ANALYSIS OF AUTHENTICATION SCHEMES FOR NONSCALABLE VIDEO STREAMS

Mohamed Hefeeda and Kianoosh Mokhtarian

School of Computing Science
Simon Fraser University
Surrey, BC, Canada

ABSTRACT

The problem of multimedia stream authentication has received significant attention by previous works and various solutions have been proposed. These solutions, however, have not been rigorously analyzed and contrasted to each other, and thus their relative suitability for different streaming environments is not clear. In this paper, we conduct comprehensive analysis and comparison among the main authentication schemes proposed in the literature. To perform this analysis, we propose five performance metrics: computation cost, communication overhead, receiver buffer size, delay, and tolerance to packet losses. We derive analytic formulas for these metrics for all schemes, and we numerically analyze these formulas. In addition, we implement all schemes in a simulator to study their performance in different environments. Our detailed analysis reveals the merits and shortcomings of each scheme and provides guidelines on choosing the most appropriate scheme for a given application. Our analysis also helps in designing new authentication schemes and/or improving existing ones.

Index Terms— Multimedia authentication, multimedia security, secure streaming.

1. INTRODUCTION

Multimedia content and services have seen wide spread adoption in recent years. Multimedia services such as Internet streaming, video on demand, video conferencing, and Internet Protocol Television (IPTV) are common place. This wide spread adoption makes ensuring the authenticity of multimedia content transported over public networks, e.g., the Internet, an important problem. Ensuring the authenticity of the multimedia content means 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.

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 [1]. 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. Moreover, 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.

In this paper, we rigorously analyze and compare the main schemes proposed in the literature to authenticate multimedia streams. We focus on the widely used nonscalable video streams. A multimedia stream is nonscalable if it is encoded as a single layer and only the complete layer is decodable [2]. Nonscalable streams provide higher coding efficiency than multi-layer scalable streams [3], which are proposed in the literature to offer more flexibility, as partial streams (substreams) can be decoded by receivers. Scalable streams, however, have not yet been widely used in practice. To conduct our 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. We numerically analyze these formulas to explore the performance of the authentication schemes for a wide range of parameters. 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. For example, we analyze the authentication schemes under 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.

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

2. QUANTITATIVE ANALYSIS OF AUTHENTICATION SCHEMES

To rigorously evaluate various multimedia authentication schemes, we propose (in Section 2.1) five performance metrics that cover all angles of the stream authentication problem. Then, in each of Sections 2.2— 2.8, we briefly describe the main idea of an authentication scheme and we analyze it using these metrics.

We note that the authentication schemes considered in this paper do not use or depend on the characteristics of the video stream. Some schemes, on the other hand, depend on these characteristics and extract a set of *content-based* features from the video to authenticate, e.g., [4, 5]. Conducting a quantitative analysis of these schemes is difficult, as the video characteristics are quite diverse and varying. Hence, we do not consider such schemes in this paper.

2.1. Performance Metrics

We propose the following performance metrics for analyzing authentication schemes for multimedia streams.

- *Computation Cost.* It is the CPU time needed to verify the authentication information by the receiver —we assume that the sender (content provider) is powerful enough. Evaluating the computation cost is important especially if the receiver has a limited processing capacity, e.g., 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 afford it, and thus cannot verify the stream at all.

- *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, 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 affects the amount of delay at receiver side, which is included in the Delay metric below.

- *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 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 above 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 paper and their notations in Table 1. We also mention the range of values used for each parameter. In Section 3, we justify these values.

2.2. Hash Chaining

Hash chaining [6] is one of the simplest techniques to authenticate multimedia streams. 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 has chaining is straightforward. For a block of n packets, hash chaining computes n hash values and verifies one digital signature. 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,

Table 1. Parameters used in this paper and their values.

Parameter	Value	Description
α	30 pkt/sec	Packet rate
l	1400 bytes	Packet size
n	128 pkts or variable	Block size
N	1,000 to 1,000,000	Number of blocks
ρ	0 to 0.5	Packet loss ratio
n_{enough}	$0.8n$	#packets for verification
t_{sig}	500 ms	Time to verify signature
t_{hash}	0.1 ms	Time to compute hash
s_{sig}	128 bytes	Signature size
s_{hash}	20 bytes	Hash size (SHA-1)
n_{rows}	32	Number of rows
s	0, 0.25, 0.5	For eSAIDA
n_{sig}	$1/16\ n$ or searched	Number of signatures
p, a	searched	For augmented chain

because hash chaining starts at the last packet in the block. Thus, with a packet generation rate of α packets per second, the total delay is n/α seconds.

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, i.e., 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 above 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.

2.3. Augmented Hash Chaining

In the augmented hash chaining scheme [7], 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 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, i.e., it depends on the loss rate and loss pattern (random or bursty). We analyze this tolerance using simulation in Section 3.

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.

2.4. Butterfly Hash Chaining

Zhang et al. [8] 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 [9]. 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 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 till 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.

2.5. Tree chaining

Wong and Lam [10] proposed the use of Merkle hash trees [11] 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 the 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 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.

2.6. SAIDA and eSAIDA

Park et al. [12] presented SAIDA (Signature Amortization using Information Dispersal Algorithm) for stream authentication. 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}, i.e., $n_{enough} = (1 - \rho)n$. H along with a signature on $h(H)$ is divided into n_{enough} ($n_{enough} \leq n$) pieces, IDA-coded into n pieces and split over all the n packets of the block. Any n_{enough} pieces suffice to re-construct the hashes and the signature to verify authenticity of the entire block. Note that the signature alone 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 one IDA-decoding per n packets, because there are efficient algorithms for erasure correction, such as Tornado codes [13] that use only XOR operations and operate in linear time of the block size, which can replace IDA-coding [14] 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), which we consider by n_{enough}, and is equal to $(s_{sig} + ns_{hash})/n_{enough}$ 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 [15]. 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 their couple's hash is parameterized by s ($0 \leq s < 1$) as an input, which governs a tradeoff 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.

2.7. cSAIDA

Pannetrat et al. in [16] 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 [14]) 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.

2.8. TFDP

Habib et al. [17] presented TFDP (Tree-based Forward Digest Protocol) for offline P2P streaming, i.e., 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 [10], a Merkle hash tree [11] 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 re-calculating 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 $(n\lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil + \lceil \log_2 N \rceil \lceil 2 s_{hash}/64 \rceil) t_{hash}$ seconds. Thus, compared to computations of other schemes, which all include one signature per block, TFDP's computations are much cheaper. 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

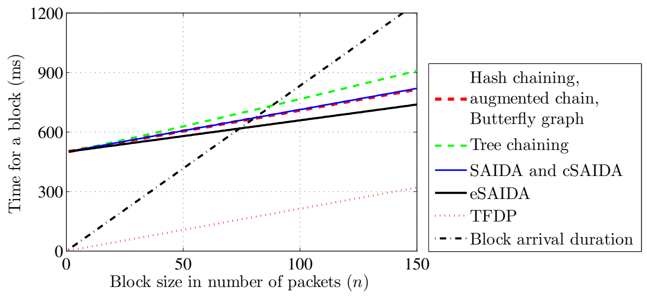

Fig. 1. Computation cost (time to verify a block of packets) versus block size.

a client requests one block, i.e., $x = 1$. 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. EVALUATION OF AUTHENTICATION SCHEMES

In this section, we compare the stream authentication techniques summarized in Section 2. 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.

3.1. Numerical Analysis

We present in Table 2 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 2.1. 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 1 and discussed below.

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 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) [18]. 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 $\alpha = \dfrac{320 \text{ kbps}}{1400 \text{ bytes}} \simeq 30$ packets per second.

Next, we estimate the computation costs of the digital signature and hashing operations. Digital signature operations

Table 2. Summary of the analysis of authentication schemes for nonscalable multimedia streams.

Scheme	Computation cost (sec/block)	Communication overhead (byte/pkt)	Total delay (sec)	Buffer size (pkt)
Hash chaining	$t_{sig} + t_{hash} n \lceil l/64 \rceil$	$\frac{s_{sig}}{n} + s_{hash}$	n/α	1
Augmented chain	$t_{sig} + t_{hash} n \lceil l/64 \rceil$	$\frac{n_{sig} s_{sig}}{n} + 2 s_{hash}$	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	$t_{sig} + t_{hash}\Big(n \lceil l/64 \rceil + (\lceil n \log_2 n \rceil - n) \lceil 2 s_{hash}/64 \rceil\Big)$	$s_{sig} + \lceil \log_2 n \rceil s_{hash}$	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	$t_{hash}\Big(n \lceil l/64 \rceil + \lceil s_{hash} n/64 \rceil + \lceil \log_2 N \rceil \lceil 2 s_{hash}/64 \rceil\Big)$	$s_{hash}(\frac{n}{n_{enough}} + \frac{1}{n} + \frac{\log_2(N/x)}{nx})$	—	n

are often very costly, because they involve modular multiplication of very large numbers. Since we assume that the signer is powerful enough, RSA [19] is an appropriate choice as the digital signature scheme, because its verification can be done efficiently when the public key is chosen properly, e.g., 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 1 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 (5-10%) of its CPU time. It is experimented in [20] 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, 32 MB of RAM, and running Windows CE Pocket PC 2002. A similar experiment [21] 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, e.g., 5-10%, and considering a safety margin, we took the value $t_{sig} = 500$ millisecond in Table 1. For the hashing algorithm, SHA-1 [22] and MD5 [23] 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 cryptanalysis have been done on MD5 and algorithms have been proposed for finding collisions [24, 25]. Although these cryptanalysis on MD5 are far from being practical for breaking a system in real-time, we

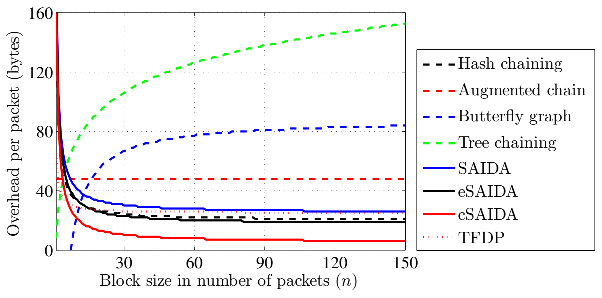

Fig. 2. Communication overhead (bytes per packet) versus block size.

chose SHA-1 as the hash algorithm for our evaluations. The digest size s_{hash} for SHA-1 equals to 20 bytes.

Results of the Analysis. We plot in Figure 1 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 hash tree used in TFDP, the whole stream needs to be available. This makes TFDP only suitable for on-demand streaming of pre-encoded video streams.

In Figure 1, 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. Oth-

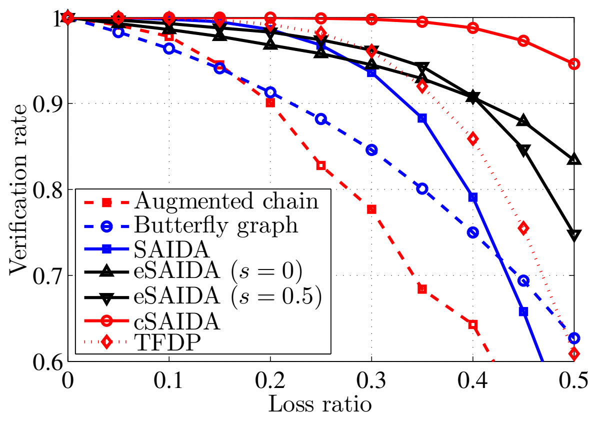

Fig. 3. Verification rate versus packet loss ratio under the bursty loss model.

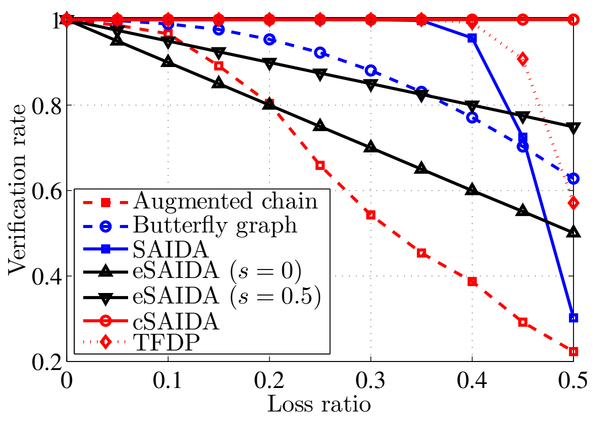

Fig. 4. Verification rate versus packet loss ratio under the random loss model.

erwise, the receiver will not have enough processing capacity to verify the authenticity of packets in real time. Therefore, by looking at Figure 1, 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 1, 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 2).

Next, we plot the per-packet communication overhead against the block size n for all authentication schemes in Figure 2. 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 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.

3.2. Simulation

Simulation Setup. We have implemented all authentication schemes described in Section 2 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 [26]. 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 [12].

Values of the other parameters used in the simulation are listed in Table 1.

Simulation Results. The results for the packet verification rates versus average packet losses are given in Figure 3 for the bursty loss model, and in Figure 4 for the random loss model. The hash chaining and tree chaining schemes are not included in these figures, since the former one 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 and does not depend of arrival of the others. In Figures 3 and 4, 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– as we shortly see how, and n = 128 packets.

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 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%, 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, i.e., 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). For SAIDA, the 40 bytes per packet leads to a FEC factor of 1.9, which means resistance to loss of up to 47% of a block. Calculation of FEC factors for cSAIDA and TFDP with 40 bytes per packet follow the same procedure. eSAIDA couples every pair of packets together, and 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 tradeoff 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.

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 4, 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 3, 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 above 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 4) 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, i.e., 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 random loss model compared to the bursty one. That can be attributed to the structure of the augmented chain, which is designed to have 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, i.e., random loss, the ratio of unverifiable packets increases. Also, the decreasing behavior of Augmented chain in Figures 3 and 4 is not so straight, which is most probably because we obtain the parameters p and a for the Augmented chain scheme (see Section 2.3) by a local search for best verification rate, given a fixed amount of overhead.

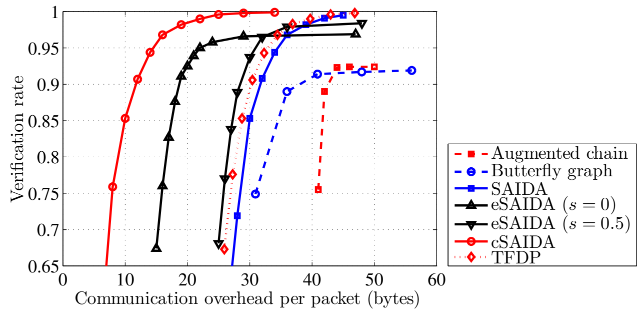

Fig. 5. Verification rate versus communication overhead.

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 replication of the signature (for replication-based schemes). Then, we analyze the verification rates for different values of the per-packet overhead. The results for all authentication schemes are given in Figure 5. 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.

3.3. 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 [16]—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 [17] 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 2, 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 [6] or tree chaining [10] 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 scheme would suffice, since loss resiliency and its associated complex operations as in other authentication schemes are not needed.

4. CONCLUSIONS

We have 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 stimulate more research to improve the performance of these authentication schemes. We considered authentication schemes for nonscalable multimedia streams. We found that the scheme proposed in [16] (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 [17] scheme, on the other hand, is 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.

5. REFERENCES

[1] Y. Challal, H. Bettahar, and A. Bouabdallah, “A taxonomy of multicast data origin authentication: Issues and solutions,” *IEEE Communications Surveys and Tutorials*, vol. 6, no. 3, pp. 34–57, July 2004.

[2] G. Sullivan and T. Wiegand, “Video compression—from concepts to the H.264/AVC standard,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, January 2005.

[3] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[4] P. Atrey, W. Yan, and M. Kankanhalli, “A scalable signature scheme for video authentication,” *Multimedia Tools and Applications*, vol. 34, pp. 107–135, July 2007.

[5] A. Sun, D. He, Z. Zhang, and Q. Tian, “A secure and robust approach to scalable video authentication,” in *Proc. of IEEE International Conference on Multimedia and Expo (ICME’03)*, Baltimore, MD, July 2003, vol. 2, pp. 209–212.

[6] R. Gennaro and P. Rohatgi, “How to sign digital streams,” in *Proc. of Advances in Cryptology (CRYPTO’97)*, Santa Barbara, CA, August 1997, vol. 1294 of *LNCS*, pp. 180–197, Springer-Verlag.

[7] P. Golle and N. Modadugu, “Authenticating streamed data in the presence of random packet loss,” in *Proc. of Network and Distributed Systems Security Symposium (NDSS’01)*, San Diego, CA, February 2001, pp. 13–22.

[8] Z. Zhang, Q. Sun, and W. Wong, “A proposal of butterfly-graph based stream authentication over lossy networks,” in *Proc. of IEEE International Conference on Multimedia and Expo (ICME’05)*, Amsterdam, The Netherlands, July 2005, pp. 784–787.

[9] Z. Zhishou, Q. Apostolopoulos, J.and Sun, S. Wee, and W. Wong, “Stream authentication based on generalized butterfly graph,” in *Proc. of IEEE International Conference on Image Processing (ICIP’07)*, San Antonio, TX, September 2007, vol. 6, pp. 121–124.

[10] C. Wong and S. Lam, “Digital signatures for flows and multicasts,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 502–513, August 1999.

[11] R. Merkle, “A certified digital signature,” in *Proc. of Advances in Cryptology (CRYPTO’89)*, Santa Barbara, CA, August 1989, vol. 435 of *LNCS*, pp. 218–238, Springer-Verlag.

[12] J. Park, E. Chong, and H. Siegel, “Efficient multicast stream authentication using erasure codes,” *ACM Transactions on Information and System Security*, vol. 6, no. 2, pp. 258–285, May 2003.

[13] M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and V. Stemann, “Practical loss-resilient codes,” in *Proc. of ACM Symposium on Theory of Computing (STOC’97)*, El Paso, TX, May 1997, pp. 150–159.

[14] M. Rabin, “Efficient dispersal of information for security, load balancing, and fault tolerance,” *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, April 1989.

[15] Y. Park and Yookun Cho, “The eSAIDA stream authentication scheme,” in *Proc. of International Conference on Computational Science and Its Applications (ICCSA’04)*, Assisi, Italy, May 2004, vol. 3046 of *LNCS*, pp. 799–807.

[16] A. Pannetrat and R. Molva, “Efficient multicast packet authentication,” in *Proc. of Network and Distributed Systems Security Symposium (NDSS’03)*, San Diego, CA, February 2003.

[17] A. Habib, D. Xu, M. Atallah, B. Bhargava, and J. Chuang, “A tree-based forward digest protocol to verify data integrity in distributed media streaming,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 1010–1014, July 2005.

[18] S. Wenger, M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, “RFC 3984; RTP payload format for H.264 video,” IETF, February 2005.

[19] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Februrary 1978.

[20] P. Argyroudis, R. Verma, H. Tewari, and D. O’Mahony, “Performance analysis of cryptographic protocols on handheld devices,” in *Proc. of IEEE International Symposium on Network Computing and Applications (NCA’04)*, Cambridge, MA, September 2004, pp. 169–174.

[21] S. Tillich and J. Groschdl, “A survey of public-key cryptography on J2ME-enabled mobile devices,” in *Proc. of International Symposium on Computer and Information Sciences (ISCIS’04)*, Antalya, Turkey, October 2004, vol. 3280 of *LNCS*, pp. 935–944.

[22] NIST, “Federal information processing standards (FIPS) publication 180: Secure hash standard,” National Institute of Standards and Technology (NIST), May 1993.

[23] R. Rivest, “RFC1321; the MD5 message-digest algorithm,” IETF, April 1992.

[24] V. Klima, “Finding MD5 collisions a toy for a notebook,” Cryptology ePrint Archive: Report 2005/075, March 2005.

[25] V. Klima, “Tunnels in hash functions: MD5 collisions within a minute,” Cryptology ePrint Archive: Report 2006/105, April 2006.

[26] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, “Measurement and modeling of the temporal dependence in packet loss,” in *Proc. of IEEE INFOCOM’99*, New York, NY, March 1999, vol. 1, pp. 345–352.