10-2009

A study of content authentication in proxy-enabled multimedia delivery systems: Model, techniques, and applications

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Yanjiang YANG
Singapore Management University, yjyang@smu.edu.sg

DOI: https://doi.org/10.1145/1596990.1596992

Published in ACM Transactions on Multimedia Computing, Communications and Applications
2009 October, 5 (4), Article 28.
https://doi.org/10.1145/1596990.1596992

A Study of Content Authentication in Proxy-Enabled Multimedia Delivery Systems: Model, Techniques, and Applications

ROBERT H. DENG
Singapore Management University
and
YANJIANG YANG
Institute for Infocomm Research

Compared with the direct server-user approach, the server-proxy-user architecture for multimedia delivery promises significantly improved system scalability. The introduction of the intermediary transcoding proxies between content servers and end users in this architecture, however, brings unprecedented challenges to content security. In this article, we present a systematic study on the end-to-end content authentication problem in the server-proxy-user context, where intermediary proxies transcode multimedia content dynamically. We present a formal model for the authentication problem, propose a concrete construction for authenticating generic data modality and formally prove its security. We then apply the generic construction to authenticating specific multimedia formats, for example, JPEG2000 code-streams and MPEG-4 video streams. The prototype implementation shows that our scheme is suitable for practical applications.

Categories and Subject Descriptors: H.4.0 [**Information Systems Applications**]: General; I.7.2 [**Document and Text Processing**]: Document Preparation—*Languages and systems; Photocomposition/typesetting*

General Terms: Security, Reliability

Additional Key Words and Phrases: Multimedia content delivery, security, end-to-end authentication

ACM Reference Format:

Deng, R. H. and Yang, Y. 2009. A study of content authentication in proxy-enabled multimedia delivery systems: Model, techniques and applications. ACM Trans. Multimedia Comput. Commun. Appl. 5, 4, Article 28 (October 2009), 20 pages.
DOI = 10.1145/1596990.1596992 http://doi.acm.org/10.1145/1596990.1596992

1. INTRODUCTION

The server-proxy-user architecture, which incorporates intermediary transcoding proxies between multimedia server and end users, is increasingly becoming accepted as a promising paradigm for multimedia

This work is partly supported by the Office of Research, Singapore Management University. Part of the article has been published in Proceedings of the 2007 Information Security Practice and Experience Conference (ISPEC'07), 284-300.

Authors' addresses: R. H. Deng, School of Information Systems, Singapore Management University; email: robertdeng@smu.edu.sg; Y. Yang, Institute for Infocomm Research, 1 Fusionopolis Way,2-1 Connexis, Singapore 138632; email: yyang@i2r.a-star.edu.sg.

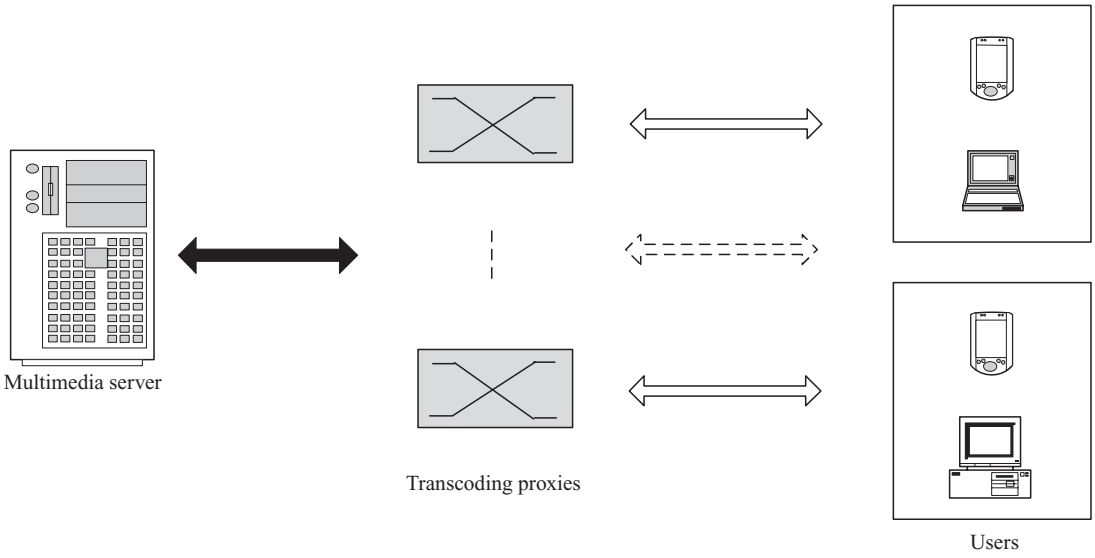

Fig. 1. A server-proxy-user multimedia content delivery system.

content delivery. In the server-proxy-user system, as depicted in Figure 1, one or more intelligent intermediary proxies reside along the path from a server that disseminates multimedia content to end users who are content consumers. Each proxy serves a distinct group of users out of the entire user population of the server and is entrusted by the multimedia server to perform certain transcoding operations upon the content according to the end users' specific capabilities, configurations, or preferences. For example, a proxy downscales a high-quality image to a small thumbnail, in replying to a user request from a handheld device with limited processing and display capabilities. In many cases, multimedia content may pass several proxies and undergo multiple kinds of transcoding operations before reaching the end users.

A major advantage of the proxy-enabled multimedia delivery system is its scalability. In a traditional server-user system, the server inevitably becomes the system bottleneck as the size of user population increases. The introduction of intermediary proxies in the server-proxy-user systems solves this problem by amortizing the processing of user requests from a single server to multiple proxies, which not only relieves the server from intense data processing, but also shortens the request response time due to caching of content at the proxies [Bhattacharjee et al. 1998; Cardellini et al. 2000; Li and Shen 2005; Shen et al. 2003; Smith et al. 1998].

As consumption of multimedia content becomes increasingly routine in our daily lives, server-proxy-user multimedia delivery systems have many applications. The following are some examples, among numerous others.

1.1 Application Scenarios

1.1.1 *Dynamic Content Adaption in Pervasive Computing.* As pervasive computing is increasingly becoming a reality, there is a growing need to adapt multimedia content delivery to a variety of client devices of varying computational and storage capacities, screen sizes and bandwidth connections. The traditional solution to content adaption is to have the multimedia server either transform the multimedia content on the fly or generate every possible version in advance and deliver the appropriate version

based on users' requests. Neither method, however, is satisfactory as they make the server heavily involved in computation or storage. Many proposals, for example, Bjork and Christopoulos [2000], Huang et al. [2004], Libsie and Kosch [2002], Ooi and van Renesse [2001], and Yeung et al. [2002], suggested using server-proxy-user systems to improve performance: the server entrusts intermediary proxies to dynamically transcode multimedia content based on the specific user constraints. Transcoding operations in these kinds of applications are typically *content downscaling*: multimedia streams originated from the multimedia server are normally the highest in quality and the richest in content; the intermediary proxies downgrade the quality of the streams by removing certain content items in order to fit the particular competence of client devices.

1.1.2 *Multimedia Composition.* Multimedia composition systems, for example, Li et al. [2005], Wagner and Kellerer [2004], and Yamaoka et al. [2005], are a special instance of the server-proxy-user multimedia delivery architecuture, where the intermediary proxies are required to transcode and syndicate multiple multimedia-related streams from the multimedia server (or from different servers) into a single stream, before sending the composite stream to end users. An example is that the multimedia server sends separate data modalities to the transcoding proxies: video, audio, and text; according to the users preferences, the proxies translate the text into a user-preferred language, and then combine the video content, the audio content, and the text together into a composite stream. Typical transcoding operations in the multimedia composition applications involve not only content downscaling but also *content alteration*: changing a part of the original multimedia content.

1.1.3 *Tiered Multimedia Distribution Systems.* Another class of applications that follow the server-proxy-user model is tiered multimedia distribution systems [Suzuki et al. 2004; Yuuichi et al. 2003]. Such a system consists of a top-tier primary content provider and a number of lower-tier affiliating providers each with its own users. An example is a multinational company that has a number of affiliated vendors worldwide; to promote a new product, the company produces a video advertisement for the product and delivers it to all the affiliated vendors; in order to better fit the local market, each affiliated vendor is entitled to derive its own local version (such as adding subtitles in the local language) based on the original advertisement. In scenarios like this, a lower-tier content provider may perform transcoding operations such as content downscaling, content alteration, and *content insertion*: adding new data to the original content.

Copyright protection may also be a motivation for transcoding by the lower-tier content providers in tiered multimedia distribution systems. For example, the primary content provider sends multimedia content to the affiliated lower-tier providers which then sell content to end users. Before delivering the content to a user, the affiliated provider transcodes the content by embedding a user-specific mark into the content in an attempt to trace pirated copies [Guo and Georganas 2002; Kirovski et al. 2002; Schonberg and Kirovski 2004; Schlauweg et al. 2006].

1.1.4 *Transcoding Proxy for Medical Images.* Halle and Kikinis [2004], Mohammed and Fiaidhi [2005], and Parmanto et al. [2005] described server-proxy-user delivery systems for images. Because of the very nature of medical applications, images need to present various modalities to meet different requirements of doctors and medical procedures. For example, one surgeon may require a 3D volume of a medical image while another may demand a series of 2D slices of the same image. As modality transformation for medical images is common, the server-proxy-user multimedia delivery systems particularly suit medical scenarios. Another important factor that justifies the use of proxy-enabled systems for medical imaging systems is the increasing deployment of wireless services in healthcare environments. Transcoding operations for medical images include *format conversion*, for example, from GIF to JPEG, and structural changes to the presentation format [Mohammed and Fiaidhi 2005].

While the server-proxy-user architecture has been accepted as a scalable new paradigm for multimedia content delivery, we notice however that security issues in such systems have not been rigorously explored and many of the existing studies have certain limitations (see Section 2 for details). In critical fields such as government, finance, health care, and the law, it is crucial and often a requirement to ensure end users of the authenticity of the content they received. Many data authentication techniques, such as MAC (Message Authentication Code) and digital signature, have been proposed in the literature. However these techniques completely ignore the internal structure of the underlying data, and thus are not efficient when applied to multimedia content.

1.2 Our Contributions

Our focus in this article is achieving end-to-end authentication from the multimedia server to end users in the presence of intermediary transcoding proxies. As we have demonstrated earlier, content transcoding is inevitable and beneficial in server-proxy-user multimedia delivery systems. However, such transcoding would definitely invalidate the integrity-checking data (or authentication data) upon the content generated by the multimedia server if standard techniques such as digital signature or MAC are used. The challenge is thus to enable content transcoding by the proxies while without disabling the authentication data generated by the server. It should be clear that sustaining and verifying the intactness of the original content is not the objective in this context; instead, *authentication* relates to entitling the end users to verify origin of data as well as authorized data alteration. In particular, we systematically study the authentication problem in the proxy-enabled multimedia delivery systems with the following specific contributions.

—We present a formal system model for the end-to-end authentication problem in server-proxy-user multimedia delivery systems.

—We propose a concrete construction for end-to-end authentication of generic data content based on the Merkle hash tree [Merkle 1989] and sanitizable signatures [Ateniese et al. 2005]. The Merkle hash tree allows for efficient verification of a subset of data given a signature upon the whole set. A sanitizable signature allows authorized semi-trusted proxies to modify parts of a signed message without interacting with the original signer. The main difference between our construction and the sanitizable signature [Ateniese et al. 2005] is that the latter allows for content alteration only while the former supports all of the aforementioned transcoding operations. Moreover, our construction is designed for end-to-end content authentication in the server-proxy-user multimedia delivery systems. We also provide a formal proof on the security of the proposed construction.

—We apply and optimize our schemes for authentication of JPEG2000 code-streams and MPEG-4 video streams. We remark that the cryptographic primitives proposed in Johnson et al. [2002], Miyazaki et al. [2006], and Steinfeld et al. [2001], while not explicitly designed for authentication of multimedia content, are applicable to server-proxy-user multimedia delivery systems, but can only address content downscaling.

1.3 Article Organization

We review related work in Section 2. In Section 3, we propose a formal system model for the end-to-end authentication problem in the server-proxy-user multimedia delivery systems. Section 4 presents our concrete construction under the proposed model by considering generic data modality, together with its security analysis. We then apply and tailor the generic scheme to authenticating specific multimedia formats such as JPEG2000 code-streams and MPEG-4 video streams in Section 5. Implementation results are presented in Section 6, and Section 7 concludes the article.

2. RELATED WORK

Our work directly relates to multimedia authentication using cryptographic techniques, which in our taxonomy is categorized into two classes: multimedia authentication in the server-user systems and in the server-proxy-user systems, and as we shall see shortly, the two have quite different focuses. Digital watermarking has also been used for the purposes of multimedia authentication (e.g., Schneider and Chang [1996], Lin and Chang [2001], Fridrich et al. [2004], and Swaminathan et al. [2006]), but it cannot protect the whole content of the underlying data and must sacrifice a small portion of the content for watermark embedding. This differs significantly from authentication using cryptography, and thus is outside the scope of our discussion. Other related efforts include Guo and Georganas [2002], Kirovski et al. [2002], and Schonberg and Kirovski [2004], which deal with Digital Right Management, and Deng et al. [2004], Furht and Kirovski [2006], Liu and Eskicioglu [2003], Shi and Bhargava [1998], and Yeung et al. [2002], which concern protecting confidentiality of multimedia content.

2.1 Multimedia Authentication in Server-User Systems

The focus of authentication in server-user systems is to efficiently authenticate multicast multimedia streams in the presence of lossy dissemination channels. A number of papers examined solutions to this problem using either secret key-only operations [Briscoe 2000; Bergadano et al. 2000; Perrig 2001; Perrig et al. 2000, 2001] or asymmetric cryptographic techniques [Golle and Modadugu 2001; Krohn et al. 2004; Lysyanskaya et al. 2004b; Miner and Staddon 2001; Park et al. 2003; Pannetrat and Molva 2003; Rohatgi 1999]. The basic rationale of the secret key-only approach is delayed disclosure of the symmetric keys by the multimedia server to the users so as to provide source authentication (the received content indeed originates from the source and has not been modified enroute). Taking TESLA [Perrig et al. 2000] for example, the multimedia server sends out a packet together with its MAC computed using a secret key k known only to itself; upon reception of a packet, a user buffers the packet without being able to immediately authenticate it; after a reasonably short period of time, the server discloses k to enable users to authenticate the packet(s) whose MAC is computed with k. The advantage of the secret key-only approach is efficiency, but it cannot achieve nonrepudiation—the users are unable to convince a third party that the received content came from the designated source.

Using digital signature solves the nonrepudiation problem, which leads to the asymmetric cryptography based approach. However, asymmetric cryptographic operations are expensive, the main objective of the asymmetric cryptography based approach is thus to amortize asymmetric operations over several packets or data blocks. For example, Rohatgi [1999] uses reduced-size online/offline k-time signatures, Golle and Modadugu [2001] adopt a tree-based method and Miner and Staddon [2001] follow a graph-based method to reduce authentication overhead. Krohn et al. [2004], Lysyanskaya et al. [2004b], Park et al. [2003], and Pannetrat and Molva [2003] in addition employ error correcting codes such as erasure coding, to strengthen authentication.

Regardless of secret key-only approaches and asymmetric cryptography-based approaches, authentication in server-user systems essentially deals with authenticating multimedia streams when they suffer from stochastic packet losses during transmission. In other words, data manipulation suffered in these systems is in an uncontrolled manner. In comparison, server-proxy-user systems have a quite different scenario to consider: data transcoding is performed in a controlled manner from the perspective of the multimedia server. As a consequence, the methods for the server-user systems are not directly applicable to the server-proxy-user systems.

2.2 Multimedia Authentication in Server-Proxy-User Systems

We are not the first to study the authentication problem in server-proxy-user systems; there are several prior works. Deng et al. [2005, 2004], Gentry et al. [2005], Peng et al. [2003], and Wu and Deng [2006]

propose using the hash chain or the Merkle hash tree to achieve authentication of multimedia content, while allowing for content, downscaling transcoding by the intermediary proxies. The basic idea is that the multimedia server signs a single piece of authentication data derived from the underlying multimedia content by organizing the content into a chain or a Merkle hash tree, and this signature is still verifiable given a portion of the content together with some auxiliary authentication information related to the missing parts. In particular, the methods in Gentry et al. [2005] consider generic multimedia content by treating the content as a set of data packets, while Deng et al. [2005, 2004], Peng et al. [2003], and Wu and Deng [2006] propose solutions to specific multimedia modalities such as JPEG2000 images and MPEG-4 video streams. While of high efficiency, these proposals only handle content downscaling, but not other transcoding operations such as content alteration.

An authentication solution to composition of MPEG-4 video streams by the intermediary proxies is presented in Li et al. [2005], where the multimedia server signs the original multimedia streams prior to dissemination, and the intermediary proxies en route sign the parts they modified and then combine the previous signatures and the current signature into an aggregate signature [Lysyanskaya et al. 2004a]. The final resulting aggregate signature enables verification of all the signatures that have been aggregated. A clear advantage of aggregate signature is space efficiency, but it loses *locality*, it is not possible to pinpoint the parts that invalidate the final aggregate signature given that certain streams experienced illegitimate manipulations en route. In our construction, we can choose to offer locality to the parts that are expected to be altered.

The work most related to ours is Suzuki et al. [2004], which also considers achieving end-to-end authentication of multimedia content while accommodating transcoding operations by the intermediary proxies. The method they employ is that the multimedia server issues a signature upon the hash value of the underlying multimedia content, which is computed by applying a trapdoor hash function; the proxy knowing the secret trapdoor can find a collision of the hash value. Unfortunately, the trapdoor hash function they use has a serious weakness: publishing different transcoding results of the same multimedia content reveals the secret trapdoor, which means the intermediary proxy can only generate a single version of transcoded content. This clearly limits the use of their method in practice. In contrast, our construction entitles an intermediary proxy for unlimited times of transcodings by utilizing the trapdoor hash function used to generate sanitizable signatures [Ateniese et al. 2005]. Moreover, we consider not only generic multimedia data, but also specific multimedia formats such as JPEG2000 code-streams and MPEG-4 video streams; we also give a formal model for the end-to-end authentication problem.

3. FORMAL SYSTEM MODEL

Participants in a server-proxy-user multimedia delivery system include a multimedia server, a group of users, and one or several intermediary proxies located between the server and the users. The multimedia server is the originator of the multimedia content, a proxy is authorized by the multimedia server to perform certain transcoding operations on the content it receives, and users are the ultimate consumers of the content. The proxies are semi-trusted in the sense that they transcode a content in a prescribed way, for example, to meet the constraints of the users' devices. Informally, authentication in the server-proxy-user multimedia delivery systems is to enable users to verify the originality of the content, and to verify that content transcoding has been performed by the authorized intermediary proxies (formal definition will be summarized in the form of security requirements). We stress that authentication in our model must be enabled in the presence of the following transcoding operations.

—Content downscaling. The multimedia server authorizes an intermediary proxy to drop some portions of the content. Normally, content downscaling performed by the proxy is to harmlessly downgrade

the quality of the multimedia content in order to fit the end users' devices. Of course, a malicious proxy or an attacker can always make the content unrenderable to the users' devices by dropping the core portions of the content, for example, dropping the R_0 subband in an JPEG2000 code-stream (see Section 5.1 for the introduction of the JPEG2000 code-stream). Such an attack constitutes a type of denial of service attack and is beyond the scope of this article.

—Content alteration. The multimedia server authorizes an intermediary proxy to modify certain designated parts of the content.

—Content insertion. The multimedia server authorizes an intermediary proxy to insert new content at designated locations of the content. In our construction, the multimedia server inserts blank data holders at the locations where the intermediary proxy is expected to insert new content; subsequently, the proxy replaces the blank data with the content it wishes to insert. As a result, content insertion is reduced to content alteration, and we do not explicitly distinguish them.

3.1 The Model

Our end-to-end authentication scheme for the server-proxy-user multimedia delivery systems consists of four efficient algorithms for key generation, signing, transcoding, and verification, respectively.

Key Generation. The key generation algorithm *KeyGen* is a probabilistic polynomial-time algorithm that takes as input the security parameter 1^k, and outputs key pairs for the multimedia server and each intermediary proxy.

$$(PK_S, SK_S; \{PK_{P_i}, SK_{P_i}\}_i) =: KeyGen(1^k),$$

where (PK_S, SK_S) is the key pair for the multimedia server to sign the original content, and (PK_{P_i}, SK_{P_i}) is the key pair for proxy i, used for transcoding. The former can be a key pair for any standard signature scheme while the latter can be a key pair for any digital signature scheme or any public key encryption scheme. The key generation algorithm may be invoked by a trusted third party (e.g., CA, certificate authority) in a preprocessing step.

Signing. The signing algorithm *Sign* takes as input a multimedia content m, the private key of the multimedia server SK_S, the set of public keys $\{PK_{P_i}\}_i$ of the proxies, and random coins r, and outputs a digital signature σ.

$$\sigma =: Sign(m, r, \{PK_{P_i}\}_i, SK_S).$$

The signing algorithm is performed by the multimedia server to produce a signature on the multimedia content to be disseminated.

Transcoding. The trancoding algorithm *Transcode* takes as input a multimedia content m,[1] the signature σ on m, the random r, the public key of the server PK_S, and the private key, SK_{P_i}; of proxy i which performs transcoding. It outputs a transcoded content m', random coins r', and possibly auxiliary authentication information AAI.

$$(m', r', AAI) =: Transcode(m, \sigma, r, PK_S, SK_{P_i}).$$

The presence of the auxiliary authentication information AAI is due to content downscaling. If there is no content downscaling, AAI will be nil: $AAI = \phi$. The transcoding algorithm is performed by the intermediary proxy.

[1] m is not necessarily the original content from the multimedia server, and could be transcoded content from other proxies.

Verification. The verification algorithm *Verify* is a deterministic algorithm with output over $\{0, 1\}$. It takes as input a multimedia content $\bar{m}$, a signature $\bar{\sigma}$ on $\bar{m}$, random coins $\bar{r}$, the public key of the server PK_S, the set of public keys $\{PK_{P_i}\}_i$ of the proxies , and $\overline{AAI}$ if any, and outputs either 0 or 1.

$$\{0, 1\} =: \mathit{Verify}(\bar{m}, \bar{\sigma}, \bar{r}, \overline{AAI}, \{PK_{P_i}\}_i, PK_S).$$

The verification algorithm can be performed by anyone.

3.2 Security Requirements

This authentication scheme must satisfy the following security requirements.

Correctness. A signature generated by the signing algorithm should be accepted by the verification algorithm.

$$\forall\, \sigma = \mathit{Sign}(m, r, \{PK_{P_i}\}_i, SK_S) \Rightarrow$$
$$\mathit{Verify}(m, \sigma, r, AAI = \phi, \{PK_{P_i}\}_i, PK_S) = 1.$$

Security. Informally, without the knowledge of the private signing key, it is computationally infeasible to generate a valid signature on a content under the corresponding public key, unless the content is a transcoded content from a *legitimate* transcoding. We next give a definition on legitimacy of transcoding.

Definition 1. *Legitimate transcoding*. Without loss of generality, let us suppose m is a set of n elements, $m = \{m_1, m_2, \ldots, m_n\}$, and we consider a particular proxy with key pair (PK_P, SK_P). (m', r', AAI) $= \mathit{Transcode}\,(m, \sigma, r, PK_S, SK_P)$ is a legitimate transcoding of m with respect to σ if one of the following conditions holds:

C_1 $m' \subseteq m$; or

C_2 $\sigma = \mathit{Sign}(m', r', PK_P, SK_S)$ in case $AAI = \phi$; or

C_3 $\mathit{Verify}(m', \sigma, r', AAI, PK_P, PK_S) = 1$ in other cases.

We shall provide some clarification on this definition. C_1 states that transcoding that only involves content downscaling (resulting in $m' \subseteq m$) is trivially legitimate and we call m' a *trivial transcoding content* of m. The reason that justifies this is that authentication in the context of server-proxy-server systems is to verify data origin and is not to verify data *intactness* or *completeness*. C_2 states that if content downscaling is not involved ($AAI = \phi$), a legitimate transcoding makes the original signature σ exactly the signature on the transcoded content m' using random coins r'. C_3 covers all other cases, where the only way to determine a legitimate transcoding is to check the verification algorithm. We call m' in C_2 and C_3 a *nontrivial transcoding content* of m. Note that C_3 has already covered C_1 and C_2, but transcoding in C_1 and C_2 yields special structures, so we explicitly list them as separate conditions.

Formally, we formulate the security property as an adversarial game given in the following.

Definition 2. *Secure End-to-end Multimedia Authentication Scheme*. Algorithms (*KeyGen*, *Sign*, *Transcode*, *Verify*) constitute a secure end-to-end multimedia authentication scheme if the advantage of any probabilistic polynomial-time adversary $\mathcal{A}$ in the following game is negligible (with respect to the security parameter k):

(1) A key pair (PK_S, SK_S) for signing and a key pair (PK_P, SK_P) for transcoding are generated:

$$(PK_S, SK_S; PK_P, SK_P) =: \mathit{KeyGen}(1^k).$$

(2) The adversary $\mathcal{A}$ is given:
—the public keys PK_S and PK_P as inputs;

—the first phase of oracle access to *Sign* algorithm and *Transcode* algorithm, respectively; that is, $\mathcal{A}$ is free to query $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$;

(3) At the end of the first phase of the game, $\mathcal{A}$ outputs a message m and a state that represents the knowledge acquired during this phase.

(4) $\mathcal{A}$ continues with the second phase of oracle queries to $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$, under the restrictions that the total number of queries issued to $\mathcal{O}^{SK_S}$ is less than Q_S, and to $\mathcal{O}^{SK_P}$ is less than Q_P during the two phases of oracle accesses, where Q_S and Q_P are the maximum number of queries allowed to the corresponding oracles, depending on the security parameter.

(5) Last, $\mathcal{A}$ outputs a signature σ on m, together with the corresponding random r and AAI.

Adversary $\mathcal{A}$ wins the game (breaking the security of the authentication scheme) if all of the following conditions are met:

—$Verify(m, \sigma, r, AAI, PK_P, PK_S) = 1$;

—m is never queried to $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$;

—m is not a trivial transcoded content of any message queried to $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$ during the game.

The advantage of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game, which is computed over the random coins generated by $\mathcal{A}$.

To keep our notations simply, we focus in the sequel, on the single proxy model in our discussion.

4. OUR CONSTRUCTION

In this section, we give a concrete construction of the end-to-end authentication scheme satisfying the preceding security requirements. We first review the idea of sanitizable signatures [Ateniese et al. 2005], which we exploit in our construction. We then present a scheme for authentication of generic multimedia content, and formally analyze its security.

4.1 Sanitizable Signatures

We only review the basic idea of sanitizable signatures [Ateniese et al. 2005], which suffices in understanding our scheme. To generate a sanitizable signature on a message, a *trapdoor* hash function is first applied to the message, and then the resulting hash value is signed by a digital signature algorithm such as RSA.

An example trapdoor hash function is the following. Let p be a prime such that $p = 2q + 1$, where q is also a prime, and g be a generator of the subgroup of squares in Z_p^* of order q. Let $(y = g^x \mod p, x)$ be an Elgamal-type key pair where x is the private key serving as the secret trapdoor. Let $H()$ be a standard collision resistant hash function. The trapdoor hash function on message m under the public key y is defined as $TH_y(m, r) = (\rho - (y^e g^\delta \mod p)) \mod q$, where $r = (\rho, \delta)$ with $\rho \in_R Z_p^*$ and $\delta \in_R Z_q$, and $e = H(m, \rho)$.

Given a hash value $v = TH_y(m, r)$, only the party that knows the secret trapdoor x can compute a collision as follows: first it chooses a random value $k' \in Z_q^*$; then computes $\rho' = (v + (g^{k'} \mod p)) \mod q$, $e' = H(m', \rho')$ and $\delta' = k' - e'x \pmod q$. It is easy to see that $(m', r') = (m', \rho', \delta')$ is a collision of (m, r). It has been shown that finding a collision without knowledge of the secret trapdoor is reduced to the forgery of the twin Nyberg-Rueppel signature [Naccache et al. 2001]. A particularly desirable feature of this trapdoor hash function we desire, is that revelation of multiple collisions does not disclose the secret trapdoor.

We shall use this trapdoor hash function in the following construction. Note that in principle, our construction can use any hash function that can find collisions with the help of certain secret information. An example is the fourth variant of the VSH [Contini et al. 2006]. However, it seems that the VSH is not as efficient as the previous trapdoor hash function, since it uses a large composite as modulus. We also notice that the sibling intractable hash [Zheng et al. 1991] is not suitable to our construction, as it cannot provide unlimited number of collisions, and the collisions must be predetermined by who for an sets up the function.

4.2 The Scheme

4.2.1 *Basic Construction.* We consider generic data modality: the multimedia content m to be dispatched by the multimedia server is a data set consisting of n elements, $m = \{m_1, m_2, \ldots, m_n\}$. At a very high level, our scheme works as follows. The multimedia server (1) inserts blank items at the places where the intermediary proxy is permitted to insert new content; (2) hashes each of the blank items and the items that the proxy is authorized to alter using the above trapdoor hash function under the public key of the intermediary proxy, and hashes each of the remaining items using a standard cryptographic hash function; (3) signs the concatenation of the hash values; and finally distributes all the items along with the signature to the intermediary proxy. With the knowledge of the secret trapdoor, the intermediary proxy is enabled to perform the expected transcoding operations on the items hashed using the trapdoor hash function. We next detail the scheme.

Let (PK_S, SK_S) be the key pair of the server for signing and (PK_{P_i}, SK_{P_i}) be the key pair of proxy i for transcoding. Let $\mathcal{H}()$ be a standard collision resistant hash function modelled as random oracle [Bellare and Rogaway 1993], and $TH_{PK_{P_i}}()$ be the trapdoor hash function under PK_{P_i} (SK_{P_i} is thus the trapdoor). Without loss of generality, we assume the multimedia server inserts a blank item, blk, at the end of the data set so as to enable proxy i to append additional content at the end of the original multimedia content: $m_{n+1} = blk$. Further we assume that the server expects the proxy i to modify l items of the data set, $m_{j_1}, m_{j_2}, \ldots, m_{j_l}$.

Signing. In the signing algorithm, the multimedia server computes:

$$\begin{aligned}\sigma &= Sign(m, r, \{PK_P\}_i, SK_S)\\ &= S_{SK_S}(ID_m||\{PK_{P_i}\}_i||h_1, \ldots, h_n, h_{n+1}),\end{aligned}$$

where $S_{SK_S}()$ is the signing function of a digital signature scheme using the private signing key SK_S, ID_m is the identifier of the multimedia content m, $h_j = TH_{PK_{P_i}}(ID_m||m_j, r_j)$ for $j \in \{j_1, j_2, \ldots, j_l, n+1\}$, and $h_j = \mathcal{H}(ID_m||m_j)$ if m_j is expected not to be altered by any proxy. The multimedia server then dispatches $\{m_1, m_2, \ldots, m_{n+1}\}$, σ, and $r = (r_{j_1}, r_{j_2}, \ldots, r_{j_l}, r_{n+1})$ to the proxies.

Transcoding by proxy i. Let m'_j denote the transcoded content of m_j, $j \in \{j_1, j_2, \ldots, j_l, n+1\}$. As the intermediary proxy i knows the private key SK_{P_i} that corresponds to PK_{P_i}, it can compute r'_j such that $TH_{PK_{P_i}}(ID_m||m'_j, r'_j) = TH_{PK_{P_i}}(ID_m||m_j, r_j)$, $j \in \{j_1, j_2, \ldots, j_l, n+1\}$. We assume that the intermediary proxy performs content downscaling by removing $m_{d_1}, m_{d_2}, \ldots, m_{d_t}$ from the original data set. We further assume that each removed item, $m_d \notin \{m'_{j_1}, m'_{j_2}, \ldots, m'_{j_l}, m'_{n+1}\}$ for $d \in \{d_1, d_2, \ldots, d_t\}$. We use m' to denote the transcoded content of m after these transcoding operations. The intermediary proxy computes:

$$(m', r', AAI) = Transcode(m, \sigma, r, PK_S, SK_{P_i}),$$

where $r' = (r'_{j_1}, r'_{j_2}, \ldots, r'_{j_l}, r'_{n+1})$, and $AAI = \{h_{d_1}, h_{d_2}, \ldots, h_{d_t}\}$. The proxy forwards m', σ, r' together with AAI to the next proxy or end users. Subsequent proxies transcode in a similar way.

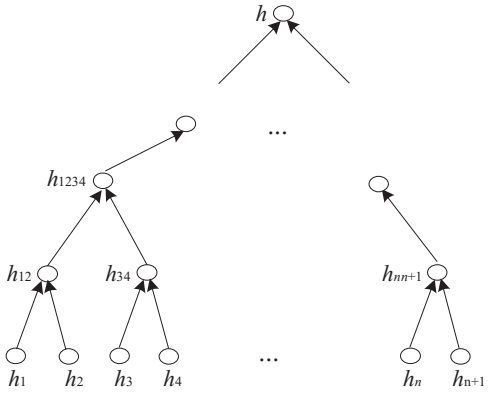

Fig. 2. Organizing hash values into a Merkle hash tree.

Verification. Upon reception of $\bar{m}$, $\bar{\sigma}$, $\bar{r}$, and $\overline{AAI}$, it is straightforward for a user to execute *Verify*$(\bar{m}, \bar{\sigma}, \bar{r}, \overline{AAI}, \{PK_{P_i}\}_i, PK_S)$: it first computes and concatenates the hash values in the same manner as in the signing algorithm. Let $\bar{h}$ denote the concatenated value; it then outputs $V_{PK_S}(\bar{h}, \bar{\sigma})$, where $V_{PK_S}(.)$ is the verification function of the digital signature scheme using PK_S.

In this basic scheme, the size of the auxiliary authentication information AAI is proportional to the number of removed items. The AAI can be minimized by employing the Merkle hash tree, as demonstrated in the following.

4.2.2 *Optimization.* In the signing algorithm, instead of directly signing the concatenation of the hash values, the multimedia server organizes the hash values into a Merkle hash tree [Merkle 1989], as depicted in Figure 2. Specifically, the leaf nodes of the tree are the set of the hash values to be signed; the value of each internal node is derived from its child nodes under a hash function (we can use the same hash function $\mathcal{H}(.)$). Referring to Figure 2, we have $h_{12} = \mathcal{H}(h_1, h_2)$, $h_{34} = \mathcal{H}(h_3, h_4)$, and $h_{1234} = \mathcal{H}(h_{12}, h_{34})$, and so on. Finally, a unique root value h is obtained. The multimedia server then generates a signature on h, $\sigma = S_{SK_S}(ID_m||PK_P||h)$.

Suppose m_1, m_2, m_3, and m_4 are removed in the subsequent transcoding operation, it suffices for the intermediary proxy to attach $AAI = \{h_{1234}\}$ to the original signature to assist the end user in verification. Comparing $AAI = \{h_{1234}\}$ to $AAI = \{h_1, h_2, h_3, h_4\}$ in the basic scheme, we see a significant reduction in the amount of auxiliary authentication information. As content downscaling is probably the most common operation in the server-proxy-user systems, the optimized scheme significantly reduces authentication overhead, as we will demonstrate more clearly in the next section.

4.2.3 *Security Analysis.* We only analyze security of the basic scheme as security of the optimized scheme is evident given the security of the basic scheme. It should also be clear that computing the values of some of the internal nodes using the trapdoor hash function does not compromise the security either.

Our construction trivially satisfies the *correctness* requirement, as we employ a standard digital signature scheme for signing. We thus focus on the *security* requirement. We call a probabilistic polynomial-time adversary a $(k, Q_S, Q_P, t, \varepsilon)$-breaker of our construction if the adversary makes at most Q_S queries to the signing algorithm and at most Q_P queries to the transcoding algorithm (Q_{TH} queries to the oracle of the trapdoor hash function and Q_H queries to the random oracle for the standard hash function, such that $Q_P = Q_{TH} + Q_H$), runs in at most t steps, and has an advantage of no less

than ε (with respect to size k) in winning the game described in Section 3.2. We have the following theorem:

THEOREM 1. *Let $\mathcal{A}$ be a (k, Q_S, Q_P, t, ε)-breaker of our construction, then there exists a (k, Q_S, t_0, ε_0)-breaker of the underlying digital signature scheme, a (k, Q_{TH}, t_1, ε_1)-breaker of the trapdoor hash function, or the regular hash function, that satisfy:*

$$\varepsilon_0 + \varepsilon_1 \geq \varepsilon$$
$$t_0 \leq t + Q_{TH}.t_{TH} + Q_H.t_H$$
$$t_1 \leq t + Q_S.t_{sign},$$

where t_{TH} is the running time for computing a collision of the trapdoor hash function, t_H is the running time of the hash function, and t_{sign} is the running time of the signing function of the digital signature scheme, all on an instance of size k.

The proof of the theorem is a *reductionist proof*—the security property of our construction is reduced to the security of the underlying digital signature scheme, the trapdoor hash function, or the regular hash function. More specifically, if there exists an attacker $\mathcal{A}$ that breaks the security property of our construction (wins the game specified in Section 3.2), then another attacker invoking $\mathcal{A}$ can break the underlying digital signature scheme, the trapdoor hash function, or the regular hash function. This leads to a contradiction, since the security of the latter has already been well established. For conciseness of presentation, we leave the detailed proof to the Appendix.

4.2.4 *Discussions.* We next discuss some extensions to this construction.

Provision of Locality. This construction does not provide locality—in the case that the verification algorithm outputs 0, it is not possible to pinpoint the items that caused the failure. We can choose to provide locality to the items to be altered, but at the price of slightly more communication overhead. For this purpose, we need an extra auxiliary authentication information AAI_{extra}, which contains the hash values generated by the trapdoor hash function; in the verification algorithm, each received $(\bar{m}_j, \bar{r}_j)$, $j \in \{j_1, j_2, \ldots, j_l, n+1\}$, is first used to compute a hash value under the trapdoor hash function. The newly computed hash value is then compared with the corresponding hash value in AAI_{extra}; the ones that do not match indicate the location of authentication failure.

Against Lossy Channels. It appears not difficult to incorporate the techniques used to counter against lossy channels such as Pannetrat and Molva [2003], Krohn et al. [2004], and Lysyanskaya et al. [2004b] into our construction. What deserves mentioning here is that along the lossy channel in a server-proxy-user system, there are two consecutive segments: one from the multimedia server to the intermediary proxy, and the other from the proxy to end users. As a result, measures must be taken at both segments in an attempt to counter the lossy channel. In particular, the multimedia server enforces the adopted anti-lossy channel mechanism first and disseminates the content to the intermediary proxy; upon receiving the content, the proxy decodes and verifies the content, and then reinforces the mechanism and continues to transmit the content to end users.

Enforcing Data Confidentiality. Data confidentiality is another important security issue in the server-proxy-user multimedia delivery systems. In this context, data confidentiality refers to keeping data hidden from eavesdroppers en route during transmission. We next describe an encryption technique that integrates naturally into our authentication scheme.

We require that the key pair of the intermediary proxy be for public key encryption. Moreover, each user also has a key pair for public key encryption. Instead of sending out the multimedia content in the clear, the multimedia server generates a random session key, encrypts the content using a symmetric

encryption scheme under the session key, and encrypts the session key under the public key of the intermediary proxy. Then the server dispatches all the ciphertexts together with its signature σ and r, to the intermediary proxy. With the private key for transcoding, the intermediary proxy first decrypts to get the session key, and then decrypts to obtain the multimedia content using the session key. The proxy continues to transcode the multimedia content as usual. Finally, the proxy generates a new session key, encrypts the transcoded content using the new session key, and encrypts the session key under the user's public key. At the user side, the user proceeds with decryptions in a similar way as the intermediary proxy.

5. APPLICATION

Unlike generic data modality, actual multimedia content are rich in structures and have semantics, which can be explored to achieve better efficiency. In this section, we apply our construction to authenticating JPEG2000 code-streams and MPEG-4 video streams based on their respective semantics and data structures.

5.1 Authentication of JPEG2000 Code-Streams

5.1.1 *Fundamentals on JPEG2000.* We first briefly introduce the concepts related to the semantics of JPEG2000 code-streams [Taubman and Marcellin 2000; Deng et al. 2005, 2004].

Tile. A JPEG2000 image can be divided into rectangular nonoverlapping regions each of which is a *tile*. Tiles of an image are compressed independently, so it suffices for us to consider a single tile in the sequel.

Component. An image is comprised of one or more components. For example, an RGB image has three components, representing red, green, and blue color planes.

Resolution-increment and resolution. Given a component, an $(n_R - 1)$-level dyadic wavelet transform is performed. The first level of transform wavelet decomposes the component into four frequency subbands: LL_1 (horizontally lowpass and vertically lowpass), LH_1 (horizontally lowpass and vertically highpass), HL_1 (horizontally highpass and vertically lowpass) and HH_1 (horizontally highpass and vertically highpass). The second level of transform further decomposes LL_1 into another four subbands: LL_2, LH_2, HL_2, and HH_2. Eventually the $(n_R - 1)$ level of transform decomposes LL_{n_R-2} into LL_{n_R-1}, LH_{n_R-1}, HL_{n_R-1}, and HH_{n_R-1}. As a consequence, an $(n_R - 1)$-level wavelet transform ends up generating n_R sets of subbands, denoted as $R_0 = LL_{n_R-1}$, $R_1 = \{LH_{n_R-1}, HL_{n_R-1}, HH_{n_R-1}\}$, $\ldots$, $R_{n_R-1} = \{LH_1, HL_1, HH_1\}$. We refer to R_i as *resolution-increment* i. These n_R resolution-increments correspond to n_R *resolutions* or image sizes. In particular, the resolution 0 image is constructed from resolution-increment 0: R_0; the resolution 1 image is constructed from R_0 and R_1; the resolution $n_R - 1$ image is constructed from all the resolution-increments: $R_0, R_1, \ldots, R_{n_R-1}$. It should be noted that the resolution $n_R - 1$ image is the original image, and the resolution 0 image is the smallest thumbnail of the image.

Layer-increment and layer. Following the wavelet decomposition, wavelet coefficients are quantized and each quantized subband is partitioned into small rectangular blocks, referred to as code-blocks. Each code-block is independently entropy encoded to create a compressed bit-stream that is distributed across n_L quality layers. Layers determine quality or signal-to-noise ratio of the reconstructed image. Let L_0 denote the code-stream data needed to form a layer 0 image, and L_l be the additional data required to form a layer l image given $L_0, L_1, \ldots, L_{l-1}$, $l = 1, 2, \ldots, n_L - 1$. In other words, a layer l image is constructed by $\{L_0, L_1, \ldots, L_{l-1}, L_l\}$. L_l is referred to as layer-increment l. The layer $n_L - 1$ image is the original image given that the total number of layers of the image is n_L.

Precinct. To facilitate accessing certain portions such as ROI of an image (i.e., locality for accessing), JPEG2000 provides an intermediate space-frequency structure known as a *precinct*. In particular, a precinct is a collection of spatially contiguous code blocks from all subbands at a particular resolution. It is important to note that unlike the tile and code-block partitions, the precinct partition does not affect the transformation or coding of the sample data, and it helps in organizing compressed data within a code-stream.

Packet. Packet is the most fundamental building block in a JPEG2000 code-stream. A data packet comprises the compressed bit-stream from code blocks belonging to a specific component, resolution, layer, and precinct.

Progression order. Progressive display allows an image to be reconstructed with increasing pixel quality or resolution for different target devices. JPEG2000 supports progression in four dimensions: layer (L), resolution (R), precinct (P), and component (C). These dimensions of progression can be mixed and matched within a single code-stream. The JPEG2000 standard defines five progression orders: LRCP, RLCP, RPCL, PCRL, and CPRL. It is very important to know that a packet in a code-stream uniquely corresponds to a layer-increment l, resolution-increment r, component c, and precinct p.

5.1.2 *Method.* We apply the idea of the optimized scheme—using the Merkle hash tree—to the authentication of JPEG2000 code-streams, and our method is built upon the authentication schemes in Deng et al. [2005, 2004]. To keep our presentation neat and without losing generality, we consider a code-stream that has one title and one component. As such, the code-stream comprises a set of resolution-increments $\{R_r : r = 0, 1, \ldots, n_R - 1\}$, a set of layer-increments $\{L_l : l = 0, 1, 2, \ldots, n_L - 1\}$, and a set of precincts $\{P_p : p = 0, 1, 2, \ldots, n_P - 1\}$.

From this, JPEG2000 inherently supports different image sizes and image qualities controlled by resolutions and layers, respectively. This feature provides a natural way to downgrade JPEG2000 images based on resolutions and layers. We observe that in the context of server-proxy-user systems, a downgrading strategy based on resolutions or followed by layers may be the most likely strategy. Hence we only consider content downscaling based on resolutions. Generalization to other strategies is straightforward.

One of our main concerns is to minimize the amount of auxiliary authentication information *AAI* in case of content downscaling by the intermediary proxy. To achieve this, we should place the nodes that correspond to resolution-increments as high as possible along the Merkle hash tree. Figure 3 shows an example of organizing the Merkle hash tree following the order of resolutions, layers, and precincts. Recall that a packet uniquely corresponds to a resolution, a layer, and a precinct. So the path from the root to a leaf node in Figure 3 identifies a packet. For example, the leftmost path is specified by resolution-increment 0 (R_0), layer-increment 0 (L_0), and precinct 0 (P_0). Thus the value of the leaf node of this path is the hash value of the packet corresponding to R_0, L_0, and P_0. As an example, let us suppose the intermediary proxy downscales the image to the resolution 0 image, then *AAI* contains only the hash value represented by node R_1. It turns out that the Merkle hash tree in Figure 3 yields the smallest amount of auxiliary authentication information under the resolution-based downscaling strategy.

What remains to consider is how to accommodate transcoding operations of content alteration. This depends on the way the intermediary proxy is expected to modify an image. For example, if the intermediary proxy is expected to modify content involving all layers except layer 0, then the hash values corresponding to nodes L_1 (shaded nodes in Figure 3) are computed by applying the trapdoor hash function while all other nodes are computed using the standard hash function. This allows the intermediary

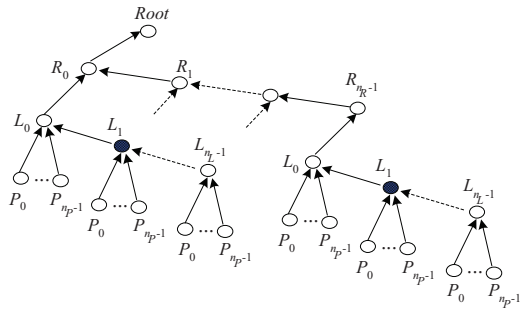

Fig. 3. Organizing the Merkle hash tree following the order of resolutions, layers, and precincts.

proxy to modify any number of the leaf nodes rooted at L_1s' while keeping the original hash values of L_1s unchanged (to find collisions). It is clear that to achieve the same objective, we can instead apply the trapdoor hash function to the leaf nodes of the subtrees rooted at L_1s, while using the standard hash function on all the remaining nodes. Note however that the trapdoor hash function is much more expensive than the standard hash function, thus the latter alternative is undesirable. For efficiency considerations, it is wise to avoid using the trapdoor hash function on individual leaf nodes, and it seems that in JPEG2000, content alteration often affects the data at higher levels, such as resolution and layer. As such, an important distinction in authentication of JPEG2000 code-streams from the construction for generic data modality is that the use of the trapdoor hash function is not necessarily at the level of the leaf nodes of the Merkle hash tree, and this clearly does not compromise security.

5.2 Authentication of MPEG-4 Video Streams

For ease of understanding, we first brief the structural semantics of MPEG4 video streams, and refer interested readers to [ISO/IEC14496-1:2001] for more details. An MPEG-4 video sequence or group (VS) comprises a series of video objects (VOs). Each VO is encoded in one or more video layers (VOLs). Each VOL contains a sequence of 2D representations of arbitrary shapes at different time intervals, which is referred to as a video object plane (VOP). The information contained in each layer corresponds to a given level of temporal and spatial resolutions, making scalable transmission and storage possible. Each VOP is divided into macroblocks (MB) of size 16×16, and each MB is encoded into 6 blocks, $B_1, \ldots, B_6$, of size 8×8 in the case of 4:2:0 format. Moreover, in an MPEG-4 stream, video objects such as foreground objects and background objects can be assigned different priorities, known as *visual_object_priority*, valued $1 \sim 7$ from lowest priority to highest; similarly, object layers have *visual_object_layer_priorities* indicating the differing importance of layers. The layer with the highest priority, called the base layer, contains the most important features of the video sequence, while each additional layer, known as an enhancement layer, serves to further enhance the quality of the base layer.

Clearly, MPEG-4 provides a natural way for transcoding operations such as content downscaling and content alteration according to video objects and video object layers in the context of server-proxy-user systems. Based on this, we organize, in Figure 4, a Merkle hash tree in order to authenticate an MPEG-4 video stream that contains n_O video object, each video object includes n_L video object layers, each video object layer has n_P video object planes, and each video object plan has n_M macroblocks. Content downscaling can be readily accomplished on the level of video objects or video object layers. For example, it is very easy for the intermediary proxy to drop some less important video objects while yielding a minimized amount of auxiliary authentication information *AAI*. Also, the intermediary proxy can easily perform transcoding operations of content alteration based on video objects and video objects layers, in which cases the hash values corresponding to the affected VOs or VOLs in the Merkle hash

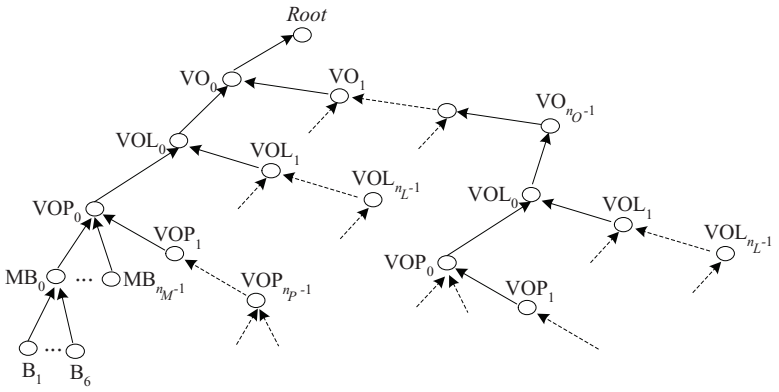

Fig. 4. The Merkle hash tree for authentication of MPEG-4 video streams.

Table I. Experiment Results

	Downscaling	1-Block Alteration	2-Block Alteration
Proxy side	0.42 ms	29.5 ms	59.6 ms
Server side	82.7 ms		
User side	70.1 ms		

tress should be computed using the trapdoor hash function while other nodes use the standard hash function.

6. EXPERIMENT RESULTS

To evaluate the performance of our construction, we implemented a prototype of the server-proxy-user architecture, which consists of three modules: a server module, a proxy module, and a user module. Implementation is done on a PC with an Intel P4 2.4Ghz processor, 1.00 GB RAM, and the source code is written in Microsoft C++. We instantiated the regular hash function and the digital signature scheme by SHA-1 and 1024-bit RSA, respectively. For the trapdoor hash function, we choose the prime number p to be 1024 bits and use 5 as the generator g. Note that to measure the worst-case performance, we did not make optimizations even where it is possible, for example, computation of $y^e g^\delta \mod p$ could be optimized to compute a single exponentiation by leveraging the techniques in Dimitrov et al. [2000].

We first tested our construction (the Merkle hash tree construction) on generic data content. In particular, 16 message blocks (each is 5KB) are organized into a Merkle hash tree, among which 2 blocks are trapdoor hashed and expected to be altered. We list the experimental results in Table I, where the digits are averaged over 500 runs. The "downscaling" column shows the result of experiments involving only content downscaling; the "1-block alteration" column is the result of experiments where only one block is altered, and the "2-block alteration" experiments involve altering both blocks. It can be seen that the time to compute a collision for the trapdoor hash function is approximately 29 ms in our implementation. We point out that the performance differences between the server and the user is due to the different time taken for RSA signing and verification.

We also applied our implementation to test JPEG2000 images. The tested images we use have 1 tile, 3 components, 7 resolution levels for each tile-component, 16 precincts for each resolution level, and 10 layers. The image size is about 200KB per image. We organize and hash images following the strategy in Figure 3 (8 L_1 nodes are trapdoor hashed). Our experimental results show that on the average, the server module can generate signed images in about 278 ms, generating up to 720 KB code-streams per second. The proxy module uses about 242 ms to compute collisions for a signed image,

which suggests that the intermediary proxy performing transcoding operations does not degrade the scalability offered by the server-proxy-user architecture. No experiments were performed on MPEG-4 video streams. However, authenticating a 700 KB per second data stream should suffice for many MPEG-4 applications. Better yet, much higher rates scan be supported in authenticating actual streams, because only a small portion of a stream is expected to be trapdoor-hashed. To summarize, we believe that our scheme is unlikely to have serious efficiency problems for practical use.

7. CONCLUSIONS

The main challenge in achieving end-to-end authentication in server-proxy-user multimedia delivery systems is that authorized transcoding operations performed by intermediary proxies will definitely invalidate the digital signature generated by the multimedia server if a standard digital signature is used. In this article, we systematically studied the authentication problem in this context: first, a formal model for the problem was specified; then, a concrete construction upon generic data formality was given, and its security was formally proven; finally, the generic construction was applied to the authentication of specific multimedia formats, JPEG2000 code-streams and MPEG-4 video streams. The experimental results showed that our construction is practical.

APPENDIX 1. PROOF OF THEOREM 1.

PROOF. We prove by contradiction, and the proof is based on Ateniese et al. [2005]. The basic idea is: suppose there exists an efficient adversary breaking the security property of our construction, then this adversary can be used to construct an efficient algorithm to break the underlying digital signature scheme, the trapdoor hash function, or the regular hash function. The details are sketched in the following.

Let χ be the intermediate value of a message m signed by the underlying signature scheme, $\sigma = S_{SK_S}(\chi)$. Consider an instance of the forging experiment in which $\mathcal{A}$ succeeds in computing a signature σ on a new message m such that $\sigma = S_{SK_S}(\chi)$. The instance must fall in at least one of the two cases:

Case 1. Every query m_i $\mathcal{A}$ made to the oracle $\mathcal{O}^{SK_S}$ resulted in signature $\sigma_i = S_{SK_S}(\chi_i)$ associated to the intermediate value χ_i, which is distinct from the value χ for the successful forgery $\sigma = S_{SK_S}(\chi)$.

Case 2. There is a query m_i to $\mathcal{O}^{SK_S}$ such that the response σ_i equates $\sigma = S_{SK_S}(\chi)$, but m_i is distinct from m.

Our proof distinguishes these two cases. In Case 1, we build an adversary $\mathcal{B}$, which breaks the underlying digital signature scheme. In its first phase, $\mathcal{B}$ generates a key pair (PK_P, SK_P) for the trapdoor hash function. It gives PK_P to the adversary $\mathcal{A}$, and uses SK_P with the collision-finding algorithm for the trapdoor hash function to emulate the oracle $\mathcal{O}^{SK_P}$. $\mathcal{B}$ also emulates the oracle for the regular hash function in a way stipulated in Bellare and Rogaway [1993]. In order to answer $\mathcal{A}$'s signature queries, $\mathcal{B}$ turns to its own signing oracle for the underlying digital signature scheme. When $\mathcal{A}$ finishes computing σ, $\mathcal{B}$ outputs the corresponding χ for its choice of target message, and the whole transcript of $\mathcal{A}$'s execution as its state after the first phase.

In its second phase, $\mathcal{B}$ just reads σ from the state information from the first phase, and terminates successfully whenever $\mathcal{A}$ succeeds, and the execution is an instance of case 1. $\mathcal{B}$'s execution time is $t_0 = t + Q_{TH}.t_{TH} + Q_H.t_H$, where t is the number of steps used by $\mathcal{A}$, Q_{TH} is the number of queries to the oracle for the trapdoor hash function, Q_H is the number of queries to the oracle for the regular hash function, t_{TH} is the maximum running time that $\mathcal{B}$ takes to compute a collision of the trapdoor hash function, and t_H is the maximum running time that $\mathcal{B}$ takes to compute the regular hash function.

In Case 2, we build an adversary $\mathcal{C}$ to the trapdoor hash function or the regular hash function, from $\mathcal{A}$. In its first phase, $\mathcal{C}$ generates a key pair (PK_S, SK_S) for the underlying digital signature scheme. It gives

PK_S to the adversary $\mathcal{A}$, and uses SK_S with the signing algorithm $S_{SK_S}(.)$ to emulate the signing oracle $\mathcal{O}^{SK_S}$. To answer $\mathcal{A}$'s transcoding queries, $\mathcal{C}$ turns to its own collision-finding oracle for the trapdoor hash function and the oracle for the regular hash function. When $\mathcal{A}$ finishes computing σ, $\mathcal{C}$ retrieves the value χ and compares it with the values χ_i that appear in $\mathcal{A}$'s transcript of queries to the signing oracle. Since we are in Case 2, there is at least one queried message m_i that differs from m, but χ_i equates χ. For simplicity, we assume no content removal is involved, and m and m_i only differ in items upon which either (a) the trapdoor hash function, or (b) the regular hash function is applied. In case (a), we have $th = TH_{PK_P}(m, r) = TH_{PK_P}(m_i, r_i)$, and $\mathcal{C}$ outputs (th, m_i, r_i) as its chosen value to seek collisions against; in case (b), we have $h = \mathcal{H}(m) = \mathcal{H}(m_i)$, and $\mathcal{C}$ outputs (h, m_i) to seek collisions against.

In its second phase, $\mathcal{C}$ just reads the values (m, th, r) or (m, h) from the transcript of $\mathcal{A}$ and outputs them. Therefore $\mathcal{C}$ succeeds whenever $\mathcal{A}$ succeeds and $\mathcal{A}$'s execution is an instance of Case 2. $\mathcal{C}$'s execution time is $t_1 = t + Q_S.t_{sign}$, where t is the number of steps used by $\mathcal{A}$, Q_S is the number of $\mathcal{A}$'s queries to the signing oracle, and t_{sign} is the maximum number of steps executed by the underlying signing algorithm, that $\mathcal{C}$ must perform to emulate the signing oracle. □

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments on this work.

REFERENCES

ATENIESE, G., CHOU, D. H., MEDEIROS, B., AND TSUDIK, G. 2005. Sanitizable signatures. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS'05)*. Springer Verlag, 159–177.

BELLARE, M. AND ROGAWAY, P. 1993. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS'93)*. 62–73.

BERGADANO, F., CAVAGNINO, D., AND CRISPO, B. 2000. Chained stream authentication. In *Proceedings of the Workshop on Selected Areas in Cryptography*. Springer Verlag, 144–157.

BHATTACHARJEE, S., CALVERT, K. L., AND ZEGURA, E. W. 1998. Self-organizing wide-area network caches. In *Proceedings of IEEE INFOCOM'98*. 600–608.

BJORK, N. AND CHRISTOPOULOS, C. 2000. Video transcoding for universal multimedia access. In *Proceedings of the ACM Workshops on Multimedia*. 75–79.

BRISCOE, B. 2000. Flames: Fast, loss-tolerant authentiation of multicast streams. Tech. rep., BT Research.

CARDELLINI, V., YU, P. S., AND HUANG, Y.-W. 2000. Collaborative proxy system for distributed Web content transcoding. In *Proceedings of the 9th International ACM Conference on Information and Knowledge Management (CIKM'00)*. 520–527.

CONTINI, S., LENSTRA, A. K., AND STEINFELD, R. 2006. VSH, an efficient and provable collision resistant hash function. In *Proceedings of the International Conference on Advances in Cryptology (Eurocrypt'06)*. Springer Verlag, 165–182.

DENG, R. H., MA, D., SHAO, W., AND WU, Y. 2005. Scalable trusted online dissemination of jpeg2000 images. *ACM Multimedia Syst. 11,* 1, 60–67.

DENG, R. H., WU, Y., AND MA, D. 2004. Securing JPEG 2000 code-streams. In *Security in the 21st Century*, 229–253.

DIMITROV, V. S., JULLIEN, G. A., AND MILLER, W. C. 2000. Complexity and fast algorithms for multi-exponentiations. *IEEE Trans. Comput. 49,* 2, 141–147.

FRIDRICH, J., GOLJAN, M., CHEN, Q., AND PATHAK, V. 2004. Lossless data embedding with file size preservation. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents (SPIE)*. 354–365.

FURHT, B. AND KIROVSKI, D. 2006. *Multimedia Encryption And Authentication Techniques And Applications*. CRC Press.

GENTRY, C., HEVIA, A., AND JAIN, R. 2005. End-to-end security in the presence of intelligent data adapting proxies: The case of authenticating transcoded streaming media. *IEEE J. Sel. Areas Comm. 23,* 2, 464–473.

GOLLE, P. AND MODADUGU, N. 2001. Authenticating streamed data in the presence of random packet loss. In *Proceedings of the Annual Symposium on Network and Distributed System Security Symposium (NDSS'01)*. 12–21.

GUO, H. AND GEORGANAS, N. D. 2002. Digital image watermarking for joint ownership. In *Proceedings of the ACM International Conference on Multimedia (MM'02)*. 362–371.

HALLE, M. W. AND KIKINIS, R. 2004. Flexible framework for medical multimedia. In *Proceedings of the ACM International Conference on Multimedia (MM'04)*. 768–775.

HUANG, J. L., CHEN, M. S., AND HUNG, H. P. 2004. A QOS-aware transcoding proxy using on-demand data broadcasting. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOMM'04)*. 2050–2059.

ISO/IEC14496-1. 2001. Information Technology - Coding of Audio-Visual Objects - Part 1: Systems; 14496-2:2003 Part 2: Visual.

JOHNSON, R., MOLNAR, D., SONG, D., AND WAGNER, D. 2002. Homomorphic signature. In *Proceedings of the RSA Security Conference Cryptographers Track (ACT-RSA)*, Lecture Notes in Computer Science, vol. 2271. Springer Verlag, 244–262.

KIROVSKI, D., MALVARN, H., AND YACOBI, Y. 2002. Multimedia content screening using a dual watermarking and fingerprinting system. In *Proceedings of the ACM International Conference on Multimedia (MM'02)*. 372–381.

KROHN, M. N., FREEDMAN, M. J., AND MAZIRES, D. 2004. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'04)*. 226–240.

LI, K. AND SHEN, H. 2005. Coordinated enroute multimedia object caching in transcoding proxies for tree networks. *ACM Trans. Multimedia Comput. Commun. Appl. 1,* 3, 289–314.

LI, T., ZHU, H., AND WU, Y. 2005. Multi-source stream authentication framework in case of composite MPEG-4 stream. In *Proceedings of the International Conference on Information and Communications Security (ICICS'05)*. Springer Verlag, 389–401.

LIBSIE, M. AND KOSCH, H. 2002. Content adaptation of multimedia delivery and indexing using MPEG-7. In *Proceedings of the ACM International Conference on Multimedia (MM'02)*. 644–646.

LIN, C.-Y. AND CHANG, S.-F. 2001. A robust image authentication method distinguishing JPEG compression from malicious manipulation. *IEEE Trans. Circ. Syst. Video Tech. 11,* 2, 153–168.

LIU, X. AND ESKICIOGLU, A. M. 2003. Selective encryption of multimedia content in distribution networks: Challenges and new directions. In *Proceedings of the IASTED International Conference on Communications, Internet, and Information Technology (CIIT'03)*. 527–533.

LYSYANSKAYA, A., MICALI, S., REYZIN, L., AND SHACHAM, H. 2004a. Sequential aggregate signatures from trapdoor permutations. In *Proceedings of the International Conference on Advances in Cryptology (Eurocrypt'04)*. Springer Verlag, 74–90.

LYSYANSKAYA, A., TAMASSIA, R., AND TRANDOPOULOS, N. 2004b. Multicast authentication in fully adversarial networks. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'04)*. Springer Verlag, 241–255.

MERKLE, R. C. 1989. A certified digital signature. In *Proceedings of the International Conference on Advances in Cryptology, (Crypto'89)*. Springer Verlag, 218–238.

MINER, S. AND STADDON, J. 2001. Graph-based authentication of digital streams. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'01)*. 232–246.

MIYAZAKI, K., HANAOKA, G., AND IMAI, H. 2006. Digitally signed document sanitizing scheme based on bilinear maps. In *Proceedings of the IACM Symposium on Information, Computer and Communications Security (ASIACCS'06)* 343–354.

MOHAMMED, S. AND FIAIDHI, J. 2005. Developing secure transcoding intermediary for SVG medical images within peer-to-peer ubiqitous environment. In *Proceedings of the IEEE 3rd Annual Communication Networks and Services Rsearch Conference (CNSR'05)*.

NACCACHE, D., POINTCHEVAL, D., AND STERN, J. 2001. Twin signatures: An alternative to the hash-and-sign paradigm. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'01)*. 20–27.

OOI, W. T. AND VAN RENESSE, R. 2001. Distributing media transformation over multiple media gateways. In *Proceedings of the ACM International Conference on Multimedia (MM'01)*. 159–168.

PANNETRAT, A. AND MOLVA, R. 2003. Efficient multicast packet authentication. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'03)*. Internet Society.

PARK, J. M., CHONG, E., AND SIEGEL, H. J. 2003. Efficient multicast packet authentication using erasure codes. *ACM Trans. Inform. Security 6,* 2, 258–285.

PARMANTO, B., SAPTONO, A., FERRYDIANSYAH, R., AND SUGIANTARA, I. W. 2005. Transcoding biomedical information resources for mobile handhelds. In *Proceedings of the 38th Hawaii International Conference on System Sciences*.

PENG, C., DENG, R. H., WU, Y., AND SHAO, W. 2003. A flexible and scalable authentication scheme for JPEG 2000 image codestreams. In *Proceedings of the ACM International Conference on Multimedia (MM'03)*. 433–441.

PERRIG, A. 2001. The BIBA one-time signature and broadcast authentication protocol. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS'01)*. 28–37.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, J. D. 2001. Efficient and secure source authentication for multicast. In *Proceedings of the Annual Symposium on Network and Distributed System Security Symposium(NDSS'01)*. Internet Society.

PERRIG, A., CANETTI, R., TYGAR, J. D., AND SONG, D. 2000. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'00)*. 56–73.

ROHATGI, P. 1999. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS'99)*. 93–100.

SCHLAUWEG, M., FROFROCK, D., ZEIBIC, B., AND MULLER, E. 2006. Dual watermarking for protection of rightful ownership and secure image authentication. In *Proceedings of the 4th ACM International Workshop on Contents Protection and Security (MCPS'06)*. 59–66.

SCHNEIDER, M. AND CHANG, S.-F. 1996. A robust content-based digital signature for image authentication. In *Proceedings of the International Conference on Image Processing (ICIP'96)*. 227–230.

SCHONBERG, D. AND KIROVSKI, D. 2004. Fingerprinting and forensic analysis of multimedia. In *Proceedings of the ACM International Conference on Multimedia (MM'04)*. 788–785.

SHEN, B., LEE, S. J., AND BASU, S. 2003. Streaming media caching with transcoding-enabled proxies. In Tech. rep, Hewlett Packard.

SHI, C. AND BHARGAVA, B. 1998. Light-weight MPEG video encryption algorithm. In *Proceedings of the ACM International Conference on Multimedia (MM'98)*. 55–61.

SMITH, J. R., MOHAN, R., AND LI, C. S. 1998. Content-based transcoding of images in the Internet. *Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*. 7–11

STEINFELD, R., BULL, L., AND ZHENG, Y. 2001. Content extraction signatures. In *Proceedings of the International Conference on Information Security and Cryptology (ICISC'01)*. Springer Verlag, 285–304.

SUZUKI, T., RAMZAN, Z., FUJIMOTO, H., GENTRY, C., NAKAYAM, T., AND JAIN, R. 2004. A system for end-to-end authentication of adaptive multimedia content. In *Proceedings of the IFIP Conference on Communications and Multimedia Security (CMS'04)*.

SWAMINATHAN, A., MAO, Y., AND WU, M. 2006. Robust and secure hashing for images. *IEEE Trans. Inform. Forensics Security 1,* 2, 215–230.

TAUBMAN, D. S. AND MARCELLIN, M. W. 2000. *JPEG2000 - Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers.

WAGNER, M. AND KELLERER, W. 2004. Web services selection for distributed composition of multimedia content. In *Proceedings of the ACM International Conference on Multimedia (MM'04)*. 104–107.

WU, Y. AND DENG, R. H. 2006. Scalable authentication of MPEG-4 streams. *IEEE Trans. Multimedia 8,* 1, 152–161.

YAMAOKA, S., SUN, T., TAMAI, M., YASUMOTO, K., SHIBATA, N., AND ITO, M. 2005. Resource-aware service composition for video multicast to heterogeneous mobile users. In *Proceedings of the ACM Workshop on Multimedia Service Composition (MSC'05)*. 37–46.

YEUNG, S. F., LUI, J. C., AND YAU, D. K. 2002. A case for a multi-key secure video proxy: theory, design, and implementation. In *Proceedings of the ACM International Conference on Multimedia (MM'02)*. 75–79.

YUUICHI, T., KAORI, T., TAKESHI, O., SHINJI, S., AND HIDEO, M. 2003. Asia: Information sharing system with derived content restriction management. *IEICE Trans. Commun. 428*, 1463–1475.

ZHENG, Y., HARDJONO, T., AND PIEPRZYK, J. 1991. Sibling intractable function families and their applications. In *Proceedings of the Annual International Conference on Advances in Cryptology (Asiacrypt'91)*. Springer Verlag, 124–138.