Authentication Schemes for Multimedia Streams: Quantitative Analysis and Comparison

MOHAMED HEFEEDA and KIANOOSH MOKHTARIAN
Simon Fraser University

With the rapid increase in the demand for multimedia services, securing the delivery of multimedia content has become an important issue. Accordingly, the problem of multimedia stream authentication has received considerable attention by previous research and various solutions have been proposed. However, these solutions have not been rigorously analyzed and contrasted to each other, and thus their relative suitability for different streaming environments is not clear. This article presents comprehensive analysis and comparison among different schemes proposed in the literature to authenticate multimedia streams. Authentication schemes for nonscalable and scalable multimedia streams are analyzed. To conduct this analysis, we define five important performance metrics, which are computation cost, communication overhead, receiver buffer size, delay, and tolerance to packet losses. We derive analytic formulas for these metrics for all considered authentication schemes to numerically analyze their performance. In addition, we implement all schemes in a simulator to study and compare their performance in different environments. The parameters for the simulator are carefully chosen to mimic realistic settings. We draw several conclusions on the advantages and disadvantages of each scheme. We extend our analysis to authentication techniques for scalable streams. We pay careful attention to the flexibility of scalable streams and analyze its impacts on the authentication schemes. Our analysis and comparison reveal the merits and shortcomings of each scheme, provide guidelines on choosing the most appropriate scheme for a given multimedia streaming application, and could stimulate designing new authentication schemes or improving existing ones. For example, our detailed analysis has led us to design a new authentication scheme that combines the best features of two previous schemes.

Categories and Subject Descriptors: H.1.1 [**Models and Principles**] Systems and Information Theory

General Terms: Performance
Additional Key Words and Phrases: Multimedia authentication, authentication schemes, scalable coding, multimedia security, secure streaming, multimedia streaming

ACM Reference Format:
Hefeeda, M. and Mokhtarian, K. 2010. Authentication schemes for multimedia streams: Quantitative analysis and comparison. ACM Trans. Multimedia Comput. Commun. Appl. 6, 1, Article 6 (February 2010), 24 pages.
DOI = 10.1145/1671954.1671960 http://doi.acm.org/10.1145/1671954.1671960

1. INTRODUCTION

Many market research reports indicate that the demand for various multimedia services, such as Internet streaming, video conferencing, and Internet Protocol Television (IPTV), has steadily been

Authors' addresses: M. Hefeeda (corresponding author), K. Mokhtarian, School of Computing Science, Simon Fraser University, 8888 University Drive, B. C. Canada V5A 1S6; email: mhefeeda@cs.sfu.ca.

DOI 10.1145/1671954.1671960 http://doi.acm.org/10.1145/1671954.1671960

increasing over the past few years and it is expected that this trend will even be stronger in the future [Insight 2006; RNCOS 2006; Global 2007]. With such a strong demand, multimedia services will become prevalent and many people will rely on them in different aspects of their daily lives, including work, education, and entertainment. This widespread adoption and reliance of people on multimedia services makes the secure delivery of multimedia content over open and generally insecure networks, for example, the Internet, an important and critical problem. Secure delivery, in this article, means ensuring the authenticity of the multimedia content such that any tampering with the data by an attacker can be detected by the receiver of the data. Attackers may tamper with the multimedia data by removing, inserting, or modifying portions of the data. Tampering attacks may be performed for commercial, political, or even personal purposes.

The goal of this article is to rigorously analyze and compare different schemes proposed in the literature to authenticate multimedia streams. Two types of multimedia streams are considered: nonscalable and scalable. We present and analyze authentication schemes for both types. A nonscalable stream is encoded using a traditional video coder as a single layer. That is, the whole single-layer stream is needed by the receiver to decode it, since the nonscalable stream is not partially decodable. Nonscalable streams provide high coding efficiency, but offer limited support for receivers with heterogeneous processing, screen, and bandwidth capacities. A scalable stream, on the other hand, is encoded using a scalable video coding technique such that portions of the stream can be extracted and decoded as valid substreams. The flexibility of decoding partial streams comes at the cost of decreased coding efficiency. In addition, this flexibility poses some challenges for the authentication scheme, because partial streams are legally truncated versions of the original streams (i.e., they are not tampered-with streams) and the scheme should be able to verify their authenticity.

We first analyze the main authentication schemes for nonscalable streams. To conduct this analysis, we define five important performance metrics, which are computation cost, communication overhead, receiver buffer size, delay, and tolerance to packet losses. Then, we derive analytic formulas for these metrics for all considered authentication schemes to numerically analyze their performance. In addition, we implement all authentication schemes in a simulator to study and compare their performance in different environments. The parameter values for the simulator are carefully chosen to mimic realistic settings. We highlight the advantages and disadvantages of each scheme, and draw several conclusions on their behavior. This detailed analysis and comparison lead us to propose a new authentication scheme that combines the best features of two previous schemes. In the second part of the article, we extend our analysis to authentication schemes for scalable streams. We pay careful attention to the flexibility of the scalable streams and analyze its impacts on the authentication schemes. Based on our analysis, we present a number of recommendations for choosing the appropriate authentications for various streaming applications.

1.1 Related Work and Article Organization

Because of its importance, the problem of multimedia stream authentication has received significant attention from academia and industry. Several schemes have been proposed to address this problem in different settings. However, no rigorous analysis and *quantitative* comparison of the different schemes have been done in the literature, to the best of our knowledge. Detailed analysis of various authentication schemes is needed in order to discover the merits and shortcomings of each scheme. Also side-by-side comparisons of authentication schemes along multiple performance metrics provide guidelines on choosing the most suitable scheme for a given multimedia streaming application, and offer insights for further research on the stream authentication problem.

A survey on authentication schemes for multicasting multimedia streams is given in Challal et al. [2004]. The authors classify authentication schemes according to the core techniques underlying them, for example, symmetric or asymmetric cryptography, partially sharing secrets with receivers, relying on time synchronization, replicating hash values, and signature amortization. The authors also provide a qualitative comparison among the schemes. However, unlike our work, the work in Challal et al. [2004] does not provide quantitative analysis and detailed comparison of the authentication schemes in realistic environments with actual values for the parameters used in the schemes and in the networks they are used in. In addition, the survey in Challal et al. [2004] is relatively old and does not cover a number of authentication schemes that we consider in this article. Furthermore, the work in Challal et al. [2004] does not discuss or analyze authentication schemes for multimedia streams encoded using scalable coders, which have been receiving increasing attention recently because of their ability to support a wide range of heterogeneous clients [Schwarz et al. 2007]. Previous authentication techniques for scalable JPEG 2000 images [Deng et al. 2005; Grosbois et al. 2001; Peng et al. 2003] are surveyed in Zhu et al. [2004]. However, we are not aware of previous works that analyze authentication of scalable video streams.

The rest of this article is organized as follows. In Section 2, we define the performance metrics and notations used in our analysis. In Section 3, we present and analyze authentication schemes for nonscalable streams. For each scheme, we provide a brief overview of the scheme followed by the analysis of the different performance metrics. In Section 4, we conduct numerical analysis of the equations derived in Section 3. We also present our simulation analysis and summarize our findings in this section. We present and analyze authentication schemes for scalable multimedia streams in Section 5. We conclude the article in Section 6.

2. NOTATIONS AND PERFORMANCE METRICS

To rigorously evaluate various multimedia authentication schemes, we define five performance metrics that cover all angles of the authentication problem, and we analyze these metrics in different environments. The performance metrics are as follows.

—*Computation Cost*. It is the CPU time needed to verify the authentication information by the receiver. Evaluating the computation cost is important especially if the receiver has a limited processing capacity, for example, a PDA or cell phone. Note that in streaming applications, the verification process of an incoming multimedia stream is invoked periodically and in real time. Thus, if its computation cost is high, some receivers may not be able to support it.

—*Communication Overhead*. It is the additional number of bytes that the authentication scheme needs to transfer over the communication channel to the receiver in order to enable it to verify the authenticity of the received multimedia stream.

—*Tolerance to Packet Losses*. Multimedia streams are typically transmitted over the Internet or lossy wireless channels, where some packets may get lost. Due to the dependency that the authentication scheme imposes among packets, packet loss may affect verifiability of some packets that are successfully received. Thus, the robustness of the authentication scheme to packet losses is important to analyze for different packet loss ratios. We quantify this robustness as the percentage of the received packets that can be verified in presence of packet losses, and we call it the verification rate.

—*Receiver Buffer Size*. Some authentication schemes require the receiver to buffer a certain amount of data before it can start verifying the stream. Quantifying the required buffer size specifies the minimum memory requirements, which is especially important for limited-capability receiver devices. Besides, the required receiver buffer determines the amount of delay at receiver side, which is included in the delay metric given next.

—*Delay Imposed by the Authentication Process*. Since most of the schemes designate one digital signature for a block of packets, they require the sender/receiver (or both) to wait for generation/reception

Table I. Parameters Used in This Article and Their Values

Parameter	Value	Description
α	30 pkt/sec	Packet rate.
l	1400 bytes	Packet size.
n	128 pkts	Block size. It is 128 packets as long as it is not one of the variable parameters.
N	1,000 to 1,000,000	Number of blocks transmitted.
ρ	0 to 0.5	Packet loss ratio.
n_{enough}	$0.8n$	Number of packets of a block that suffice for verification (Section 3.5).
t_{sig}	500 ms	Time to verify a digital signature (1024-bit RSA with public exponent = 65537).
t_{hash}	0.1 ms	Time to compute a hash over a 512-bit (64-byte) block.
s_{sig}	128	Signature size in bytes (1024-bit RSA).
s_{hash}	20	Hash size in bytes (SHA-1).
n_{rows}	32	Number of rows in Butterfly graph (see Section 3.3).
s	0, 0.25, 0.5	An input to eSAIDA (see Section 3.5).
n_{sig}	Augmented chain: 8, Butterfly graph: searched	Number of signature replications in the block. It is set to 1/16 n (i.e., 8) for Augmented Chain. For Butterfly Graph, it is obtained by a local search for best verification rate.
p, a	searched	Inputs to Augmented chain (see Section 3.2). They are obtained by a local search for best verification rate.

of a certain amount of data before being able to transmit/verify it. This delay, which is the sum of sender-side and receiver-side delays, specifies whether or not the authentication information can be produced or verified online. For example, a delay beyond a few seconds is not suitable for live streaming. Note that we consider the delay imposed by the authentication process only; delays caused by the transmission through the networks or by the media encoding/decoding process are not accounted.

The preceding metrics are analyzed under different scenarios. For example, we consider a wide range of packet loss rates and using two common loss models: bursty and random. Several other parameters are used in the analysis, such as packet rate α, packet size l, and block size n. For quick reference, we list all parameters used in this article and their notations in Table I. We also mention the range of values used for each parameter. In the simulation section, we discuss why we use these values.

3. AUTHENTICATION SCHEMES FOR NONSCALABLE STREAMS

Several schemes have been proposed for authenticating nonscalable multimedia streams. A multimedia stream is nonscalable if it is encoded as a single layer and only the complete layer is decodable [Sullivan and Wiegand 2005]. We study and analyze the most important schemes in the literature. A naive solution for authenticating such stream may be to sign every packet. This clearly does not work in practice due to its high computational cost. Accordingly, to amortize this cost, authentication schemes often divide a stream into blocks of n packets, and designate one digital signature for each block.[1] In each of the following subsections (Sections 3.1 through 3.7), we briefly describe the main idea of an authentication scheme and we analyze it using the performance metrics defined in Section 2. We also mention the authentication schemes that are not analyzed in this article and why we do not consider them in Section 3.8. We use the parameters listed in Table I in the analyses.

[1]Note that we only consider authentication schemes that rely on digital signatures for assuring authenticity. That is, we do not consider schemes that only rely on Message Authentication Codes (MAC), which requires all parties of the streaming setting to be fully trusted.

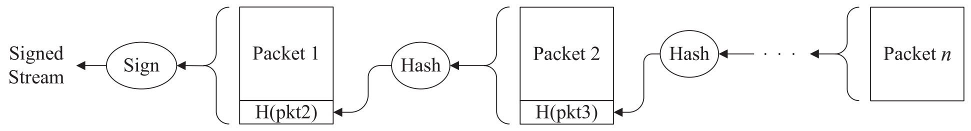

Fig. 1. Stream authentication using hash chaining.

3.1 Hash Chaining

Hash chaining [Gennaro and Rohatgi 1997] is one of the simplest techniques to authenticate multimedia streams. Figure 1 illustrates its basic idea. Packets of the stream are divided into blocks, each of size n packets. Then, the hash of each packet is attached to its previous packet, and the first packet of each block is signed. Due to the one-way property of the hash function, the signature authenticates the whole block.

Analysis of hash chaining is straightforward. For a block of n packets, hash chaining computes n hash values and verifies one digital signature.[2] Therefore, the computation cost to verify a block is $t_{sig} + n\lceil l/64 \rceil t_{hash}$ seconds. The communication overhead is $s_{sig}/n + s_{hash}$ bytes per packet. Hash chaining does not tolerate any packet losses. There is no receiver buffer requirement for this scheme, as packets can be verified as they arrive after receiving the first packet with the signature. The sender, however, needs to wait for n packets to be generated, because hash chaining starts at the last packet in the block. Thus, with a packet generation rate of α, the total delay is n/α.

To enable the hash chaining scheme to tolerate packet losses, the hash value of a packet is replicated and attached to multiple packets. According to the way hashes are replicated in packets, a block of packets can be modeled as a Directed Acyclic Graph (DAG), whose nodes represent data packets and each directed edge from node A to node B indicates that the hash of packet A is attached to packet B, that is, if packet B is verified then packet A can also be verified. In this DAG, a packet is verifiable if there is a path from its corresponding node to the signature node. When loss occurs among packets of a block, some nodes of the DAG and their associated edges are removed, which may threaten the verifiability of some of the received packets. The simple hash chaining described before can be viewed as a linear DAG with n nodes and $n-1$ edges, where the first node carries the signature. The following two subsections present two authentication methods that improve the robustness of a linear DAG to packet losses.

3.2 Augmented Hash Chaining

In the augmented hash chaining scheme [Golle and Modadugu 2001], the authentication DAG is constructed as follows. First, the hash of packet p_i is attached to packets p_{i+1} and p_{i+a}, where a is an integer parameter that affects resistance against bursty losses as well as receiver delay and buffer. The last packet is designated as the signature packet. Then, $p-1$ additional packets (p is an input to the algorithm) as well as their relevant edges are inserted between each two packets of this chain to make it an *augmented chain*. Two methods are proposed for this insertion, which have equal resistance to bursty losses. The first method attaches the hash of each new packet to the packet preceding it and to the packet from the original chain succeeding it. Thus, the number of hashes carried by packets of the original chain grows linearly with p, while the average number of hashes per packet is two. The second method is more complex and follows a recursive structure to keep the degree of each node equal to two; we consider the second structure in our analyses. Since the delivery of the signature packet is

[2]Readers that are not familiar with secure hashing, digital signatures, and other basic cryptographic functions are referred to Menezes et al. [1996] for detailed explanations.

vital, the scheme sends it multiple (n_{sig}) times within a block of packets. The packet loss tolerance of this technique depends on the loss model, that is, it depends on the loss rate and loss pattern (random or bursty). We analyze this tolerance using simulation in Section 4.

Computations needed to verify a block in augmented hash chaining take $t_{sig} + n\lceil l/64\rceil t_{hash}$ seconds, and the communication overhead is $n_{sig}s_{sig}/n+2s_{hash}$ bytes per packet. The receiver has to buffer a whole block, thus the receiver delay and buffer size are n/α seconds and n packets, respectively. Moreover, the sender needs to buffer p packets before transmission, which makes the sender delay p/α seconds. Since p is small compared to n, we consider the total delay to be n/α seconds.

3.3 Butterfly Hash Chaining

Zhang et al. [2005] proposed to use butterfly graphs to construct the authentication DAG. Assuming the number of packets of a block is $n = n_{rows}(\log_2 n_{rows} + 1)$, the nodes of the authentication DAG are arranged into $\log_2 n_{rows} + 1$ columns of the same length n_{rows}. Each node in a column is linked to two other nodes of the previous column, according to the column it belongs to. Nodes of the first column are all linked to the signature packet. These butterfly graphs, however, do not work for arbitrary number of packets, and make the size of the signature packet grow almost in proportion to the block size. To mitigate these limitations, the authors later extended their work to utilize a generalized butterfly graph [Zhishou et al. 2007]. This graph is made more flexibly such that the number of rows n_{rows} is set independently of n and is taken as an input. Then, nodes are arranged into $\lceil n/n_{rows}\rceil$ columns, where the last column does not necessarily consist of n_{rows} nodes. The way the nodes are linked to each other is similar to the previous butterfly graph. We consider the generalized version of the butterfly graph scheme in our analyses.

The computation cost of the butterfly authentication is the same as the augmented hash chaining: $t_{sig} + n\lceil l/64\rceil t_{hash}$ seconds to verify a block of n packets. Denoting the number of rows in the butterfly graph by n_{rows}, the communication overhead of this scheme is equal to $n_{sig}(s_{sig}+n_{rows}s_{hash})/n+s_{hash}(2n-n_{rows})/n$ bytes per packet. According to losses, receivers need to buffer packets untill a copy of the signature arrives. In the worst case they may need to have a buffer of up to n packets, though it is unlikely. Thus, for the total delay, we neglect the receiver delay when summing it with the sender of n/α seconds, which makes a total delay of n/α seconds. However, the receiver buffer required cannot be neglected even it fills infrequently. Similar to the augmented hash chaining, the loss tolerance of this schemes depends on the packet loss model, which we evaluate in the simulation section.

3.4 Tree Chaining

Wong and Lam [1999] proposed the use of Merkle hash trees [Merkle 1989] for stream authentication. In their scheme, one signature is designated for each block of packets. At the sender side, a balanced binary Merkle hash tree is built over packets of each block. Leaves of this tree are hashes of packets, and each interior node represents the digest of concatenation of its children. The root of this tree is then signed. Due to the collision-free property of the hash function, the whole set of leaf packets is authenticated if authenticity of the root of the tree is successfully verified. Each packet is individually verifiable by traversing and partially reconstructing the tree from the bottom (the leaf node corresponding to the given packet) to top (the root) and verifying the root digest using the given signature. For this procedure, only siblings of the nodes on the path are needed. Therefore, in this scheme, each packet carries the block signature, its location in the block, and the set of siblings on the path from itself to the root. This makes each packet individually verifiable.

The computations needed for verifying a block consist of one signature verification, $n\lceil l/64\rceil$ hash computations over packets, and $(\lceil n\log_2 n\rceil - n)\lceil 2s_{hash}/64\rceil$ hash computations interior to the tree, which in total takes $t_{sig} + t_{hash}(n\lceil l/64\rceil + (\lceil n\log_2 n\rceil - n)\lceil 2s_{hash}/64\rceil)$ seconds for a block. The communication

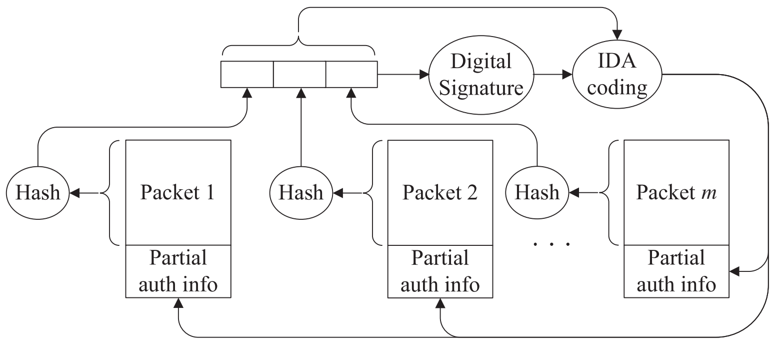

Fig. 2. Stream authentication using SAIDA.

overhead of this scheme is equal to $s_{sig} + \lceil \log_2 n \rceil s_{hash}$ bytes per packet. In tree chaining, there is no need to buffer any packet, thus a packet can be verified once it arrives. The total delay imposed by tree chaining, which consists of the sender delay only, is that of generating a block: n/α seconds. Moreover, loss resilience is always 100% given that a packet is either arrived or lost atomically.

3.5 SAIDA and eSAIDA

Park et al. [2003] presented SAIDA (Signature Amortization using Information Dispersal Algorithm) for stream authentication. As shown in Figure 2, SAIDA divides the stream into blocks of n packets. Then, it hashes each packet and concatenates the hash values. Let us denote the result of this concatenation by $H = h(p_1)||h(p_2)|| \cdots ||h(p_n)$, and the number of packets that are expected to be received out of a block of n packets by n_{enough}, that is, $n_{enough} = (1 - \rho)n$. H along with a signature on $h(H)$ is divided into n_{enough} ($n_{enough} \leq n$) pieces, IDA-coded[3] into n pieces, and split over all the n packets of the block. Any n_{enough} pieces suffice to reconstruct the hashes and the signature to verify authenticity of the entire block. Note that the signature itself is sufficient for authenticating the whole block if no loss occurs, but the concatenation of packet hashes is also IDA-coded and carried by packets so that the block is still verifiable if some packets are lost.

Computations needed by SAIDA to verify a block are n hash computations over packets, one hash over the concatenation of packet hashes, one signature verification, and one IDA-decoding. We ignore the cost of IDA-coding, because there are efficient algorithms for erasure correction, such as Tornado codes that use only XOR operations and operate in linear time of the block size, which can replace IDA-coding in SAIDA. Hence, the time it takes for a receiver to verify a block is $t_{sig} + (n\lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil) t_{hash}$. The communication overhead of SAIDA depends on the parameters of the IDA algorithm (or any other FEC technique used instead), and is equal to $(s_{sig} + n s_{hash})/n_{enough}$ bytes per packet. The receiver needs to buffer at least n_{enough} packets, which typically is a significant fraction of n. Thus the receiver delay can be considered n/α, which results in a total delay of $2n/\alpha$ seconds when summed to the sender delay of n/α.

As an enhancement on SAIDA, Park and Cho presented eSAIDA [Park and Cho 2004]. In eSAIDA, one hash is designated for each pair of adjacent packets, rather than one for each packet as in SAIDA. This reduces the overhead, but will cause a packet to be unverifiable if its couple is not received. Thus, a packet in a block may also contain the hash value of its couple. The fraction of packets containing

[3]See Rabin's Information Dispersal Algorithm (IDA) [Rabin 1989], which belongs to a broader set of erasure codes called Forward Error Correction (FEC) codes.

their couple's hash is parameterized by s ($0 \leq s < 1$) as an input, which governs a trade-off between successful verification rate and communication overhead. Computations needed by eSAIDA per each block are $(1 + s)n/2$ hash computations over packets, one hash over the concatenation of hashes of packet pairs, one signature verification, and one IDA-decoding that we neglect. Thus, the time it takes for eSAIDA to verify a block is $t_{sig} + (\lceil l/64 \rceil (1+s)n/2 + \lceil s_{hash} n/128 \rceil) t_{hash}$. The communication overhead of eSAIDA is $(s_{sig} + s_{hash} n/2)/n_{enough} + s_{hash} s$ bytes per block. The receiver buffer size and the total delay in eSAIDA are similar to those in SAIDA.

3.6 cSAIDA

Pannetrat and Molva [2003] developed another improvement of SAIDA, which we call cSAIDA because it significantly reduces the communication overhead of SAIDA. Recall that in SAIDA, the concatenation of the packet hashes (H) along with a signature on $h(H)$ are FEC-coded (using the IDA algorithm [Rabin 1989]) and distributed among the n packets of the block. However, a considerable fraction of these packet hashes can be computed from the received packets. Thus, there is no need for the whole H to be transmitted. To achieve this, cSAIDA uses FEC coding twice as follows. First, a systematic erasure code is employed to encode H. A systematic erasure code encodes data pieces $D_1, D_2, \ldots, D_n$ into m ($m \geq n$) pieces $D'_1, D'_2, \ldots, D'_m$ such that any subset of n pieces are sufficient for reconstructing the original data and the first n pieces of the encoded result are equal to the original data. That is, $D_i = D'_i$ ($1 \leq i \leq n$). In this case, the extra redundancy pieces $D'_{n+1}, \ldots, D'_m$ are called parity check pieces. Denoting the expected loss rate by ρ ($0 \leq \rho < 1$), in cSAIDA, the n pieces of H are systematically FEC-coded into $\lceil n + \rho n \rceil$ pieces $H'_1, H'_2, \ldots, H'_{\lceil n+\rho n \rceil}$. Then, only parity pieces $H'_{n+1}, \ldots, H'_{\lceil n+\rho n \rceil}$ and a signature on $h(H)$ are concatenated, divided into $\lfloor n(1-\rho) \rfloor$ pieces, and FEC-coded again into n pieces to be attached to all the n packets of the block. At the receiver side, if $\lfloor n(1-\rho) \rfloor$ (i.e., n_{enough}) packets are successfully received, then $\lfloor n(1-\rho) \rfloor$ of the hash values, the signature on $h(H)$, and the parity pieces $H'_{n+1}, \ldots, H'_{\lceil n+\rho n \rceil}$ can all be successfully retrieved. Thus, H can be reconstructed in order to verify the whole block using the signature.

Computations needed by cSAIDA to verify a block are equal to those of SAIDA, plus one extra FEC-decoding. Since we ignore the cost of FEC-decoding, the time it takes cSAIDA to verify a block is $t_{sig} + (n \lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil) t_{hash}$ seconds. The communication overhead of cSAIDA is $(s_{sig} + (n - n_{enough}) s_{hash})/n_{enough}$ bytes per packet. The total delay and receiver buffer size of cSAIDA are similar to those of SIADA and eSAIDA.

3.7 TFDP

Habib et al. [2005] presented TFDP (Tree-based Forward Digest Protocol) for offline P2P streaming, that is, distribution of already-recorded media files. Similar to SAIDA, packets are hashed, and packet hashes are concatenated and hashed again to form the digest of the block. Unlike SAIDA, only one signature is generated for the whole stream, because the entire file being streamed is given initially. Similar to tree chaining [Wong and Lam 1999], a Merkle hash tree [Merkle 1989] is built over blocks of the stream, whose leaves are block digests. The root of this tree is then signed. At the beginning of the streaming session, the client receives the signed root of the tree along with a list of senders. The client asks one of the senders for information needed to verify a number (x) of blocks, which includes hashes of packets of these blocks (FEC-coded), digests of the x blocks, and auxiliary digests in the tree needed for reconstructing and verifying the root digest. Having received the digests, the client checks their genuineness by recalculating the root hash. Once verified, they can be used for verifying authenticity of the x data blocks, one by one once they arrive. The client repeats the same procedure for the next sets of blocks. Therefore, the communication overhead is amortized over a number of blocks.

Computations needed by TFDP for each block (assuming $x = 1$ for simplicity) are n hash computations over packets, one hash over the concatenation of packet hashes, and $\lceil \log_2 N \rceil$ hashes corresponding to nodes interior to the tree, which takes $(n\lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil + \lceil \log_2 N \rceil \lceil 2s_{hash}/64 \rceil) t_{hash}$ seconds. Thus, compared to computations of other schemes, which all include one signature per block, TFDP's computations are much less expensive. The communication overhead of TFDP is at most equal to $s_{hash}(n/n_{enough} + (1 + \log_2 N)/n)$ bytes per packet, where N denotes the total number of blocks in the file. This worst-case communication overhead occurs when the clients requests one block, that is, $x = 1$. In TFDP, a receiver needs to buffer at most n packets. In addition, since TFDP works for offline streams only, the delay is not relevant.

3.8 Other Authentication Schemes

Perrig et al. [2000] proposed TESLA (Timed Efficient Stream Loss-tolerant Authentication) and EMSS (Efficient Multi-chained Stream Signature) to authenticate online streams in lossy networks. The core idea of TESLA is to take advantage of the online nature of the stream. That is, after a packet is received, any forgery on the content of that packet is of no danger. TESLA and a later improvement on it [Perrig et al. 2001] are very interesting techniques for authentication of online multicast streams. However, because they depend on the online nature of the stream and on a time synchronization (though loose) between the sender and receivers, they are not applicable to other streaming scenarios, such as video on demand, where receivers may be receiving different parts of the stream at the same time, or to P2P streaming, where delay constraining is extremely difficult. Therefore, TESLA cannot be a solution for the general stream authentication problem. The other method presented in Perrig et al. [2000], EMSS, does not depend on the online nature of the stream, which comes at the cost of more expensive computations. For a block of packets, the hash of each packet is put in multiple later packets of the block, and a signature is put in the last packet of the block. The signature suffices for verifying the entire block of packets. EMSS does not clearly specify how to designate hash links. It is one of the earliest techniques based on hash replication, and is outperformed by other schemes summarized in this section in terms of loss tolerability with less overheads. Thus, we do not include EMSS in our analysis.

The authentication schemes presented thus far do not use or depend on the characteristics of the video stream. There are other schemes that do use these characteristics in the authentication process. These are usually called *content-based* authentication schemes. Conducting a quantitative analysis of these schemes is difficult, because of the dependence on the video characteristics which are quite diverse and varying. Nonetheless, to make this article comprehensive, we summarize a few of such schemes in this category.

Liang et al. [2007] proposed a method for extracting a content-based feature vector from video. The goal is to inexpensively extract the feature vector such that it is robust against transcoding operations. The feature vector is then embedded back into the video as a watermark. Sun et al. [2003] extended an earlier work of theirs on image authentication [Sun et al. 2002] for video authentication with robustness against transcoding. In Sun et al. [2002], some content-based invariant features of an image are extracted, and the feature vector is FEC-coded. The resulting codeword can be partitioned into two parts: the feature vector itself and parity check bits. Only the parity check bits are taken as a kind of Message Authentication Code (MAC) for the image and are then embedded back into the image as a watermark. Furthermore, the whole FEC codeword (extracted feature vector and its parity check bits) is hashed and then signed to form the signature of the image. The signature can either be attached to the image or again be embedded back into the image as another watermark. At the receiver side, to verify the authenticity, the feature vector is extracted from the content, which together with the watermark (parity check bits) can reconstruct the FEC codeword whose signed hash is received and ready for verification. Feature extraction and watermark embedding are done in the transform domain, since transcoding operations are usually performed in that domain. Robustness against requantization is achieved by following the approach taken in an earlier work [Lin and Chang 2000].

Another instance of content-based approaches is the scheme presented by Atrey et al. [2007] for video authentication based on Shamir's secret sharing [Shamir 1979]. They divide a video into shots, and then extract a number of key frames from each shot. They define three levels of authentication and create one authentication frame for: (i) each sequence of frames between two successive key frames, (ii) each shot, and (iii) the whole video.

For detecting forgery of offline (recorded) videos, Wang and Farid [2006] approached authentication of MPEG-coded video by detecting the effects of double MPEG compression. Recall the video coding process where each Group of Pictures (GoP) consists of an I-frame followed by a number of P- and B-frames. When a sequence of frames is removed from (or inserted to) a video and the video is encoded again, the type (I, P, or B) of some frames could change. Accordingly, two frames belonging to a GoP in the new video might belong to two different GoPs in the old one. Since the retrieved content of P- and B-frames is mainly based on the motion vectors, there is a strong correlation between frames belonging to the same GoP. However, when some frames are in different GoPs in the old video and in the same GoP in the new one, this correlation becomes weaker, and the motion error is increased; this increased motion error is observed periodically in the video sequence. In addition to migration between GoPs, there is another effect when a video is compressed twice. For the I-frames that still remain I-frame in the new coded video, the quantization factor would be different if the old and new videos are encoded at different bitrates. Such double quantization results in a detectable statistical pattern that occurs periodically in the histogram of DCT coefficients. Taking advantage of these two behaviors, a forgery can be detected.

4. EVALUATION OF AUTHENTICATION SCHEMES FOR NONSCALABLE STREAMS

In this section, we compare nonscalable stream authentication techniques summarized in Section 3. We first conduct a numerical analysis of the computation cost and communication overhead of the schemes. Then, we analyze their tolerance to packet losses using simulation under different loss models. Then, a summary of our findings is presented. Finally, we propose to combine the best features of two authentication schemes to design a more efficient one.

4.1 Numerical Analysis

We present in Table II a summary of the analysis of all authentication schemes presented in the previous section. Each row corresponds to one authentication scheme, and the four columns represent four of the five performance metrics defined in Section 2. To shed some light on the performance of the different authentication schemes, we numerically analyze their computation cost and communication overhead as the number of packets in the group n varies. n is the most important parameter that impacts the performance of the authentication schemes. To conduct this analysis, we choose realistic values for other parameters as summarized in Table I and discussed next.

Choosing values of the parameters for simulation. We first choose the values of packet size and packet sending rate. To improve the performance of video streaming applications over the Internet, it is usually preferred to fit each application data unit in an IP packet, which should be smaller than the Maximum Transmission Unit (MTU) [Wenger et al. 2005]. Assuming a video encoding rate of 320 kbps and an MTU of 1,500 bytes, the data in a packet should be roughly 1,400 bytes. This takes into account the RTP/UDP/IP headers and authentication information attached to each packet. Thus, the packet rate (α) is equal to $\alpha = \frac{320 \text{ kbps}}{1400 \text{ bytes}} \simeq 30$ packets per second.

Table II. Summary of the Analysis of Authentication Schemes for Nonscalable Multimedia Streams

	Computation cost: time to verify a block (sec)	Communication overhead (bytes per packet)	Total delay (sec)	Receiver buff size (pkts)
Hash chaining	$t_{sig} + t_{hash} n \lceil l/64 \rceil$	$\frac{s_{sig}}{n} + s_{hash}$	n/α	1
Augmented chain	$t_{sig} + t_{hash} n \lceil l/64 \rceil$	$\frac{n_{sig} s_{sig}}{n} + 2s_{hash}$	n/α	n
Butterfly chaining	$t_{sig} + t_{hash} n \lceil l/64 \rceil$	$\frac{n_{sig}(s_{sig} + n_{rows} s_{hash})}{n} + \frac{s_{hash}(2n - n_{rows})}{n}$	n/α	n
Tree chaining	$t_{sig} + t_{hash} \Big(n \lceil l/64 \rceil + (\lceil n \log_2 n \rceil - n) \lceil 2s_{hash}/64 \rceil \Big)$	$s_{sig} + \lceil \log_2 n \rceil s_{hash}$	n/α	1
SAIDA	$t_{sig} + t_{hash} \Big(n \lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil \Big)$	$\frac{s_{sig} + n s_{hash}}{n_{enough}}$	$2n/\alpha$	n
eSAIDA	$t_{sig} + t_{hash} \Big(\lceil l/64 \rceil (1+s) n/2 + \lceil s_{hash} n/128 \rceil \Big)$	$\frac{s_{sig} + s_{hash} n/2}{n_{enough}} + s_{hash} s$	$2n/\alpha$	n
cSAIDA	$t_{sig} + t_{hash} \Big(n \lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil \Big)$	$\frac{s_{sig} + (n - n_{enough}) s_{hash}}{n_{enough}}$	$2n/\alpha$	n
TFDP	$t_{hash} \Big(n \lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil + \lceil \log_2 N \rceil \lceil 2s_{hash}/64 \rceil \Big)$	$s_{hash} (\frac{n}{n_{enough}} + \frac{1}{n} + \frac{\log_2 (N/x)}{nx})$	—	n

Next, we estimate the computation costs of the digital signature and hashing operations. Digital signature operations are often very costly, because they involve modular multiplication of very large numbers. Since we assume that the signer is powerful enough, RSA [Rivest et al. 1978] is an appropriate choice as the digital signature scheme, because its verification can be done efficiently when the public key is chosen properly, for example, a value of 65537 ($2^{16} + 1$) for the public exponent. The size of a 1024-bit RSA signature is $s_{sig} = 128$ bytes. t_{sig} in Table I denotes the time it takes to verify a 1024-bit RSA signature with such exponent. That is estimated for a typical limited-capability device, by using a small fraction (5% to 10%) of its CPU time. It is experimented in Argyroudis et al. [2004] that 1024-bit RSA verification when the public exponent is 65537 takes about 5 milliseconds on an iPAQ H3630 with a 206 MHz StrongARM processor, 32MB of RAM, and running Windows CE Pocket PC 2002. A similar experiment [Tillich and Groschdl 2004] measures 1024-bit RSA verification time on a number of J2ME-enabled mobile devices and reports that the time taken ranges from a few to more than a hundred milliseconds. Since the authentication scheme should not take more than a small fraction of CPU time, for example, 5% to 10%, and considering a safety margin, we took the value $t_{sig} = 500$ millisecond in Table I. For the hashing algorithm, SHA-1 [NIST 1993] and MD5 [Rivest 1992] are two popular one-way hash functions, both of which operate on blocks of 512 bits. MD5 has higher performance and smaller digest size, but some successful cryptanalyses have been done on MD5 and algorithms have been proposed for finding collisions [Klima 2005, 2006]. Although these cryptanalyses on MD5 are far from being practical for breaking a system in real time, we chose SHA-1 as the hash algorithm for our evaluations. The digest size s_{hash} for SHA-1 equals to 20 bytes.

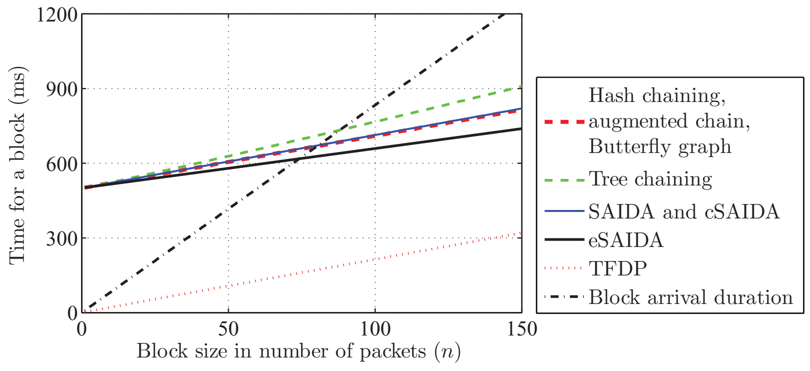

Fig. 3. Computation cost (time to verify a block of packets) versus block size.

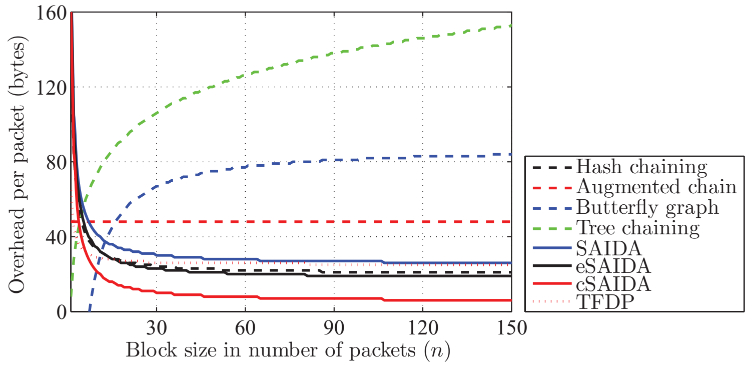

Fig. 4. Communication overhead (bytes per packets) versus block size.

Results of the analysis. We plot in Figure 3 the computation costs for all considered authentication schemes as n varies from 0 to 150 packets. The figure shows that the TFDP scheme is more efficient than the others. This is because it does not verify a digital signature per each block. Recall, however, that to build the Merkle hash tree used in TFDP, the whole stream needs to be available. This makes TFDP only suitable for on-demand streaming of preencoded video streams.

In Figure 3, we also plot the time it takes for a block of packets of size n to arrive at the receiver, which is computed from the packet generation rate α. Clearly, the block arrival time should be larger than the block verification time. Otherwise, the receiver will not have enough processing capacity to verify the authenticity of packets in real time. Therefore, by looking at Figure 3, we notice that small block sizes may not be suitable for all authentication schemes except TFDP. This implies that a minimum block size is required to support devices with limited processing capacity. For example, for the data used in producing Figure 3, a block size of approximately 100 packets would be needed to safely use any of the authentication schemes. This also indicates that the receiver needs to allocate a buffer of size at least 100 packets for most of the schemes (see buffering requirements in Table II).

Next, we plot the per-packet communication overhead against the block size n for all authentication schemes in Figure 4. The communication overhead is the number of additional bytes added to each packet to implement the authentication scheme. The figure shows that the cSAIDA authentication scheme imposes the least amount of communication overhead. Moreover, the per-packet overhead

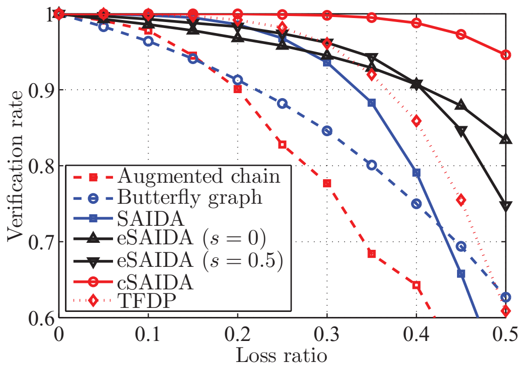

Fig. 5. Verification rate versus packet loss ratio under the bursty loss model.

stabilizes for block sizes greater than 50 packets for all schemes, except for the simple tree chaining scheme in which the overhead keeps increasing as the block size increases.

4.2 Simulation

Simulation setup. We have implemented all authentication schemes described in Section 3 in a simulator to study the impact of packet losses on their performance. The main performance metric used is the packet verification rate, which is the fraction of packets successfully verified over all received packets when packets carrying the authentication information could be lost. Notice that we are analyzing the loss tolerance for each authentication scheme, not the loss tolerance of the video decoder which may employ various error concealment methods.

We consider two common models for packet losses: bursty and random. The bursty loss model is typical in wired networks where a sequence of packets may get dropped because of a buffer overflow in one of the routers on the network path from sender to receiver, whereas the random loss model is usually used to capture bit errors in wireless environments. Notice that some multimedia streaming techniques over wired networks use interleaved packetization of data, which can change the observed loss pattern at the receiver from bursty to random. Thus, it is important to analyze the performance of the authentication schemes under both models of packet losses.

For simulating bursty packet losses, we implemented a two-state Markov chain, as it has been shown to accurately model bursty losses [Yajnik et al. 1999]. In the two-state Markov chain, one state indicates that a packet is received and the other indicates the packet is lost. Transition probabilities between these two states are computed based on the target average loss ratio and the expected burst length using the method in Park et al. [2003].

Values of the other parameters used in the simulation are listed in Table I.

Simulation results. The results for the packet verification rates versus average packet losses are given in Figure 5 for the bursty loss model, and in Figure 6 for the random loss model. The hash chaining and tree chaining schemes are not included in these figures, since the former does not tolerate any packet loss and the latter always has a loss resilience of 100%; each packet in tree chaining carries all information needed for its verification. In Figures 5 and 6, we fixed the communication overhead to 40 bytes per packet (except for the augmented chain which has 42 bytes since it cannot work with 40 bytes).

A few observations can be made on these two figures. First, cSAIDA clearly exhibits the best resilience to the loss under both bursty and random loss models. For example, for bursty losses with an average

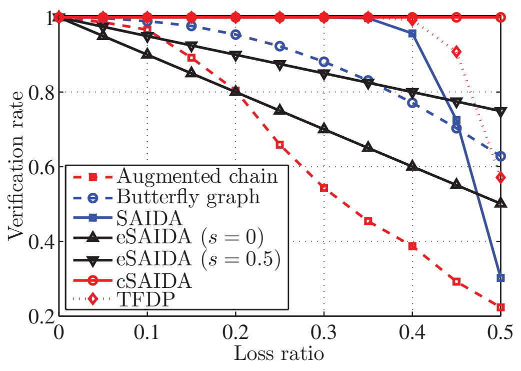

Fig. 6. Verification rate versus packet loss ratio under the random loss model.

Table III. Loss Counteraction by the Authentication Schemes with Fixed Overhead of 40 Bytes (42 bytes for the augmented chain)

Scheme	Loss counteraction
Augmented chain	Replicating the signature twice within a block
Butterfly graph	The best compromise of number of edges and signature replication is local-searched
SAIDA	An FEC factor of 1.9, i.e., 47% expected loss ratio
eSAIDA ($s = 0$)	An FEC factor of 3.6, i.e., 72% expected loss ratio
eSAIDA ($s = 0.5$)	An FEC factor of 2.7, i.e., 63% expected loss ratio
cSAIDA	An FEC factor of 2.8, i.e., 65% expected loss ratio
TFDP	An FEC factor of 2, i.e., 50% expected loss ratio

loss rate of 40% in the authentication information, about 99% of the received packets can be verified. Second, the loss tolerance of the authentication schemes does indeed depend on the loss model, not only on the average loss rate. For example, with an average loss rate of 40% of the authentication information, the SAIDA scheme can verify up to 79% of the received packets under bursty losses, while this ratio is 96% under random losses. We now briefly discuss the reasons underlying these behaviors in the figures.

Recall that loss is counteracted either by FEC-coding or by replicating some authentication information, that is, digests and block signature. Let us call the fraction n/n_{enough} the FEC factor. Depending on the scheme, the overhead of 40 bytes per packet results in different FEC factors for FEC-based schemes (SAIDA variants and TFDP) or different number of replications for replication-based ones (augmented chain and butterfly graph), as shown in Table III. For SAIDA, the 40 bytes per packet leads to FEC factors of 1.9, which means resistance to loss of up to 47% of a block, as indicated in the table. Calculation of FEC factors for cSAIDA and TFDP with 40 bytes per packet follow the same procedure. eSAIDA does not only rely on FEC. Because it couples every pair of packets together, it also attaches the hash value of a packet to its couple packet with a probability parameter s. The 40 bytes per packet for eSAIDA with $s = 0$ and $s = 0.5$ leads to the high FEC factors of 3.6 and 2.7, respectively. Thus, with losses up to 72% and 63%, the block signature and hash values of packet couples can be retrieved. However, since loss of a packet threatens verifiability of its couple, verification ratio of eSAIDA is not as high as cSAIDA even though its FEC factor is almost equal or higher. Augmented chain has a fixed number of edges and allows customization of loss resilience versus communication overhead only by varying the number of replications of the signature packet. The 42 bytes per packet allows it to replicate the signature twice within the block. The butterfly graph allows customizing the communication overhead by replicating signature as well as varying the number of edges of the graph. These two parameters are in trade-off with each other. We perform a local search to find the best balance for that in our simulations, given a fixed amount of communication overhead.

Recall that with 40 bytes per packet, the FEC factor of cSAIDA would be 2.8 and up to 65% loss is tolerated. This can be observed in Figure 6, which depicts that with random loss up to 50%, cSAIDA keeps the verification rate almost 1. However, the effect of bursty losses is different and more serious, as expected. Intuitively, if a loss of ratio 50% has a random pattern, for each block almost half of the packets are lost, which is easily resisted by the FEC factor of 2.8. On the other hand, with bursty loss of the same average ratio, a significant fraction of packets of a block could be lost during bursts, while some other blocks observe much less losses. This results in unverifiability of a few blocks, since the authentication information for those blocks cannot be retrieved from the received packets at all. This unverifiability can be seen in Figure 5, even though the loss ratio 50% is less than the ratio 65% we prepared the stream for. That is why the FEC-based schemes perform better under random loss model compared to bursty loss.

We can also notice the different decreasing behavior of FEC-based schemes (SAIDA variants and TFDP) with the two different loss patterns. With random losses, the verification rate sharply falls if the loss ratio exceeds the ratio supported by the FEC factor. This is clear in the plot for SAIDA and TFDP; same phenomenon happens to cSAIDA at 65% loss that is not shown in the figure. With bursty loss, on the other hand, the decreasing behavior is more smooth, because according to the preceding implication, there is no explicit loss ratio value, below which is easily tolerated and beyond which it suddenly gets too hard to resist.

The third point that can be noticed is the linear decreasing behavior of eSAIDA with random loss. The 40 bytes per packet allows eSAIDA with $s = 0$ (no packet hash value is attached to its couple) and with $s = 0.5$ (hash of half of packets is attached to their couples) to have FEC factors of 3.6 and 2.7, which resist 72% and 63% loss, respectively. That means, a random loss of up to 50% (Figure 6) is easily tolerated by them. Hence, the decrease in verification ratio is not because of being unable to retrieve the FEC-coded authentication information of a block. Rather, the unverifiability of some packets, say p_x, is only because their couple, say p_{x+1}, is lost, and the hash of p_{x+1} is not attached to p_x, so the hash value $h(p_x||p_{x+1})$ cannot be reconstructed to be verified. The more the loss ratio, the more the number of packets that are missing the hash of their couples. Also, it can be observed that with $s = 0.5$, this increase in unverified packets ratio is less, as can be expected. With bursty loss, on the other hand, both packets of a pair are more likely to be lost together. That is, with each burst of loss, at most two packets can be left unverifiable: the ones right before and right after the burst begins and ends. Thus, unverifiability can be both due to not being able to retrieve authentication information of a block (which was very unlikely with random loss below the loss ratio supported by the FEC factor) and not being able to verify a packet because its couple is missing. Therefore, this phenomenon, that is, linear decrease of verification rate, does not take place.

We can also notice that, unlike most of the schemes, the augmented chain performs worse under the random loss model compared to the bursty one. That can be attributed to the structure of the augmented chain, which is designed to have the least unverifiability effect with bursty losses. With each burst of loss, up to a few packets can be left unverifiable in the augmented chain. Thus, when each burst consists of one packet, that is, random loss, the ratio of unverifiable packets increases. Also, the decreasing behavior of augmented chain curves in Figures 5 and 6 is not so straight, which is most probably because we obtain the parameters p and a for the the augmented chain scheme (see Section 3.2) by a local search for best verification rate, given a fixed amount of overhead.

Finally, we fix the average loss rate at 20% with bursty losses, and we vary the communication overhead per packet. That is done either by varying the FEC factor (for FEC-based schemes) or by changing the number of replications of the signature (for replication-based schemes). Then, we analyze

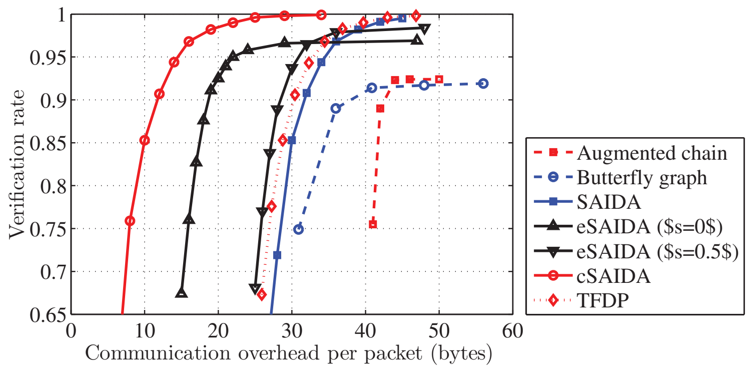

Fig. 7. Verification rate versus communication overhead.

the verification rates for different values of the per-packet overhead. The results for all authentication schemes are given in Figure 7. The figure confirms the efficiency of the cSAIDA scheme in carefully minimizing the number of bytes needed to encode the authentication information. With less than 30 additional bytes per packet, cSAIDA can achieve 100% verification rate under bursty losses with an average loss rate of 20%, whereas other schemes need almost double this number of overhead bytes to achieve comparable loss resilience.

4.3 Summary and Discussion

Our analysis and simulation with realistic parameters and in different environments indicate that cSAIDA (the improved version of the SAIDA authentication scheme proposed in Pannetrat and Molva [2003]) imposes the least amount of communication overhead and achieves the best tolerance to the loss of the authentication information. cSAIDA capitalizes on the fact that not all hash values of packets in a block need to be transmitted, since a large portion of these hashes can be reconstructed from the received packets. cSAIDA, however, requires a digital signature verification per block, which is costly. The TFDP [Habib et al. 2005] scheme, on the other hand, is very efficient in terms of computation cost, but only for offline streams. That is because TFDP performs one digital signature verification for the whole stream, which requires the whole stream to be available. Therefore, TFDP is not suitable for live streaming applications where packets are generated online in real time.

In addition, as shown in Table II, most of the authentication schemes for nonscalable video streams require the receiver to buffer a block of packets, which needs memory space. In case that the receiver has a limited memory space, the simple hash chaining [Gennaro and Rohatgi 1997] or tree chaining [Wong and Lam 1999] authentication schemes can be used.

Furthermore, we mention that some streaming applications employ TCP to reliably transport data from the sender to receivers. TCP could be a possible option for streaming if there are infrequent packet losses and the round-trip time is small. In this case, the simple hash chaining authentication scheme would suffice as loss resiliency and its associated complex operations in other authentication schemes are not needed.

4.4 iTFDP

We propose to combine the best features of cSAIDA (communication efficiency) and TFDP (computation efficiency) in designing an efficient authentication scheme for streaming of preencoded streams. We call this scheme the improved TFDP and we refer to it by iTFDP. Similar to cSAIDA, iTFDP divides data into blocks, each with with n packets: $p_1, p_2, \ldots, p_n$ and it computes $H = h(p_1)||h(p_2)|| \cdots ||h(p_n)$. It

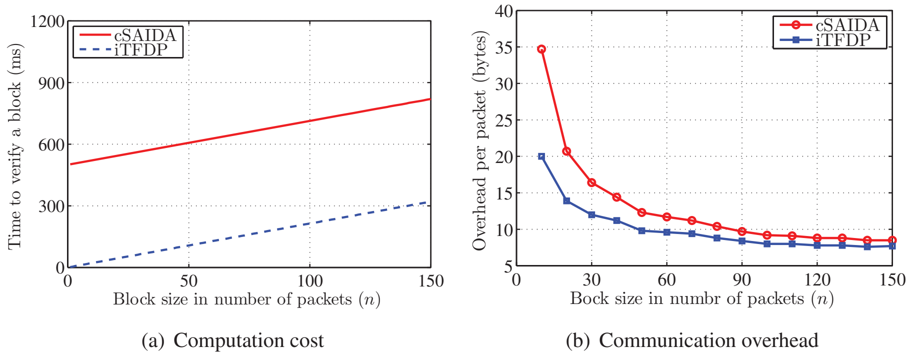

Fig. 8. Comparison between the proposed authentication scheme (iTFDP) and cSAIDA.

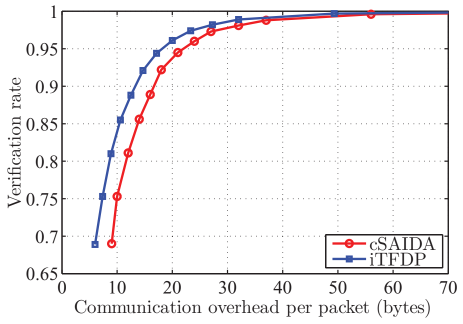

Fig. 9. Verification rate of iTFDP versus cSAIDA for different per-packet overhead values.

also systematically FEC-codes the n pieces of H into $m \geq n$ pieces $H_1, H_2, \ldots, H_m$, and only takes the parity symbols $H_{n+1}, H_{n+2}, \ldots, H_m$. These symbols are concatenated, divided into $\lfloor n(1-\rho) \rfloor$ pieces, and FEC-coded again into n pieces to be attached to all the n packets of the block. Unlike cSAIDA, iTFDP does not sign $h(H)$, whose authenticity is essential for verifying the block. Instead, it authenticates the values $h(H)$ of blocks using a Merkle hash tree, as done in TFDP. Accordingly, the computation cost can be significantly reduced since only one signature is designated for the whole stream. Therefore, iTFDP can achieve the low communication overhead of cSAIDA and the low computation cost of TFDP, without impacting other factors including delay and buffering requirements.

To validate the efficiency of iTFDP, we have implemented it in our simulator and compared it against cSAIDA. We stream a video sequence with 1 million packets in an environment with a bursty loss and average loss rate of 20%. All other parameters in this simulation are the same as in Table I. In Figure 8, we plot the computation cost and communication overhead of iTFDP and cSAIDA. The figure clearly shows that iTFDP has a significantly smaller computation cost than cSAIDA, and iTFDP further reduces the (already-optimized) communication overhead of cSAIDA. In Figure 9, we compare the verification rate of the received packets for iTFDP versus cSAIDA in presence of bursty losses of the authentication information. The figure also shows that iTFDP improves the verification rate of cSAIDA. Therefore, iTFDP achieves its goals of reduced computation cost and communication overhead as well as high verification rate in presence of losses.

5. AUTHENTICATION OF SCALABLE MULTIMEDIA STREAMS

In the previous two sections, we presented and analyzed authentication schemes for multimedia streams that are encoded in a nonscalable manner. Nonscalable streams offer very limited flexibility in supporting heterogeneous receivers. In contrast, multimedia streams created using scalable video coding techniques can easily be adapted to support a wide range of diverse receivers and network conditions. This ease of adaptation comes at a cost of reduced coding efficiency, that is, at the same bitrate a nonscalable stream yields better visual quality than a scalable stream. However, this quality gap has been significantly reduced with the recent H.264/SVC standard [Schwarz et al. 2007], which adds scalable coding to the state-of-the-art H.264/AVC video coding standard. Thus, scalable streams encoded using H.264/SVC coder are increasingly being used in many applications, such as Schierl et al. [2007]. In this section, we present and analyze authentication schemes for scalable multimedia streams. We also analyze the impact of the adaptation on the authentication of such streams.

It is important to note that any authentication scheme for a scalable stream must be able to authenticate all valid substreams that can be extracted from it. Substreams are truncated versions of the original full stream, created by dropping some layers. However, these truncations are intentionally made to customize the stream based on the capacity of the receiver and the network conditions. These truncations are also different from packet losses that may occur because of network congestion, in the sense that we usually try to recover from packet losses using erasure coding schemes or retransmissions while we do not recover truncated parts of the stream, yet we need to authenticate the received substream.

We present in the next subsection possible authentication schemes for scalable streams. Then, we evaluate these schemes in the subsection following the next.

5.1 2D Hash Chaining

A scalable stream is a sequence of video frames. Each frame has a base layer and multiple cumulative enhancement layers. The base layer is essential and provides the basic video quality, while each enhancement layer adds improvements to its predecessor layers. Thus, a scalable stream can be thought of as a sequence with two dimensions: horizontal and vertical. The horizontal dimension represents successive frames, while the vertical dimension represents the layers in each frame. An authentication scheme for scalable streams needs to validate the authenticity across the two dimensions.

The hash chaining scheme can be extended to scalable streams as follows. First, each enhancement layer of a frame is hashed and its hash is attached to its predecessor layer of the same frame. Thus, the base layer of a frame contains a digest of all enhancement layers of the frame. Then, the hash of the base layer (and the digest) of a frame is attached to the previous frame. The two-dimensional hash chaining scheme for scalable streams is illustrated in Figure 10. Notice that a substream with a base layer and 0 or more (consecutive) enhancement layers can be verified using the authentication information carried only in that substream; no additional information about the truncated layers is needed for the verification of the received layers. This is a good property especially for streaming in large-scale systems where proxy servers and/or third-party content delivery networks can be involved in the streaming. For example, a proxy server that has a substream can serve it to clients without needing to worry about the layers that it does not have. In fact, if the authentication information is embedded in the multimedia stream, the proxy server does not even need to know whether or not the stream carries authentication information. This embedding of authentication information can be done as supplemental enhancement information NAL (Network Abstraction Layer) units in case of H.264/SVC scalable video [Schwarz et al. 2007].

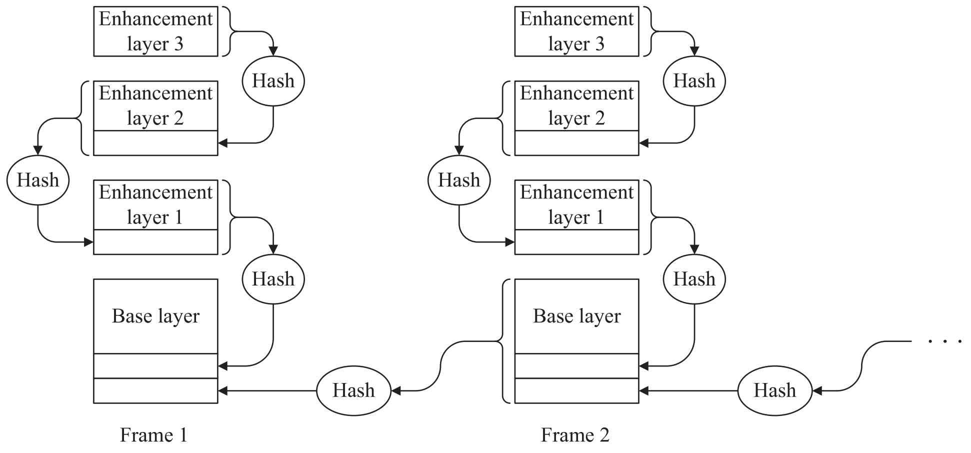

Fig. 10. Authenticating scalable streams by hash chaining.

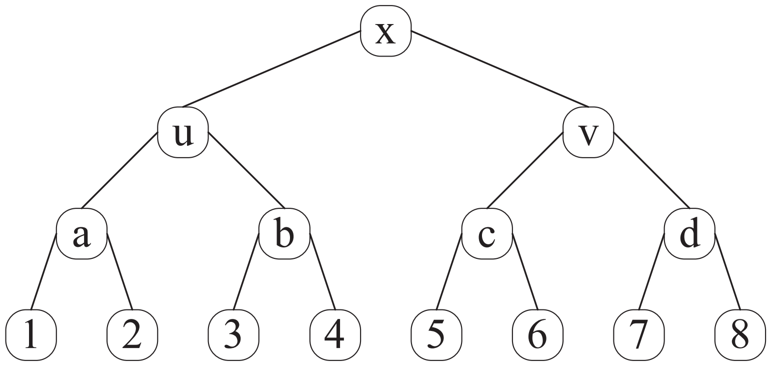

Fig. 11. A Merkle hash tree built over 8 layers. As an example, for authenticating layer 1 only, the hash of layer 2 as well as *b* and *v* are needed.

5.2 2D Tree Chaining

In the tree chaining scheme for authenticating scalable streams, Merkle hash trees [Merkle 1989] are utilized [Wu and Deng 2006; Li et al. 2005; Kaced and Moissinac 2006a; 2006b; Suzuki et al. 2004; Gentry et al. 2005]. The enhancement layers of a video frame are arranged as leaves of a Merkle hash tree, the root of which forms the digest of the whole set of layers. This is illustrated in Figure 11. Then, the roots of the hash trees representing digests of video frames form another hash tree. Upon removal of some layers, a receiver needs some extra digests for verifying the remaining layers, that is, for reconstructing the root digest of the hash tree. For the example in Figure 11, if a client is receiving the base layer only (layer 1 in the figure), he/she needs digests 2, *b*, and *v* to reconstruct and verify the root. If the first five layers are being received, the client needs digests 6 and *d*.

Notice that a proxy server serving a substream with only a subset of the enhancement layers will need to attach some hash values of the other enhancement layers, such that the clients can verify the received layers. For example, if the proxy server is adapting a stream for a limited-capacity client that can receive only the base layer, the proxy will have to attach the digests 2, *b*, and *v* to enable the client to reconstruct and verify the root. This means that the proxy (or the third-party delivery network) must know about the structure of the Merkle hash tree and the employed authentication scheme. This is may not be desirable as proxy servers of content delivery networks serve thousands of streams at the same time from different content providers who may be employing different authentication schemes. To mitigate this problem, a content provider can embed with each layer all the necessary information to verify it in case that the stream is truncated beyond this layer. While this solution makes proxy servers unaware and transparent of the authentication scheme, it increases the communication overhead.

We next analyze the preceding two authentication schemes. The computation cost analysis is similar to the analysis in Section 3, except that there is an additional hash computation for each enhancement layer. In addition, the delay and the receiver buffer size for the two-dimensional hash chaining and tree chaining are the same as the basic hash chaining and tree chaining schemes. The communication overheads, however, are different and they depend on whether an intermediate proxy server is aware of the authentication scheme or not. In the following subsection, we evaluate these communication overheads using simulation.

5.3 Evaluation of Authentication Schemes for Scalable Streams

We evaluate the authentication schemes for scalable streams using the the recent H.264/SVC video coding standard [Schwarz et al. 2007]. The H.264/SVC standard supports three types of scalability: temporal, spatial, and quality or Signal-to-Noise Ratio (SNR). Temporal scalability employs a hierarchical prediction structure among video frames to enable up to 6 different frame rates. Each frame rate is achieved by a temporal layer that adds enhancement to its preceding layer with lower frame rate. Spatial scalability allows up to 8 different resolutions as 8 spatial layers using various pixel sampling and interpolation schemes. The quality scalability supports up to 16 quality layers, which can be achieved either by using a different quantization parameter for each quality layer, or by distributing the transform coefficients into several quality layers. The file format for H.264/SVC streams [Amon et al. 2007] is well structured to support efficient extraction and streaming of substreams with layers from the three types of scalability. Thus, an H.264/SVC scalable stream can be abstractly modeled as a base layer with potentially many enhancement layers, where each layer improves the stream along one of the three dimensions of scalability.

An H.264/SVC multimedia stream could (theoretically) contain: up to $6 \times 8 \times 16 = 768$ layers. Thus, the authentication information for the enhancement layers can add significant communication overhead, especially for limited capacity receivers. The presence of middle proxies for helping in streaming and whether they are aware of the authentication scheme or not also impact the amount of this communication overhead.

We implemented the two-dimensional hash chaining and tree chaining in our simulator. Using this simulator, we analyze the communication overhead for both schemes. We encode a video sequence with 16 enhancement layers, with corresponding bitrates ranging from 128 kbps to almost 1 Mbps. We create authentication information for the 16 layers using hash chaining and tree chaining. Then, we simulate a wide range of receivers with different bandwidth. Each receiver will obtain a truncated version of the stream based on its bandwidth. The truncated stream should contain the needed authentication information to verify the received layers. We measure the percentage of this authentication information to the total amount of data transmitted to the receiver. We plot the results in Figure 12, where the curve denoted by "Hash chain" represents the two-dimensional hash chaining scheme, and the curve denoted by "Tree chain" represents the tree chaining scheme. For these two curves, there are no intermediary streaming proxies involved. If we consider proxies, the tree chaining scheme needs to include more authentication information with each enhancement layer to make it possible for the receiver to verify that layer in case no higher enhancement layers are received, as discussed earlier. The presence of

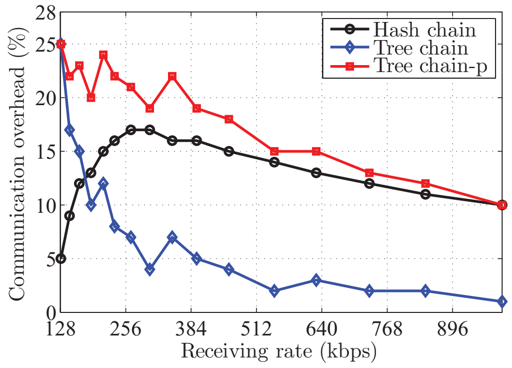

Fig. 12. Communication overhead versus bitrate.

proxies does not impact the hash chaining scheme, as an individual layer can be authenticated with the information carried by that layer only. We also plot the overhead of the tree chaining scheme in presence of proxies in Figure 12, and we denote the curve by "Tree chain-p".

A few observations can be made on Figure 12. First, for low-bandwidth receivers that obtain a few number of layers, the hash chaining scheme imposes the least amount of communication overhead. Minimizing the overhead for such receivers is crucial as they have limited bandwidth in the first place. As the number of received layers increases, the tree chaining scheme becomes more desirable because of the efficiency in organizing the hashes in Merkle hash trees. The figure shows that the tree chaining scheme imposes lower overhead than hash chaining in most cases, except when the number of layers is very few. Second, if proxies are involved in adapting scalable streams, the content provider needs to add extra authentication information with each layer in case of tree chaining. Accounting for this overhead will actually make the simple hash chaining scheme more efficient than the tree chaining scheme. Figure 12 shows that the Tree-Chain-P curve is always higher than the Hash Chaining curve for all bitrates.

6. CONCLUSIONS

We have surveyed, analyzed, and compared the most important solutions proposed in the literature for the problem of verifying the authenticity of multimedia streams. We carried out numeric analyses and simulations for all authentication schemes to study their performance in terms of computation cost, communication overhead, delay, receiver buffer size, and tolerance to packet losses. The results from our study can be used to understand the merits and shortcomings of each authentication scheme. Therefore, our results provide guidelines in choosing the appropriate authentication scheme for various multimedia streaming applications. In addition, by scrutinizing the details of each authentication scheme and contrasting them to each other in different environments, our results could also stimulate more research to improve the performance of these authentication schemes.

We considered authentication schemes for nonscalable and scalable multimedia streams. For nonscalable streams, we found that the scheme proposed in Pannetrat and Molva [2003] (denoted by cSAIDA) imposes the least amount of communication overhead and achieves the best tolerance to the loss of authentication information. cSAIDA, however, requires one digital signature verification per block, which is costly. The TFDP [Habib et al. 2005] scheme, on the other hand, is very efficient in terms of computation cost, but only for offline streams. That is because TFDP performs one digital signature verification for the whole stream, which requires the whole stream to be available. Therefore, TFDP is not suitable for live streaming applications where packets are generated online in real time. We then proposed an authentication scheme for offline streams that combines the advantages of both cSAIDA (communication efficiency) and TFDP (computation efficiency), which we call the improved TFDP (iTFDP). Using simulation, we showed that iTFDP has a significantly lower computation cost than cSAIDA, while it maintains its communication efficiency. Therefore, we believe that the proposed authentication scheme (iTFDP) is the best performing one for on-demand streaming applications of preencoded streams, and that cSAIDA is a good candidate for authenticating live streams.

In addition, we found that most of the authentication schemes for nonscalable streams require the receiver to buffer a block of packets. Besides imposing a delay on the start of the verification process, this buffering requirement needs memory space. In case that the receiver has a limited memory space, the simple hash chaining [Gennaro and Rohatgi 1997] or tree chaining [Wong and Lam 1999] authentication schemes can be used, which do not require buffering. We note, however, that the hash chaining is very sensitive to packet loss, and thus it is best used with reliable data transport protocols, whereas the tree chaining has a high communication overhead.

For scalable multimedia streams, we showed how the hash chaining and tree chaining schemes can be extended to support them. We also analyzed the impact of adapting scalable streams (by dropping some enhancement layers) on the authentication schemes, which should enable receivers to verify substreams that contain only a subset of the enhancement layers of the original streams. Our analysis showed that this support for adaptation increases the communication overhead of the tree chaining authentication scheme, especially if there are third-party proxy servers involved in delivering the streams to the receivers. From simulating the hash chaining and tree chaining schemes for wide range of receiver bandwidths, we found that the tree chaining scheme imposes lower overhead than hash chaining in most cases, except when the number of layers is very few. However, if proxies are involved in adapting scalable streams, the content provider needs to add extra authentication information with each layer in case of tree chaining. Accounting for this overhead will make the hash chaining scheme more efficient than the tree chaining scheme for all bitrates.

REFERENCES

Amon, P., Rathgen, T., and Singer, D. 2007. File format for scalable video coding. *IEEE Trans. Circ. Syst. Video Technol. 17*, 9, 1174–1185.

Argyroudis, P., Verma, R., Tewari, H., and O'Mahony, D. 2004. Performance analysis of cryptographic protocols on handheld devices. In *Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA'04)*. 169–174.

Atrey, P., Yan, W., and Kankanhalli, M. 2007. A scalable signature scheme for video authentication. *Multimedia Tools Appl. 34*, 107–135.

Challal, Y., Bettahar, H., and Bouabdallah, A. 2004. A taxonomy of multicast data origin authentication: Issues and solutions. *IEEE Comm. Surv. Tutor. 6*, 3, 34–57.

Deng, R., Ma, D., Shao, W., and Wu, Y. 2005. Scalable trusted online dissemination of JPEG2000 images. *Multimedia Syst. 11*, 1, 60–67.

Gennaro, R. and Rohatgi, P. 1997. How to sign digital streams. In *Proceedings of the Advances in Cryptology (CRYPTO'97)*. Lecture Notes in Computer Science, vol. 1294. Springer, 180–197.

Gentry, C., Hevia, A., Jain, R., Kawahara, T., and Ramzan, Z. 2005. End-to-End security in the presence of intelligent data adapting proxies: The case of authenticating transcoded streaming media. *IEEE J. Select. Areas Comm. 23*, 2, 464–473.

Global. 2007. Video conferencing—A global strategic business report. Global Industry Analysts Inc. `http://www.strategyr.com/MCP-1062.asp`.

Golle, P. and Modadugu, N. 2001. Authenticating streamed data in the presence of random packet loss. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'01)*. 13–22.

Grosbois, R., Gerbelot, P., and Ebrahimi, T. 2001. Authentication and access control in the JPEG2000 compressed domain. In *Proceedings of the SPIE Applications of Digital Image Processing XXIV*. Vol. 4472. 95–104.

Habib, A., Xu, D., Atallah, M., Bhargava, B., and Chuang, J. 2005. A tree-based forward digest protocol to verify data integrity in distributed media streaming. *IEEE Trans. Knowl. Data Engin. 17*, 7, 1010–1014.

INSIGHT. 2006. Streaming media, IPTV, and broadband transport: Telecommunications carriers and entertainment services 2006–2011. The Insight Research Corporation. `http://www.insight-corp.com/execsummaries/iptv06execsum.pdf`.

KACED, R. AND MOISSINAC, J. 2006a. Multimedia content authentication for proxy-side adaptation. In *Proceedings of the International Conference on Digital Telecommunications (ICDT'06)*.

KACED, R. AND MOISSINAC, J. 2006b. SEMAFOR: A framework for authentication of adaptive multimedia content and delivery for heterogeneous networks. In *Proceedings of the International Conference on Internet Surveillance and Protection (ICISP'06)*.

KLIMA, V. 2005. Finding MD5 collisions a toy for a notebook. Cryptology ePrint Archive: Rep. 2005/075.

KLIMA, V. 2006. Tunnels in hash functions: MD5 collisions within a minute. Cryptology ePrint Archive: Rep. 2006/105.

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)*. Lecture Notes in Computer Science, vol. 3783. Springer, 389–401.

LIANG, C., LI, A., AND NIU, X. 2007. Video authentication and tamper detection based on cloud model. In *Proceedings of the Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP'07)*. Vol. 1. 225–228.

LIN, C. AND CHANG, S. 2000. Semi-Fragile watermarking for authenticating JPEG visual content. In *Proceedings of the SPIE Security and Watermarking of Multimedia Content II*. Vol. 3971. 140–151.

MENEZES, A., VANSTON, S., AND VAN OORSCHOT, P. 1996. *Handbook of Applied Cryptography*, 1st Ed. CRC Press, Boca Raton, FL.

MERKLE, R. 1989. A certified digital signature. In *Proceedings of the Advances in Cryptology (CRYPTO'89)*. Lecture Notes in Computer Science, vol. 435. Springer, 218–238.

NIST. 1993. Federal information processing standards (FIPS) publication 180: Secure hash standard. National Institute of Standards and Technology (NIST).

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

PARK, J., CHONG, E., AND SIEGEL, H. 2003. Efficient multicast stream authentication using erasure codes. *ACM Trans. Inf. Syst. Secur. 6*, 2, 258–285.

PARK, Y. AND CHO, Y. 2004. The eSAIDA stream authentication scheme. In *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA'04)*. Lecture Notes in Computer Science, vol. 3046. Springer, 799–807.

PENG, C., DENG, R., WU, Y., AND SHAO, W. 2003. A flexible and scalable authentication scheme for JPEG2000 image codestreams. In *Proceedings of the ACM Multimedia Conference*. 433–441.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, J. 2001. Efficient and secure source authentication for multicast. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'01)*. 35–46.

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

RABIN, M. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM 36*, 2, 335–348.

RIVEST, R. 1992. RFC1321; the MD5 message-digest algorithm. IETF.

RIVEST, R., SHAMIR, A., AND ADLEMAN, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21*, 2, 120–126.

RNCOS. 2006. Global IPTV market analysis (2006–2010). RNCOS. `http://www.rncos.com/Report/IM063.htm`.

SCHIERL, T., STOCKHAMMER, T., AND WIEGAND, T. 2007. Mobile video transmission using scalable video coding. *IEEE Trans. Circ. Syst. Video Technol. 17*, 9, 1204–1217.

SCHWARZ, H., MARPE, D., AND WIEGAND, T. 2007. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Trans. Circ. Syst. Video Technol. 17*, 9, 1103–1120.

SHAMIR, A. 1979. How to share a secret. *Comm. ACM 22*, 612–613.

SULLIVAN, G. AND WIEGAND, T. 2005. Video compression—From concepts to the H.264/AVC standard. *Proc. IEEE 93*, 1, 18–31.

SUN, A., HE, D., ZHANG, Z., AND TIAN, Q. 2003. A secure and robust approach to scalable video authentication. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'03)*. Vol. 2. 209–212.

SUN, Q., CHANG, S., KURATO, M., AND SUTO, M. 2002. A new semi-fragile image authentication framework combining ECC and PKI infrastructure. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'02)*. Vol. 2. 440–443.

SUZUKI, T., RAMZAN, Z., FUJIMOTO, H., GENTRY, C., NAKAYAMA, 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 (IFIP CMS'04)*. Lecture Notes in Computer Science, vol. 175. Springer, 237–249.

TILLICH, S. AND GROSCHDL, J. 2004. A survey of public-key cryptography on J2ME-enabled mobile devices. In *Proceedings of the International Symposium on Computer and Information Sciences (ISCIS'04)*. Lecture Notes in Computer Science, vol. 3280. Springer, 935–944.

WANG, W. AND FARID, H. 2006. Exposing digital forgeries in video by detecting double MPEG compression. In *Proceedings of the ACM Multimedia and Security Workshop*. 37–47.

WENGER, S., HANNUKSELA, M., STOCKHAMMER, T., WESTERLUND, M., AND SINGER, D. 2005. RFC 3984; RTP payload format for H.264 video. IETF.

WONG, C. AND LAM, S. 1999. Digital signatures for flows and multicasts. *IEEE/ACM Trans. Netw. 7*, 4, 502–513.

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

YAJNIK, M., MOON, S., KUROSE, J., AND TOWSLEY, D. 1999. Measurement and modeling of the temporal dependence in packet loss. In *Proceedings of the IEEE InfoCom'99*. Vol. 1. 345–352.

ZHANG, Z., SUN, Q., AND WONG, W. 2005. A proposal of butterfly-graph based stream authentication over lossy networks. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'05)*. 784–787.

ZHISHOU, Z., APOSTOLOPOULOS, J. SUN, Q., WEE, S., AND WONG, W. 2007. Stream authentication based on generalized butterfly graph. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'07)*. Vol. 6. 121–124.

ZHU, B., SWANSON, M., AND LI, S. 2004. Encryption and authentication for scalable multimedia: Current state of the art and challenges. In *Proceedings of the SPIE Internet Multimedia Management Systems V*. Vol. 5601. 157–170.

Received October 2008; revised December 2008; accepted January 2009