 100 Mbps network setting, the compared policy takes 8.3025 seconds, while the time for our system is only 0.9676 seconds. On average, the time savings provided by our system can achieve about 97% in the 20Mbps network setting and about 88% in the 100Mbps network setting, respectively.

8. RELATED WORK

8.1 Secure Deduplication

Convergent encryption (CE) is first proposed by Douceur et al. [10] for secure deduplication, which enforces data confidentiality while enabling deduplciation. It encrypts/decrypts a file with a convergent key obtained through computing the hash value of the file content. Thus, the same file will always map to the same ciphertext, making deduplication feasible. Later, Bellare et al. [3] formalize CE under the name of message-locked encryption (MLE) and explore its application in secure space-efficient cloud storage, which can support both client-side and server-side deduplication. Recently, Li et al. [14] present a key management scheme based on secret sharing to protect the convergent keys in secure deduplication. The scheme constructs and distributes secret shares of keys across multiple independent servers. However, CE is inherently vulnerable to off-line brute-force attacks over predictable files [2]. The security of CE only holds for unpredictable files.

To securely use server-side deduplication and resist off-line brute-force attacks over predictable files in CE, Bellare et al. [2] resort to a key server (KS) to first provide message-based keys (i.e., hash values protected under the KS's secret key) without disclosing any information on the users' data. Then they adopt rate-limiting strategies on KS to mitigate the online brute-force attacks in practice. Another work proposed by Puzio et al. [17] employ a server to perform additional encryption over the convergent-encrypted data collected from all users. Without knowing the server's secret key, cloud cannot launch off-line brute-force attacks over predictable files. However, their design is only suited for server-side deduplication and the server has to suffer from heavy communication overhead.

Apart from the above inherent vulnerability, CE is insecure in the bounded leakage setting, where the data hash (i.e., the convergent key) may be disclosed [25]. Accordingly, Xu et al. [25] propose a client-side deduplication scheme for encrypted files in the bounded leakage setting. However, for predictable data, the proposed scheme does not consider the defense on the off-line brute-force attacks against the honest-but-curious cloud. Our system design enhances the security via the help of the agency.

8.2 Security Protection for SVC Videos

In [22], Wei et al. present a scalable and format-compliant encryption scheme to protect SVC bitstreams when being disseminated through an open network. The scheme constructs new NALUs to replace the original ones to preserving the SVC scalability. And the resulting encrypted SVC bitstream has the original SVC structure without emulation markers or illegal codewords for the standard SVC decoder, and thus achieves format-compliance. Deng et al. [9] propose an efficient block-based encryption scheme for SVC bitstream encryption. They consider a scenario where a pay TV broadcaster intends to provide a base layer version of the broadcasted program for everyone, but only allows authorized users to further get access to the enhancement layers. Therefore, the base layer is left in the cleartext while the enhancmenet layers are encrypted. The encryption of enhancement layers is achieved by employing secure pseudorandom permutations on macroblocks and subblocks. In [23], Wu et al. study the problem of attribute-based access control on SVC videos in cloud-based content sharing networks. Specifically, they present a novel multiple-message ciphertext policy attribute-based encryption (MCP-ABE) scheme, which can deliver multiple messages within one ciphertext, compared with the traditional CP-ABE scheme [5]. For a SVC video, the scheme constructs a key graph, encrypts layers with the corresponding keys and employs the MCP-ABE to encrypt the key graph. Users with different privileges can first decrypt the encryption of the key (sub)graph, and then decrypt the corresponding encrypted layers. Different from the above works, we investigate secure deduplication over encrypted SVC videos.

9. CONCLUSIONS

In this paper, we have designed and implemented an encrypted cloud media center that hosts encrypted SVC videos and supports secure deduplication. We first formulate a secure deduplication framework with strong protection for videos, which can protect the confidentiality in the bounded leakage setting and defend the off-line brute-force attacks over predictable data, respectively. Under the proposed framework, we then leverage the layered nature of SVC and propose the layer-level deduplication over encrypted SVC videos. We thoroughly analyze the security guarantee of our system against both malicious outside adversaries and honest-but-curious inside adversaries. Our implementation adopts the encryption strategy compatible with the structure and the format of SVC, and optimizes the way encrypted SVC videos are stored to improve the dissemination efficiency. The extensive experiments on Amazon cloud platform further demonstrate the practicality of our system.

In future work, we plan to investigate the support for multiple scalabilities (e.g., time and resolution) of SVC in our layer-level deduplication construction and extend our design for more general cases.

Acknowledgments

This work was supported in part by Research Grants Council of Hong Kong (Project No. CityU 138513), NSFC under grants 61472316 and 61172090, Ph.D. Programs Foundation of Ministry of Education of China under grant 20120201110013, and Shaanxi Science and Technology Innovation Project under grant 2013SZS16-Z01/P01/K01. The authors would like to thank Prof. Robert H. Deng and Dr. Zhuo Wei for sharing their code in [23] on SVC related software during the early stage of this work.

10. REFERENCES

[1] Akamai. The Akamai State of the Internet Report. `http://www.akamai.com/stateoftheinternet/`.

[2] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In *Proc. of USENIX Security*, 2013.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *Proc. of EUROCRYPT*, 2013.

[4] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proc. of IEEE SP*, 2007.

[6] M. Blestel and M. Raulet. Open svc decoder: a flexible svc library. In *Proc. of ACM MM*, 2010.

[7] J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. In *Proc. of EUROCRYPT*, 2007.

[8] Cisco Visual Networking Index. Global mobile data traffic forecast update: 2012-2017.

[9] R. H. Deng, X. Ding, Y. Wu, and Z. Wei. Efficient block-based transparent encryption for h.264/svc bitstreams. *Multimedia Systems*, 20(2):165–178, 2014.

[10] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Proc. of IEEE ICDCS*, 2002.

[11] S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography, Lecture Notes in Computer Science*, pages 577–594. Springer, 2009.

[12] A. Kathpal, M. Kulkarni, and A. Bakre. Analyzing compute vs. storage tradeoff for video-aware storage efficiency. In *Proc. of USENIX HotStorage*, 2012.

[13] S. Lederer, C. Müller, and C. Timmerer. Dynamic adaptive streaming over http dataset. In *Proc. of ACM MMSys*, 2012.

[14] J. Li, X. Chen, M. Li, J. Li, P. P. Lee, and W. Lou. Secure deduplication with efficient and reliable convergent key management. *IEEE Trans. on Parallel and Distributed Systems*, 25(6):1615–1625, 2014.

[15] North Carolina Daily. Snapchat nude photos, videos reportedly leaked online. `http://www.northcarolinadaily.com/index.php/sid/226634683`, 2014.

[16] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *Proc. of IEEE CVPR*, 2011.

[17] P. Puzio, R. Molva, M. Önen, and S. Loureiro. Cloudedup: secure deduplication with encrypted data for cloud storage. In *Proc. of IEEE CloudCom*, 2013.

[18] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, and Y. L. Louédec. Efficient http-based streaming using scalable video coding. *Signal Processing: Image Communication*, 27(4):329–342, 2012.

[19] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Trans. on Circuits and System for Video Technology*, 17(9):1103–1120, 2007.

[20] T. Stutz and A. Uhl. A survey of h. 264 avc/svc encryption. *IEEE Trans. on Circuits and Systems for Video Technology*, 22(3):325–339, 2012.

[21] Z. Wei, Y. Wu, R. H. Deng, and X. Ding. A hybrid scheme for authenticating scalable video codestreams. *IEEE Trans. on Information Forensics and Security*, 9(4):543–553, 2014.

[22] Z. Wei, Y. Wu, X. Ding, and R. H. Deng. A scalable and format-compliant encryption scheme for h.264/svc bitstreams. *Signal Processing: Image Communication*, 27(9):1011–1024, 2012.

[23] Y. Wu, Z. Wei, and R. H. Deng. Attribute-based access to scalable media in cloud-assisted content sharing networks. *IEEE Trans. on Multimedia*, 15(4):778–788, 2013.

[24] S. Xiang. Scalable streaming. `https://sites.google.com/site/svchttpstreaming/storagesaving`.

[25] J. Xu, E. Chang, and J. Zhou. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In *Proc. of ACM AISACCS*, 2013.

[26] W. Zhu, C. Luo, J. Wang, and S. Li. Multimedia cloud computing. *IEEE Signal Processing Magazine*, 28(3):59–69, 2011.

APPENDIX

A. THE RSA-OPRF PROTOCOL

Table 1: The RSA-OPRF protocol. (N, e) **and** (N, d) **are the agency's public key and secret key, respectively, which are as in the RSA system.** M **denotes the message to be signed.** $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{Z}_N^*$ **and** $\mathsf{H}_2 : \mathbb{Z}_N^* \to \{0,1\}^\lambda$ **are two hash functions.**

User		**Agency**
1. $\gamma \xleftarrow{\$} \mathbb{Z}_N^*$		
2. $h \leftarrow \mathsf{H}_1(M)$		
3. $x \leftarrow h \cdot \gamma^e mod N$		
	$x \longrightarrow$	
		4. $y \leftarrow x^d mod N$
	$\longleftarrow y$	
5. $z \leftarrow y \cdot \gamma^{-1} mod N$		
6. If $z^e mod N \neq h$ then ret $\perp$; Else ret $\mathsf{H}_2(z)$		