Authentication of Scalable Video Streams With Low Communication Overhead

Kianoosh Mokhtarian, *Student Member, IEEE*, and Mohamed Hefeeda, *Senior Member, IEEE*

***Abstract*—The large prevalence of multimedia systems in recent years makes the security of multimedia communications an important and critical issue. We study the problem of securing the delivery of scalable video streams so that receivers can ensure the authenticity of the video content. Our focus is on recent scalable video coding (SVC) techniques, such as H.264/SVC, which can provide three scalability types at the same time: temporal, spatial, and visual quality. This three-dimensional scalability offers a great flexibility that enables customizing video streams for a wide range of heterogeneous receivers and network conditions. This flexibility, however, is not supported by current stream authentication schemes in the literature. We propose an efficient and secure authentication scheme that accounts for the full scalability of video streams, and enables verification of all possible substreams that can be extracted from the original stream. In addition, we propose an algorithm for minimizing the amount of authentication information that need to be attached to streams. The proposed authentication scheme supports end-to-end authentication, in which any third-party entity involved in the content delivery process, such as stream adaptation proxies and caches, does not have to understand the authentication mechanism. Our simulation study with real video traces shows that the proposed authentication scheme is robust against packet losses, incurs low computational cost for receivers, has short delay, and adds low communication overhead. Finally, we implement the proposed authentication scheme as an open source library called *svcAuth*, which can be used as a transparent add-on by any multimedia streaming application.**

***Index Terms*—Multimedia authentication, multimedia streaming, scalable video streams.**

I. Introduction

MARKET research reports [1], [2] indicate that the demand for multimedia content is rapidly increasing. Furthermore, a growing fraction of the general population is getting accustomed to, even relying on, various multimedia services such as streaming and video conferencing. In these services, multimedia content is typically served over the public Internet, which makes the content vulnerable to malicious manipulation and alteration. This makes the problem of ensuring the authenticity of multimedia content an important and critical issue for various multimedia applications. Ensuring the authenticity means that any tampering with the content by an attacker will be detected. In this paper, we study authentication of scalable video streams. Efficient authentication of these streams is a challenging problem since the authentication schemes must have low computational cost, tolerate packet losses, and incur small communication overhead. Most importantly, the scheme must support the flexibility of scalable streams: it has to successfully verify *any* substream extracted from the original stream.

We focus on recent scalable video streams, which offer great flexibility while incurring much lower overheads than traditional scalable videos [3]. For example, the scalable video coding (SVC) extension of the state-of-the-art H.264/AVC video coding standard, known as H.264/SVC [4], supports adapting a video stream along three scalability dimensions: temporal, spatial, and quality, which provide different frame rates, different spatial resolutions, and different visual qualities, respectively. This flexibility makes it possible to encode a video once and decode it on a wide spectrum of devices, ranging from cellular phones to high-end workstations. These features have attracted significant attention and these streams are being increasingly adopted in many applications, e.g., [5] and [6]. However, these streams cannot be authenticated using schemes designed for traditional scalable videos. The three-dimensional scalability model, which is depicted in Fig. 1(a), is more general than the previous, and much simpler, linear layered models. It allows different combinations of layers along the three dimensions. Even for extracting the same number of layers, there could be several possible *paths* in the scalability cube [4]. For example, a possible substream of Fig. 1(a) is shown in Fig. 1(b) with shaded cubes, in which the first two temporal layers, both of the spatial layers, and a valid subset of quality layers exist. Because of the many possible combinations of layers, previous authentication schemes are not applicable to this model. In addition, there are new useful coding tools designed for scalable coding, such as hierarchical prediction of frames and medium-grained scalability, which require new authentication algorithms. To the best of our knowledge, there are no authentication schemes in the literature that can efficiently support the full flexibility of the three-dimensional scalability model.

The contributions of this paper are as follows.

- We empirically demonstrate the importance of supporting the full flexibility of H.264/SVC video streams and authenticating all layers. We also show that the current authentication schemes in the literature fail to efficiently authenticate recent scalable video streams.
- We propose a new authentication algorithm that supports the full flexibility of three-dimensionally scalable video

Manuscript received December 09, 2009; revised March 23, 2010; accepted May 13, 2010. Date of publication June 01, 2010; date of current version October 15, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Adbulmotaleb El Saddik.

K. Mokhtarian is with Mobidia, Inc., Richmond, BC V6X 2W8, Canada (e-mail: kianoosh@cs.sfu.ca).

M. Hefeeda is with the School of Computing Science, Simon Fraser University, Surrey, BC V3T 0A3, Canada (e-mail: mhefeeda@cs.sfu.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2010.2051410

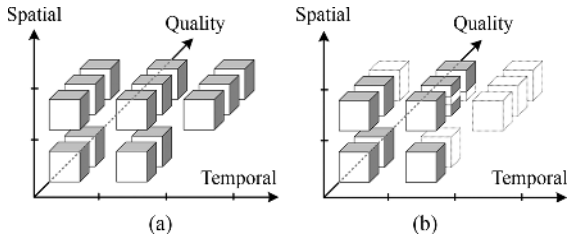

Fig. 1. Three-dimensionally scalable video and a possible substream of that. (a) A video encoded in three temporal, two spatial, and up to three quality layers. (b) A possible substream of Fig. 1(a) with two temporal, both of the spatial, and a valid subset of quality layers.

streams. We analytically show that it guarantees correct verification of any valid substream extracted from the original stream. The algorithm is designed for *end-to-end* authentication of streams. That is, a third-party content delivery network in charge of delivering (and possibly adapting) the streams does not have to be aware of the authentication scheme, which is an important advantage.

- We propose an algorithm for minimizing the communication overhead imposed by the authentication scheme. The algorithm can be of interest in its own right, since it can be used to minimize the communication overhead of other scalable stream authentication schemes as well.
- We implement our authentication scheme for H.264/SVC streams in a prototype called *svcAuth*, which is available as an open source library and can be employed by any multimedia streaming application as a transparent add-on, without requiring changes to the encoders/decoders.
- We conduct a simulation study with real video traces to evaluate different aspects of our scheme. Our results show that our scheme is robust against packet losses, incurs low computational cost and buffer requirement, and is suitable for live streaming as it has short delay. Furthermore, it adds small communication overhead, particularly after using the overhead reduction algorithm.

We summarize the related work and their limitations for supporting SVC streams in Section II. In Section III, we review the structure of SVC streams and the importance of authenticating all layers. Section IV presents our authentication scheme, followed by an algorithm for minimizing the communication overhead in Section V. We evaluate the performance of our scheme in Section VI, and conclude the paper in Section VII.

II. Related Work

Because of its importance, the problem of authenticating video streams has attracted significant attention from academia and industry. We summarize the main ideas in the following. To support adaptation of videos, some works follow content-based methods, whereas others explicitly consider the scalable structure of videos. In content-based methods, such as [7] and [8], the general procedure is to extract a feature set from the video content and sign it. The main challenge is to have the features robust against adaptations, but fragile against malicious manipulations. In these approaches, there is no clear boundary for differentiating valid changes to the content from malicious ones, e.g., [7] relies on threshold numbers provided as input. In addition, it is not clear how significantly one can tamper with the video while preserving the feature set, e.g., [8] uses the energy distribution of I-frames as the feature set, which may not be difficult to preserve while changing the content. In general, content-based methods are more suitable for authenticating video streams that are adapted by traditional stream adaptation techniques such as transcoding and re-compression, in which the adaptations to be performed on the video stream are not exactly known beforehand. An alternative way for making sure the video is not tampered with is that the sender embeds/hides a *watermark* inside the video. The watermark could be a shared secret between the sender and the receivers, such as in [9], or be a digital signature on the video content, such as in [10]. The former case needs to trust all receivers, which is not desirable. In the latter case, there still exists the problem of deciding which features of the content to choose and sign.

On the other hand, several authentication schemes that explicitly consider the scalable structure of video streams have been proposed in the literature for *classic* scalable videos, in which a video consists of a base layer and a number of enhancement layers that progressively improve the video in terms of spatial resolution or visual quality. These schemes generally rely on two cryptographic techniques: hash chaining and Merkle hash trees. Authentication schemes that are based on hash chaining, e.g., [11] and [12], work as follows. First, each enhancement layer of a frame is hashed and its hash is attached to its preceding layer of the same frame. The base layer hash will thus serve as a digest for all layers of the frame, which is called a frame digest. The sequence of frames in the stream can be authenticated by hash-chaining the frame digests, making a two-dimensional hash chaining scheme, or by using any other authentication technique for nonscalable packet streams, such as [13] and [14]. Dropping higher enhancement layers has no impact on the authentication of the remaining layers.

Authentication schemes that are based on Merkle hash trees, e.g., [15] and [16], work as follows. The enhancement layers of a video frame are hashed, and the hash values are arranged as leaves of a tree. Each interior node of this tree consists of the digest of its children. The root of the tree represents the frame digest. Due to the collision-free property of the hash function, the whole set of layers represented by the leaves is authenticated if the root of the tree is successfully verified. The sequence of frames can be authenticated by building another Merkle tree over frame digests. A similar procedure is employed in [17] for MPEG-4 scalable videos by specifically employing MPEG-4's tree-like structure in building the hash tree. In either case, upon removal of some layers, a receiver may need some extra digests for verifying the remaining layers, i.e., for reconstructing the root digest of the hash tree. This means that an adapting proxy on the delivery path must understand and be compatible with the authentication scheme, which may not be desirable. A high level framework for secure scalable streaming (SSS) is presented in [18] which focuses on the encryption of scalable videos and on enabling the proxies to perform adaptations without requiring decryption. For authentication, however, the SSS framework recommends using Merkle hash trees, which requires the cooperation of proxies. Scalable authentication based on hash trees [15], [17], [18] can be employed for end-to-end authentication

if we embed in each layer i all information needed to authenticate the first i layers. This, however, significantly increases the communication overhead [19].

In summary, the current authentication techniques for scalable streams are designed for traditional, and much simpler, linear layered videos in which scalability is provided along one dimension and the layers in that dimension are cumulative. Even for these videos, they may incur a significant communication overhead if the stream is providing more than a limited flexibility, i.e., it is encoded in several layers. An efficient and end-to-end authentication service for modern scalable streams cannot be provided by current authentication schemes or by simple extensions of them such as forming a hash chain or tree on each scalability dimension. For example, if previous schemes are applied on a temporal scalable stream, frames in a temporal layer of the stream have to be all kept or all dropped together, since these schemes operate on a layer basis. In addition, applying previous techniques to authenticate quality enhancement packets may result in unverifiability of many of the received packets, because they are not necessarily dropped in a cumulative manner. We take into account the complete scalability structure of three-dimensional scalable streams to authenticate all their valid substreams. We also propose an additional algorithm for significantly reducing the communication overhead.

III. Background

A. Overview of H.264/SVC

The recently standardized H.264/SVC video coding standard [4] adds scalability to the widely used H.264/AVC video coding technique [20]. In addition to generating highly flexible video streams, H.264/SVC significantly outperforms previous scalable coding techniques in terms of coding efficiency [3]. That is, at the same bitrate, it provides a higher visual quality. H.264/SVC supports temporal, spatial, and quality scalability at the same time.

Temporal scalability is achieved by employing a hierarchical prediction structure among video frames belonging to the same group-of-pictures (GoP), as shown in Fig. 2. In this structure, frames of higher temporal layers can only be predicted from lower temporal layers. A GoP consists of one frame in the temporal base layer, which is generally coded as P-frame, and several hierarchically coded B-frames that are located between the temporal base layer frames. In the spatial scalability of SVC, a spatial layer s of a frame can be predicted from the sth spatial layer of some other frames (in lower temporal layers), as well as lower spatial layers in its own frame. For providing quality scalability, there are two different possibilities. The first one follows the spatial scalability structure, but assigns the same resolution and different quantization parameters to layers. This produces a coarse-grained scalable (CGS) video with limited number of quality layers. A finer granularity can be provided by the second possibility, which uses medium-grained scalability (MGS) coding to divide a single CGS quality layer into multiple sub-layers, which are referred to as MGS layers. This is done by partitioning the residual DCT coefficients of a CGS layer into multiple MGS layers. A stream can be truncated at any CGS or MGS layer. In addition, some packets of an MGS layer can be discarded, while the remaining ones can still be decoded to improve quality. Packet discarding can be done in different ways, depending on the bitstream extraction process [21]. H.264/SVC allows up to seven temporal, eight spatial, and 16 quality layers [4].

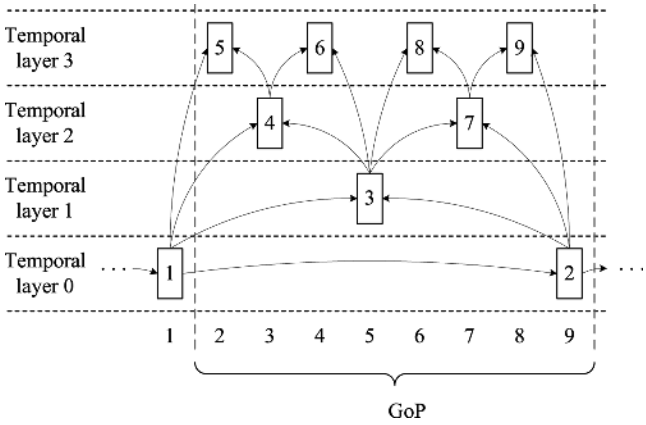

Fig. 2. Hierarchical prediction structure of H.264/SVC temporal scalability. Arrows represent prediction. Numbers listed in the bottom row show the displaying order, while numbers inside frames show the coding order.

In H.264/SVC, the coded video data and other related information are organized into network abstraction layer (NAL) units, which we alternatively refer to as a *video packet* or a *truncation unit* because these are the smallest units that can be truncated from an SVC stream. Each NAL unit has *temporal_id*, *spatial_id*, and *quality_id* field in its header, which identify to which temporal, spatial, and quality layer to which the NAL unit belongs. NAL units can be video coding layer (VCL) units, which contain the coded video data, or non-VCL NAL units, which contain associated additional information. The structure of an SVC video stream can be summarized as follows. The stream is a sequence of GoPs. Each GoP consists of a number of video frames, each of which belongs to a certain temporal level. Each frame, in turn, contains multiple spatial layers. A spatial layer then includes a few CGS quality layers, each one possibly partitioned into several MGS layers. Each MGS layer can be divided into multiple NAL units, to which we alternatively refer as *truncation units* or *video packets*, because these are the smallest units that can be truncated from an SVC stream. A video packet can be transmitted as more than one network packet, but without loss of generality, we assume that a video packet fits in a network packet, i.e., it is encoded at a size not exceeding the desired packet size for network transmission; otherwise, the video packet is divided into multiple video packets.

B. Importance of Protecting All Layers

We aim at authenticating every video packet in a received substream. For this purpose, we need to protect every packet from potential malicious manipulations. One might argue that this may not be necessary and it could suffice, for example, to authenticate every two or more packets together. In this case, if both packets are received, they can be verified, but an individual one cannot because we need both packets to re-compute and verify the given digest. The argument that considers this

Fig. 3. Results of tampering with one MGS packet. (a) Original picture. (b) Result of tampering with the first MGS layer (after four frames). (c) Result of tampering with the second MGS layer (after four frames).

unverifiability to be insignificant is based on the conjecture that exploiting a few single packets which are typically in enhancement layers can only have a quality enhancement/degradation effect, and thus cannot lead to a successful manipulation of the video content.

We show in the following that it is necessary to verify all portions of the received data and discard the unverifiable packets. In H.264/SVC for quality scalable coding with MGS, the highest quality picture of a frame is often used for motion compensation at receiver side [4]. Consequently, a change in the small unprotected portion of some highest few MGS layers will fast propagate through frames of a GoP, since those layers are used as prediction references for other frames. To confirm the importance of authenticating every packet, we empirically demonstrate a simple attack on an SVC video by manipulating one enhancement packet only. The original video is a short sequence consisting of ten frames which shows the face of a man with no motion, as depicted in Fig. 3(a). The video is encoded as one base and one CGS quality enhancement layer, whose quantization factors are 30 and 0 (no quantization), respectively. Transform coefficients of the CGS enhancement layer are further divided into three MGS layers, consisting of 4, 6, and 6 coefficients in the zigzag traversal order of the 4×4 coefficient table. Each MGS layer consists of a few small video packet. The tampering is done by modifying a small portion of the *enhancement data*, trying to add a simple scratch to the face. We first assume we are allowed only to tamper with up to one packet of the first MGS layer, as the base layer is protected. We gradually modify over a few successive frames one packet of the first MGS layer in each frame. After only four frames, we could make a meaningful alteration on the video by creating a scratch on the face of the man, as shown in Fig. 3(b). In another attack, we assume that packets of the base layer and the first MGS layer are protected, and we (as attacker) are allowed only to modify a single packet in the second MGS layer. Again after a few frames, we were able to meaningfully tamper with the video content, as shown in Fig. 3(c).

This experiment highlights the risk of leaving any portion of the video data unprotected. Notice that we tried to tamper with the video in a *very simple* way by replacing the small number of unprotected transform coefficients with those of the desired picture. Now consider a real attacker who chooses the target video content carefully and/or employs more complicated image processing techniques for replacing unprotected coefficients. Clearly this attack may successfully make significant changes to the video content.

IV. Proposed Scheme

In this section, we first present our scheme for authenticating SVC streams, and then analyze its security and complexity.

A. Proposed Authentication Scheme

At a high level, the proposed authentication scheme works as follows. First, the content provider prepares the additional information needed for verification, and attaches them to the stream. Each receiver either receives the whole or a subset of the original stream, along with the corresponding authentication information. The task of substream extraction may be carried out by stream adaptation proxies belonging to the delivery network, which do not have to understand the authentication scheme. The authentication information is transparent to these proxies; it is attached to specific NAL units in a video format-compliant manner. Some packets of the stream may be lost during transmission. Unlike loss of a video packet that can be tolerated to some extent by error concealment techniques, loss of the authentication information may have a serious effect: some layers cannot be verified and thus cannot be used, although they are successfully received. We therefore need to appropriately protect the authentication information against loss. If the video is being transmitted over the Internet, where bursts of packets can be lost, it is a common practice to distribute video data over network packets in an interleaved manner [22], which changes the loss pattern from bursty to random. Relying on such packetization technique, we assume packet losses have a random pattern.

A high level pseudocode of the proposed authentication scheme is given in Fig. 4. This algorithm is executed by the content provider to protect video streams. We give a brief description of the algorithm, and refer the interested reader to [23] for more details. First, within each spatial layer of each frame, quality layers are authenticated. Since CGS quality layers are encoded the same way as spatial layers and follow the same dependency structure, we treat them as spatial layers. For authenticating MGS quality layers, we note that quality packets of MGS layers are not necessarily extracted in a specific order [21]. Thus, we authenticate MGS layers of each CGS/spatial layer and we compute the *CGS/spatial layer digest*

SVC_Authenticate

AuthenticateStream (*GoPs*[], n, k, α[])
1. *GoPblocks* = DivideToBlocks(*GoPs*, n)
2. **for** *block* $\in$ *GoPBlocks* **do**
3. X = null //contains GoP digests
4. **foreach** *GoP* $\in$ *block* and $i = 0$ **to** $n-1$
5. $X = X$ || AuthenticateGoP(*GoP*, k, α)
6. Embed { X||*Sign*($h(X)$) } in layer $\{0,0,0\}$ of all of the GoPs in *block*

AuthenticateGoP (*GoP*, k, α[])
1. T = *GoP.NumTemporalLayers*
2. $n_{t\in[0,T-1]}$ = *GoP.NumFramesAtLayer*[t]
3. $n_1 = 2$
4. *ToEmbed*[] = MakeEmptyArray(n_T)
5. **for** $t = T-1$ **to** 2 **do**
6. X = null //contains frame digests
7. **for** $i = 0$ **to** $n_t - 1$
8. X = AuthenticateFrame(*GoP.FramesAtLayer*[t].*get*(i), k, *ToEmbed*[i]) || X
9. x[] = DivideToPieces(X, $\lceil \alpha[t] \times n_{t-1} \rceil$)
10. *ToEmbed*[] = FEC_encode(x, n_{t-1})
11. f_0, f_1 = *GoP.FrameAtLayer*[0], [1]
12. y = AuthenticateFrame(f_1, k, *ToEmbed*[1])
13. x = AuthenticateFrame(f_0, k, *ToEmbed*[0])
14. **return** $\{x \,||\, y\}$

AuthenticateFrame (*frame*, k, W)
1. S = *frame.NumSpatialAndCGSlayers*
2. *ToEmbed*[] = MakeEmptyArray(S)
3. **for** $s = S-1$ **to** 0 **do**
4. *layer* = *frame.layer*[s]; Q = *layer.NumMGSes*
5. F'[] = MakeEmptyArray(Q)
6. **for** $q = 0$ **to** $Q-1$
7. p[] = *layer.MGS*[q].*Units()*
8. $F'[q] = h(p[0]) \,||\, \ldots \,||\, h(p[\text{size}(p)-1])$
9. Embed $F'[q]$ in *layer.MGS*[q] in k copies
10. $F = h(F'[0]) \,||\, \ldots \,||\, h(F'[Q-1]) \,||\, ToEmbed[s]$
11. **if** $s = 0$ **then**
12. Embed $\{F||W\}$ in *layer.MGS*[0] in k copies
13. **return** $h(F||W)$
14. Embed F in *layer.MGS*[0] in k copies
15. *ToEmbed* [*layer.HighestRef()*] $||= h(F)$

Fig. 4. Proposed authentication scheme.

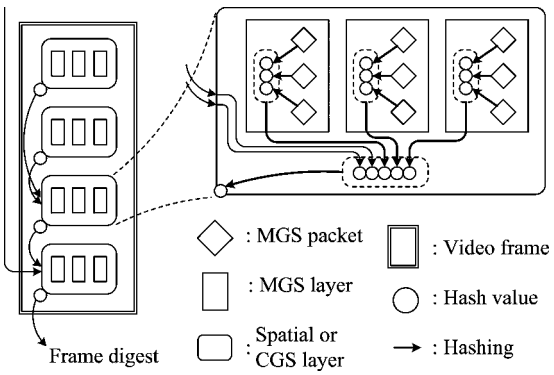

Fig. 5. Authenticating a video frame.

in a two-level hierarchy, as shown in Fig. 5 and lines 4–10 of the "AuthenticateFrame" procedure in Fig. 4. Then, in each frame, spatial/CGS layers and their digests are authenticated by attaching the digest of each layer to its highest reference layer. This is because the intra-frame layer dependency in SVC, unlike previous scalable videos, does not have to be linear and is in general a directed acyclic graph (DAG). In this step, the *frame digest* is created.

To protect the authentication information of quality and spatial layers against loss, we replicate them in two or more (k) packets, rather than FEC-coding them; note that there can be several quality and spatial layers in each frame, and that many

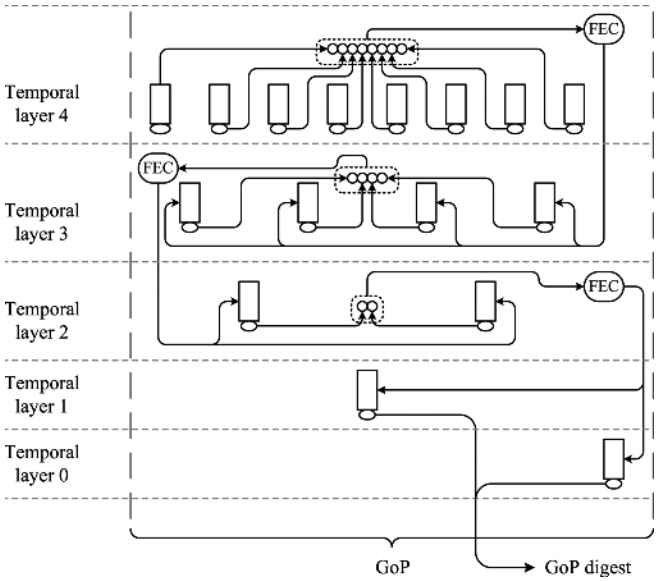

Fig. 6. Authenticating frames of a GoP.

FEC operations per each frame impose a high computational cost. The value of k can balance a tradeoff between loss tolerance and communication overhead, and we show in the evaluation section that only two copies can be enough.

The next step authenticates the frames of each GoP according to their temporal levels using the "AuthenticateGoP" procedure. This step is illustrated in Fig. 6 and the "AuthenticateGoP" function in Fig. 4; GoPs with non-dyadic structure can be authenticated in a similar manner. Having obtained the frame digests in the previous step, for each temporal layer t, we concatenate the digests of all frames of layer t, and distribute the concatenation over the frames of level $t-1$. This is done using FEC coding so that we can verify layer-t frames even when the authentication information of some frames in layer $t-1$ is lost. We then create a *GoP digest* out of a GoP as shown in Fig. 6 and the code. Then, in the last step, the whole stream of GoPs is authenticated by dividing the sequence of GoPs into blocks, and digitally signing the *block digest* computed on each block.

The verification process proceeds in the same way as generating the authentication information. Given a valid substream and its authentication information, a receiver recomputes spatial layer, frame, GoP, and block digests from the reconstructed video. In case of any mismatch between the recomputed digest and the digest provided by the server in the substream, the mismatching part of data, such as a video frame, is marked as unauthentic and is discarded. The remaining part of the substream is known as authentic if and only if the digital signature of the corresponding block is successfully verified.

B. Security Analysis

We prove that the proposed scheme authenticates any valid substream extracted from the original stream, provided that the underlying cryptographic primitives (hash function and digital signature) are secure on their own. The scheme enables a client to assure that the received content has gone under no manipulation, such as changes in the contents of any video frame, frame insertion, frame reordering, frame removal, or any other type

of tampering—frame removals due to frame rate reduction are legitimate as long as they are compliant to the SVC standard, which cannot be used for malicious purposes such as cropping a number of consecutive frames.

Theorem 1: The proposed scheme ensures the authenticity of any substream extracted from the original SVC stream.

Proof: Recall the hierarchical structure of an SVC video stream, whose levels consist of GoPs, temporal layers, frames, spatial/CGS layers, MGS quality layers, and finally, video packets. We prove the authenticity of any valid substream in a bottom-up manner in this structure. We first show that the authenticity of any valid subset of quality layers can be successfully verified if the corresponding spatial layer digest is authentic. The procedure continues similarly for the digests of spatial layers, frames, GoPs, and GoP blocks. Finally, successful verification of the digital signature of a GoP block is shown to be the *necessary and sufficient* condition for authenticity of any valid substream, which proves the theorem.

Step 1: Authenticity of Quality Layers: We first analyze the deepest level of the scalability hierarchy, which corresponds to quality layers. We prove that any valid subset of video packets of a spatial/CGS layer, given the authentication information of the packets and the digest of the corresponding spatial/CGS layer, are authentic iff the spatial/CGS layer digest is authentic. The forward direction of the statement, i.e., no manipulated/inserted video packet is accepted, is proven as follows. Since the spatial/CGS layer digest $h(F)$ (where F is obtained in line 10 of the code) is authentic and $h(\cdot)$ is a collision-free hash function, the concatenation $F = h(F'[0])||\dots||h(F'[Q-1])||ToEmbed[s]$ is also authentic. This proves the authenticity of all $h(F'[i])$ values, which are present in the received substream since they are included in the authentication information of the spatial/CGS layer; they enable a receiver to reconstruct F even if it has received no packet from some of the quality layers. For each MGS layer q from which at least one packet is received, the attached $F'[q]$ value can be verified since $h(F'[q])$ is authentic, meaning that no $F'[q]$ value can be forged or inserted. Since an MGS layer q's $F'[q]$ equals $h(p[0])||\dots||h(p[size(p)-1])$ (line 8 of Fig. 4), the authenticity of $F'[q]$ proves the authenticity of $h(p[i])$ values, and accordingly, the packets $p[i]$ that exist in the substream. Any change to the contents of a packet will result in a change in the corresponding $h(p[i])$, $F'[q]$, $h(F'[q])$, F, and $h(F)$ values, and no change to a packet $p[i]$ can preserve these values since $h(\cdot)$ is collision-free. Moreover, no video packet can be inserted, as the integrity of $F'[q] = h(p[0])||\dots||h(p[size(p)-1])$ is already proven. This shows that no content other than a subset of authentic packets can pass the verification process. The backward direction of the statement, i.e., no original subset of packets is rejected, is clear. First, one can reconstruct the spatial/CGS layer digest out of any valid subset of quality packets. Moreover, out of an authentic subset of packets, the same $h(F)$ value as the one provided by the content provider is calculated by a receiver. Hence, any subset of packets will pass the verification iff it is authentic.

Step 2: Authenticity of Spatial/CGS Layers: Having authenticated each of the received or partially received spatial/CGS layers, we now show that in a valid subset of spatial/CGS layers of a frame, each layer that has a path of digests to the base layer is authentic iff the frame digest is authentic. Each layer that does not have this path, which in turn has a path to another frame digest, is authentic iff the digest of that frame is authentic. This statement is obvious for the base layer, since a frame digest is actually the digest of the base layer along with its attached authentication information. Any other spatial/CGS layer has its digest embedded in a lower layer, which makes a biconditional relationship between the authenticity of the two layers. Hence, this relationship is created between frame digests and all of the layers: from each layer there is a path to the base layer of the same frame, or in the unlikely case that a layer is not predicted from any lower layer, there is a path from it to the base layer of another frame. Hence, the authenticity of frame digests is necessary and sufficient for authenticity of any valid subset of spatial/CGS layers.

Step 3: Authenticity of Video Frames: We now show the biconditional relationship between the digest of a GoP and the digests of its frames. Recall that a valid subset of frames consists of all video frames of temporal layers $1, 2, \dots, T-1$, possibly a fraction of frames at temporal level T, and no frame from levels $T+1$ and higher. In the GoP authentication procedure, if enough number of frames of level t are received by a client, the concatenation of frame digests of level $t+1$ can be verified, which assures for any subset of frame digests of level $t+1$ that all digest have integrity and no frame (with its digest) is inserted. Therefore, the authenticity of the frame digests of each temporal level is necessary and sufficient for the authenticity of those in the next temporal level. This enables the authenticity of the GoP digest to propagate from the temporal base layer to all of the received temporal layers and assures the authenticity of all frame digests.

Step 4: Authenticity of GoPs: A GoP block digest, which is digitally signed, is nothing but a hash value over the digests of the GoPs of the block. This clearly shows the bidirectional dependency between the authenticity of GoP digests and that of the GoP block digest; it prevents any changes to a GoP digest or insertion/removal of a GoP.

According to the above steps, the correctness of Theorem 1 can now be readily proven as follows. The stream consists of independently signed blocks of GoPs. According to steps 4 through 1, successful verification of the digital signature of a GoP block is *necessary and sufficient* for the authenticity of all packets of an extracted substream. □

C. Complexity Analysis

1) Computation Cost: We calculate the computation cost in terms of the number of hash computations, FEC encoding/decoding operations, and digital signature generations/verifications per each GoP block. Since there is one hash computation per each GoP, frame, spatial/CGS layer, MGS layer, and truncation unit, the total number of hash operations is $n+F+S+Q+P$, where F, S, Q, and P represent the total number of frames, spatial layers, quality layers, and video packets in a block of n GoPs. Since a hash operation can be performed very fast compared to decoding of a video packet, we can practically ignore the computation cost of the above hashes. The total number of

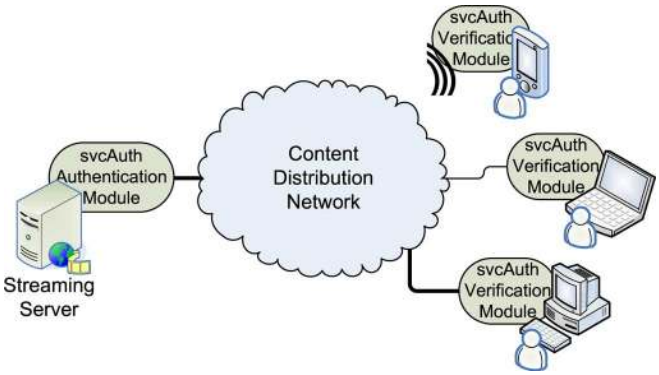

Fig. 7. Deployment of svcAuth Authentication and Verification Module.

FEC operations per block of n GoPs equals the number temporal layers times n, which is typically a few per second. This cost is negligible, too, as there are fast FEC coding algorithms to be employed. For example, Tornado codes for FEC coding use only XOR operations and operate in linear time of the input block. The dominant computation cost is that of digital signature operations, the number of which in our scheme is one per n GoPs. This cost is still low in our scheme as we show in the evaluation section.

2) Communication Overhead: We denote by s_{hash} the size of a hash, by s_{sig} the size of a digital signature, by k the number of copies of authentication information of quality and spatial/CGS layers, and by α the FEC factor for authenticating temporal layers, i.e., the fraction of pieces enough for reconstructing the original data. The communication overhead of our scheme for each block of n GoPs, C, is as follows:

$$C = \left(k \times (P + Q + S) + \frac{F}{\alpha} + n\right) \times s_{hash} + s_{sig}. \quad (1)$$

This communication overhead can become non-negligible for highly flexible scalable streams that provide many possibilities for extracting substreams. We propose an algorithm to reduce this communication overhead in Section V.

D. svcAuth Library

We have implemented the proposed authentication scheme for H.264/SVC streams in a prototype called *svcAuth*.[1] svcAuth is available as an open source library implemented in Java to support portability across different platforms. It can be employed by any video streaming application as a transparent software add-on, without requiring any change to the encoders/decoders. As illustrated in Fig. 7, we add an Authentication Module to the provider side, which performs postprocessing of the encoded stream, and creates and embeds the authentication information in the stream. At the receiver, we add a Verification Module which verifies the received stream using the information embedded in it, and passes the verified stream to the player. Note that receivers that do not have the svcAuth Verification Module can still decode streams, since svcAuth is transparent.

[1]The latest version of svcAuth and the related documentations can be found at http://nsl.cs.sfu.ca/wiki/index.php/svcAuth.

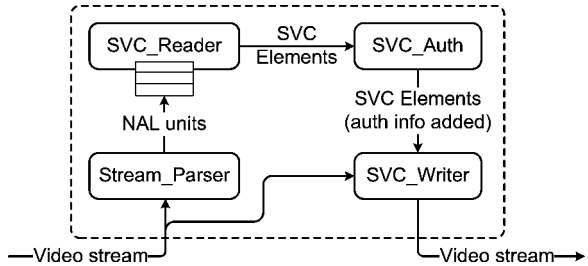

Fig. 8. svcAuth Authentication Module.

svcAuth is a rather large library with over 7000 lines of code, including sample secure streaming usages and utilities for working with SVC streams. Here we briefly review the svcAuth Authentication module. This module, which is placed after the video encoding process and before transmission, is shown in Fig. 8 and operates as follows. The video bitstream is first parsed by the Stream_Parser component, which extracts NAL units from the bitstream, parses their headers, and delivers them as logical objects to the SVC_Reader component. The SVC_Reader component determines the structure of the SVC stream using the NAL units. For this purpose, as shown in the figure, it needs to buffer a number of NAL units, e.g., to determine the last NAL unit of the current video frame which is done by detecting the first NAL unit of the next frame. The SVC_Reader component outputs a logical view of the stream as GoPs, frames, and different types of layers. We refer to these entities as SVC Elements. Each SVC Element of this structure in the logical view returned by SVC_Reader contains an array of authentication information messages, which is initially empty. These arrays are filled by the SVC_Auth component. The SVC_Auth component implements the algorithm described in Fig. 4. It takes as input a block of n GoPs, computes the required authentication information, and adds them to the SVC Elements of those n GoPs.

The info-added SVC Elements are delivered to the SVC_Writer component, which converts back the logical structure to a raw bitstream. This is done by encapsulating the authentication information as appropriate NAL units and inserting them in specified locations in the original bitstream. For this purpose, we exploit the supplemental enhancement information (SEI) NAL units of SVC [4]. These NAL units (NAL unit type 6) are non-VCL NAL units that can carry auxiliary information related to decoding, displaying, or other processing operations of the video. An SEI NAL unit can contain one or more SEI Messages. To attach some information to a specific layer, we embed it in an Unregistered User Data SEI Message, relate it to the desired temporal/spatial/quality layer by encapsulating (nesting) it in a Scalable Nesting SEI Message [24], and we finally encapsulate the result in an SEI NAL unit.

V. Reducing the Overhead

The amount of authentication information that needs to be added to an SVC stream can be non-negligible if the stream is providing a high flexibility, i.e., when there is a large number of truncation points. We refer to this information as the communication overhead of the scheme. Note that the non-negligible

communication overhead is not specific to our scheme and will be suffered from by other authentication schemes as well. This is because a hash value is needed for each truncatable unit of data, and the number of these units grows with flexibility of scalable streams.

In order to reduce the communication overhead, we compute one hash value for a *group of truncation units* rather than for each unit. Note that the atomic unit of authentication would then be a group of units, i.e., partial groups cannot be authenticated. That is because unverified video packets of a partially received group have to be discarded, as shown in Section III-B. Accordingly, by aggregating truncation units into groups, on one hand we reduce the communication overhead of hashes, and on the other hand, we may also reduce the flexibility of the stream. That is, some receivers may receive a smaller number of layers than the number of layers they would have received if there was no grouping. In this section, we propose an optimal algorithm for grouping the truncation units in order to minimize this twofold overhead. The application of this algorithm is not limited to our authentication scheme, and it can be used with other scalable video authentication techniques in the literature, such as those for traditional scalable streams, e.g., [11] and [12], for optimizing their communication overhead. Thus, this overhead reduction algorithm can be of interest in its own right.

Since only a complete group of units can be authenticated, in order for the grouping process to perform well, the selection of quality truncation units in the substream extraction process should not be arbitrary. Otherwise, it is possible that in a received substream, for example, all groups are received partially, none of which is then verifiable. Thus, we can embed (during encoding) the quality extraction information inside the stream so that stream adaptation proxies follow the same extraction procedure. The H.264/SVC standard provides specific means for signaling this information in the streams [21]. This enables us to assume an ordering on truncation units, and cases such as the aforementioned example will not happen. That is, at most one group of units can be received partially. Note that this assumption does not restrict the flexibility of SVC quality scalable streams, as the discarding of quality packets and layers can still be non-cumulative. However, grouping units and not allowing the use of partial groups limits this flexibility, since it reduces the number of possible truncation points. Clearly, there is a tradeoff between the flexibility of streams and the communication overhead imposed by the authentication scheme, and this tradeoff is controlled by the size of truncation unit groups.

Our grouping algorithm works on a frame basis and divides the truncation units of each frame into a number of groups. Note that the number of truncation units of a frame can grow to hundreds as we show for actual videos in the evaluation section. The algorithm uses the communication overhead as a cost function and employs dynamic programming to determine the grouping of truncation units that minimizes this cost.

Suppose there are n quality truncation units in a frame f, and denote them by $u_1, u_2, \ldots, u_n$. These units are to be partitioned into a number of, say h, groups as shown in the equation at the bottom of the page. The algorithm finds the optimal number h^* of (non-empty) groups, and divides the n units into h^* groups. The cost of grouping $u_i, \ldots, u_j$ of frame f is $c_f(i,j)$, which consists of the streaming bitrate that users lose when we omit truncation points at $u_i, \ldots, u_{j-1}$, as well as the overhead of an s-bit hash value that will be designated to the group. For calculation of the overall streaming bitrate that users lose by grouping u_i through u_j, we can also take into account the distribution of user bandwidths, leading to a more accurate calculation of $c_f(i,j)$. Calculation of $c_f(i,j)$ based on the truncation units of a frame and user bandwidth distribution is discussed in Section V-A. The cost of multiple groups equals the sum of the group costs. It could be possible, although unlikely, for some highest-layer truncation units not to be in any being-hashed group. Let $c'_f(i)$ denote the cost for not having $u_i, \ldots, u_n$ in any hashed group, i.e., these units are preferred to be ignored.

The proposed dynamic programming algorithm is based on solving subproblems (h,l), which represent the grouping of the first l units into h groups $(1 \le h \le l \le n)$ where the hth group exactly ends at the lth unit. The minimum cost for the (h,l) subproblem is kept in a matrix $A_f[h,l]$ for frame f. For the first row of the matrix A_f, which refers to the case where only one hash value is to be calculated for the frame, $A_f[1,l]$ indicates that units $\{u_1, \ldots, u_l\}$ will constitute the only. The rest of the rows of the matrix A_f are calculated as follows:

$$A_f[h,l] = \min \begin{cases} A_f[h-1,l-1] + c_f(l,l) \\ A_f[h-1,l-2] + c_f(l-1,l) \\ \vdots \\ A_f[h-1,h-1] + c_f(h,l). \end{cases} \tag{2}$$

In (2), the ith row $(1 \le i \le l-h+1)$ represent the case where the hth group consists of the $l-i+1$st through the lth units. Thus, the cost equals the cost of having the previous $l-i$ units in $h-1$ groups, which is $A_f[h-1,l-i]$, as well as the cost of grouping the $l-i+1$st through the lth units together, which is $c_f(l-i+1,l)$. To maintain how each of the subproblem solutions is constructed, we keep another $n \times n$ matrix B_f where $B_f[h,l]$ represents the number of units in the hth group when grouping l units in h groups, i.e., the row index in (2) that led to the minimum cost. Therefore, the solution to subproblem (h,l) consists of $B_f[h,l]$ units in the hth group, $B_f[\,h-1, l-B_f[h,l]\,]$ units in the $h-1$st group, and so on.

$$\underbrace{u_1 \cdots u_{x_1}}_{x_1} \mid \underbrace{u_{x_1+1} \ldots u_{x_1+x_2}}_{x_2} \mid \ldots \mid \underbrace{u_{x_1+\cdots+x_{h-1}+1} \ \ldots \ u_n}_{x_h}$$

Having filled the cost matrix A_f with the minimum cost of all valid subproblems, we now calculate the minimum cost $\hat{A}_f[h]$ of the optimal solution for grouping all units into h groups. If we were sure that all units belong to some group, i.e., no unit is decided to be ignored, $\hat{A}_f[h]$ would have been equal to $A_f[h,l]$. However, note that it is possible, though unlikely, that the optimal solution leaves some units $u_{x+1}, u_{x+2}, \ldots, u_n$ out of any group, e.g., hashing those truncation units does not worth its communication overhead. Thus, $\hat{A}_f[h]$ is calculated as

$$\hat{A}_f[h] = \min\{A_f[h,x] + c'_f(x+1) \mid h \le x \le n\}. \quad (3)$$

To obtain the final optimal solution, we first determine the best number of groups h^* as $h^* = \arg\max_h\{A'[h]\}$. Then, to construct those h^* groups, we first note that the number of truncation units considered in those h^* groups, denoted by x^* ($1 \le x^* \le n$), is actually the x value that minimized (3). In the optimal solution, the number of units in the ith group ($1 \le i \le h^*$), denoted by $X[i]$, is obtained from $X[h^*]$ to $X[1]$ as follows. Let p be a pointer representing the number of units in the remaining groups, and thus initially assigned as $p = x^*$. According to the construction of the matrix B, we have $X[h^*] = B[h^*, p]$. Having put $X[h^*]$ units in the h^*th group, we update the pointer p as $p = p - X[h^*]$ and obtain the number of units in the second last group as $X[h^*-1] = B[h^*-1, p]$. Updating p as $p = p - X[h^*-1]$, we get $X[h^*-2] = B[h^*-2, p]$, and similarly are obtained the rest of the $X[i]$ values. The values h^* and $X[i]$ ($1 \le i \le h^*$) form the optimal grouping solution.

The following theorem shows the optimality and complexity of the proposed algorithm.

Theorem 2: The proposed dynamic programming solution for aggregating truncation units of a frame into groups finds the minimum-cost grouping. It has a memory requirement of $O(n_{\max}^2)$, where $n_{\max}$ is the maximum number of truncation units in a frame, and it has a time complexity of $O(n_{\max}^3)$.

Proof: The key point of the proof is the optimality of matrix A_f that keeps the minimum costs of subproblems and is calculated as (2). We show this optimality by showing that the problem exhibits the *optimal substructure* property: an optimal solution to the problem, which is h^* optimally formed groups of $X[1], X[2], \ldots, X[h^*]$ truncation units, contains within it optimal solutions to subproblems, which is optimal formation of the first h ($1 \le h \le h^*$) groups using the first l ($h \le l \le n_f$) units. This is true because in the final optimal solution, if the grouping of the first $l = \sum_{i=1}^{h} X[i]$ units in the first h groups is not optimal, we simply replace this part with an optimal sub-solution and obtain a smaller cost, leading to a contradiction. Given the optimality of the matrix A_f, the optimality of $\hat{A}_f[h]$ values (3) immediately follows.

The memory required by the algorithm for a frame f is that of two $n_f \times n_f$ matrices, A_f and B_f, and one vector $\hat{A}_f$ of length n_f. Thus, the memory required is of $O(n_{\max}^2)$, where $n_{\max}$ is the maximum number of truncation units in a frame and hardly reaches 1000. The required memory is thus a small amount.

The algorithm calculates the upper triangular half of matrices A_f and B_f, i.e., $A_f[i,j]$ ($i \le j$). For each $A_f[i,j]$ and $B_f[i,j]$ together, $j - i + 1$ comparisons are performed. Hence, the running time in terms of the total number of comparisons Ω_f for a frame f is

$$\begin{aligned}\Omega_f &= \sum_{i=1}^{n_f}\sum_{j=i}^{n_f}(j-i+1) \\ &= \frac{1}{6}n_f(n_f+1)(2n_f-5) + n_f^2 + n_f. \end{aligned} \quad (4)$$

This number of iterations makes the running time of the algorithm $O(n_{\max}^3)$, though its constant factor is considerably small (1/3). □

We have run our algorithm on a commodity PC for high bitrate video streams (> 10 Mbps) with 500-byte truncation units, which makes 30 to 300 units per frame, and it could easily perform the grouping faster than the frame rate of the stream, e.g., two to three times faster in our experiments for 30-Hz videos at 10+ Mbps.

A. Cost Function

This section describes the details of computing cost values $c_f(i,j)$, which is the cost of grouping the ith through the jth truncation units of frame f together. This is done based on the bitrate of the video up to each truncation unit, and optionally, the distribution of users bandwidths.

The cost $c_f(i,j)$ consists of two parts: the loss of truncation points at truncation units $u_i, \ldots, u_{j-1}$, as well as an s-bit overhead of the hash value being designated to the group of units. To have an accurate calculation of these costs for a frame f, we need the time duration $d(f)$ it takes for f to be downloaded, which may not be equal among different frames. $d(f)$ is calculated based on the bitrate of the video and the sizes of other frames as we see shortly. Let $b(u_i) = size(u_i)/d(f)$ denote the bitrate of a unit u_i, $b_i = \sum_{x=1}^{i} b(u_x)$ the bitrate of the frame under process when it contains up to unit u_i, and $\hat{s}_f = s/d(f)$ the bitrate overhead of a hash value for frame f. The cost function, which is reflecting the communication overhead, can be thought of as the bandwidth waste of all clients, that is, the difference between the bitrate of the authenticated video that a client will receive and the bitrate of the video if there was no authentication information attached and no grouping performed. To calculate these costs, we can also take into account *a priori* knowledge about the streaming scenario in terms of the distribution of client bandwidths $\mathcal{Z}$, leading to a more efficient overall cost minimization. Clearly, a uniform distribution can be assumed if no such *a priori* knowledge can be obtained. The cost $c_f(i,j)$ for a group of units consists of the cost of a hash value and that of making clients with bandwidths in $[b_i, b_j)$ receive a video bitrate of b_{i-1} rather than b_i, b_{i+1}, …, or b_{j-1}. The cost $c_f(i,j)$ is calculated as

$$c_f(i,j) = \hat{s} + \sum_{x=i}^{j-1}\left((b_x - b_{i-1}) \times \int_{b_x}^{b_{x+1}} Pr(\mathcal{Z}=z)\,dz\right). \quad (5)$$

Similarly the cost $c'_f(i)$ of not having units $u_i, \ldots, u_n$ in any hashed group is calculated as

$$c'_f(i) = \sum_{x=i}^{n_f}\left((b_x - b_{i-1}) \times \int_{b_x}^{b_{x+1}} Pr(\mathcal{Z}=z)\,dz\right). \quad (6)$$

Calculation of the cost values $c_f(i,j)$ impacts the running time of the algorithm negligibly, since they can be calculated for each frame prior to running the grouping algorithm for the frame. For each frame f, a lookup matrix C_f is created, which is discarded after the vector $\hat{A}_f$ of the frame is obtained. The matrix is constructed as $C_f[i,j] = c_f(i,j)$. Thus, during the calculation of A_f and $\hat{A}_f$ in the grouping algorithm, each $c_f(i,j)$ is immediately retrieved. Calculation of the matrix C_f can be done efficiently due to the progressive nature of the cells. That is, having $C_f[i,i] = c_f(i,i) = \hat{s}$, for $1 \leq i < j \leq n_f$, we have

$$C_f[i,j] = C_f[i,j-1] + (b_{j-1} - b_{i-1}) \times \int_{b_{j-1}}^{b_j} Pr(\mathcal{Z} = z)dz. \tag{7}$$

Hence, each $C[i,j]$ is calculated in $O(1)$ and calculation of the entire matrix takes $O(n_f^2)$, which can be neglected compared to other calculations of the overhead reduction algorithm. The memory required by this matrix is also of $O(n_f^2)$, which is similarly negligible.

The last step is to compute $d(f)$, the duration of a frame f, for being downloaded. Denoting the frame rate of the video by α, $d(f)$ simply equals $1/\alpha$ for a constant bitrate (CBR) video. For a variable bitrate (VBR) video, however, frames do not have the same size and thus are not expected to be downloaded in the same duration. Therefore, we calculate $d(f)$ values as follows. Our algorithm works on a basis of a few (w) GoPs, and no information about frames of further GoPs is known—accordingly, for live streaming scenarios w has to be small, preferably $w = 1$, in order to impose a very short delay. Let the size of the given w GoPs be S bits and their playback duration be D seconds, making the bitrate of the corresponding sequence of GoPs be S/D bps. In order for a client to keep the download rate of the video constantly equal to its bandwidth, the download rate of each frame f should also be S/D bps, since the download rate does frequently vary from frame to frame. Hence

$$\frac{\sum_{i=1}^{n_f} size(u_i)}{d(f)} = \frac{S}{D} \Rightarrow d(f) = \frac{D}{S} \sum_{i=1}^{n_f} size(u_i). \tag{8}$$

Clearly, this is done in $O(1)$ for each truncation unit.

B. Impact of Packet Losses on the Overhead Reduction Algorithm

The proposed algorithm for minimizing the communication overhead may be vulnerable to packet losses. When gathering a number of truncation units in a group and designating one hash value for the group, loss of any of the units will result in unverifiability of the rest of the units in the group. Nevertheless, the proposed algorithm can gain over 50% overhead reduction when streaming over reliable channels or channels with low loss ratios, which is the typical case in today's Internet. For example, in a large-scale measurement study [25], over 85% of traces experienced a loss ratio of below 1%, and 99% of traces experienced a loss of less than 10%. If the loss ratio is significant $(> 10\%)$, the algorithm will have a tendency to form smaller groups. Consequently, the overhead saving by the algorithm reduces when loss ratio increases. Clearly, the algorithm never results in a higher overhead than authentication with no grouping.

Recall that with simple authentication with no grouping, we send the authentication information of quality layers in two copies for lossy transmission scenarios. For grouping the video packets to reduce the overhead, we send one copy of the hashes of units as it is, and apply the optimal grouping on the replication copies. In this case, (5) is updated as follows to capture the *expected* cost of losing a unit, which is the loss of other truncation units, shown in (9) at the bottom of the page, where ρ is the expected loss ratio, and thus the probability that at least a unit from the group is lost is $\left(1 - (1-\rho)^{j-i+1}\right)$, and the probability that the hash values of the units of the group (carried in the same authentication NAL unit) gets lost is ρ.

VI. Performance Evaluation

We use trace-based simulations to evaluate the performance of our scheme in terms of computation cost, delay, buffer requirements for receivers, loss tolerance, and communication overhead. As described in Section II, we are not aware of other schemes in the literature designed for end-to-end authentication of scalable video streams that support the flexible, three-dimensional scalability. Previous authentication schemes are not applicable to such streams, since they cannot authenticate all their possible substreams. Hence, quantitatively comparing our scheme against them is not possible.

A. Simulation Setup

We simulate the transmission of H.264/SVC scalable video streams over a channel with packet losses. The packet size is 1 KB. We consider three diverse videos from the Joint Video Team (JVT) test sequence set, namely "Crew", "Soccer", and "Harbour", and encode them using the H.264/SVC reference software, called JSVM. Each encoded stream consists of four temporal layers (GoP size 8) and two spatial layers providing CIF and 4CIF resolutions. In each of the streams, both spatial representations provide a high quality of approximately 40 dB

$$c_f(i,j) = \hat{s} + \sum_{x=i}^{j-1} \left((b_x - b_{i-1}) \times \int_{b_x}^{b_{x+1}} Pr(\mathcal{Z} = z)\, dz \right) + \rho \left(1 - (1-\rho)^{j-i+1}\right) (b_j - b_{i-1}) \int_{b_j}^{+\infty} Pr(\mathcal{Z} = z)dz \tag{9}$$

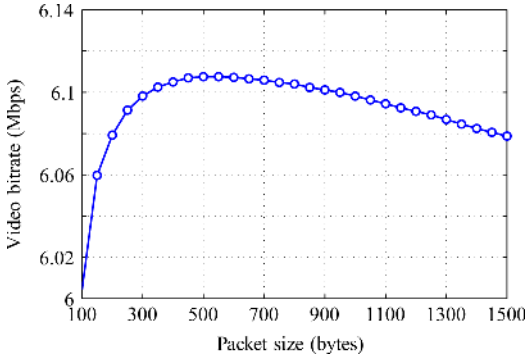

Fig. 9. Average bitrate extracted from the video versus the size of truncation units.

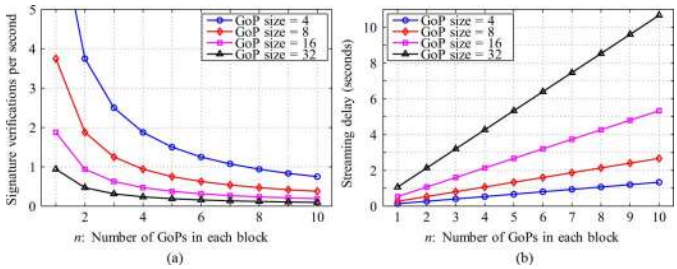

Fig. 10. Computation cost and delay of the proposed authentication scheme. (a) Rate of signature verifications. (b) Streaming delay.

in terms of Y-PSNR. The considered streams are quite diverse in their content, resulting in different bitrates of 1.5 Mbps, 2.1 Mbps, and 3.7 Mbps for the CIF representation of the stream, and 5.5 Mbps, 8.6 Mbps, and 12.3 Mbps for the 4CIF representation. Each spatial layer contains two CGS quality layers. The CGS base layer of each spatial layer provides the minimum quality for that resolution, and has a bitrate between 85 kbps and 140 kbps (at full frame rate) for the considered streams. The CGS enhancement layer of the first and the second spatial layer is divided into four and five MGS layers, respectively. Each MGS layer in turn is divided into multiple truncation units.

To obtain the size of a truncation unit, we performed a local search in the tradeoff between having small truncation units, which means finer granularity but higher NAL header overhead, and having large truncation units, which incurs lower NAL header overhead but provides coarser granularity.

Fig. 9 illustrates this tradeoff as the average bitrate that can be extracted from the video versus the unit size for the "soccer" stream; the other two demonstrate similar results. Accordingly, we chose 500 bytes as the unit size, which means that two such units can fit in a single packet. A high degree of flexibility is provided by the considered streams since a subset of 500-byte packets of any MGS quality layer can be discarded. We determined the 500-byte truncation units of each video frame using our own utilities for parsing SVC streams—this was needed to be done as a postprocessing of the stream created by the JSVM encoder because the JSVM encoder often embeds all video data of an MGS layer in one (possibly big) NAL unit. We then add the authentication information to the layers and transfer the authenticated streams to the receiver through a loss simulator. The loss simulator drops a number of packets according to the desired loss ratio. We employ SHA-1 as the hash function (20-byte hashes), and RSA as the digital signature scheme (128-byte signatures) due to its inexpensive verification, which is an advantage for accommodating limited-capability receivers.

B. Simulation Results

1) Computation Cost: This is the most important performance factor of an authentication scheme. If some receivers cannot afford the computations needed by the scheme, they cannot verify the video at all; we assume the server is powerful enough for providing the authenticated stream in real-time. The dominant operation in the verification process is verifying the digital signatures, as discussed in Section IV-C. Fig. 10(a) depicts the number of signature verifications needed per second for different values of n (the number of GoPs in a signed block). The value of n can balance a tradeoff between delay and computation cost, since the content provider needs to generate a complete block of n GoPs before being able to transmit any of them. Assuming that one to two signature verifications per second are easily affordable by nowadays limited-capability video playback devices [19], Fig. 10(a) shows that gathering only $n = 5$ GoPs in each block suffices for having the authentication operations affordable by all receivers.

2) Delay and Buffering Requirements: When streaming live content, the delay is in proportion to the block size. Fig. 10(b) depicts the delay caused by the authentication scheme for different values of n. For example, with $n = 5$ and a GoP size of 8, the delay is less than 2 s, which is quite acceptable. Moreover, a value of $n = 5$ indicates that in the worst case, where the authentication information of the first four GoPs of a block is lost, the receivers need to buffer five GoPs. Therefore, receivers need a small buffer only: less than 2 MB if receiving the highest-bitrate version of the stream. Note that this represents the buffering required by the authentication scheme; the streaming application may already be buffering a few seconds of video data before playing back, which can be utilized by the authentication scheme and in this case no additional buffering is needed.

3) Robustness Against Loss: Packet losses can negatively impact an authenticated video in two ways. First, some video packets can be lost. Second, some packets, although received, can be unusable as they cannot be verified. Thus, authentication may amplify the effect of losses. We show that our scheme does not suffer from these issues. In our simulator, once a GoP is received, the receiver checks the attached authentication information and may drop a subset of packets from the stream, which cannot be authenticated due to the loss of the corresponding authentication information. The result of this process is shown in Fig. 11, where the fraction of the packets received and verified over the total packets is depicted. As discussed earlier, the authentication information of quality/spatial layers is replicated in a few, k, copies for protection against loss. The first finding by Fig. 11 is that our authentication scheme increases the impact of loss only marginally: as the loss ratio increases from 0 to higher values, the gap between the plain (unauthenticated) video and the authenticated video with $k = 2$ or 3 increases negligibly. As a second finding, Fig. 11 also helps us to determine how many copies the authentication information packet should be sent in:

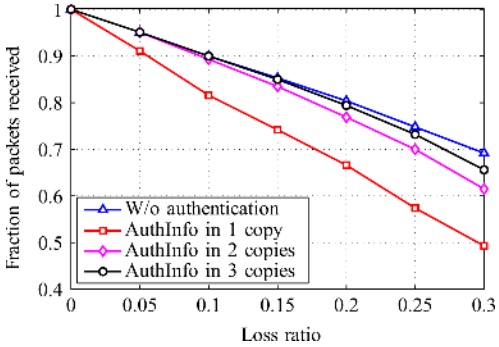

Fig. 11. Loss resilience of the proposed scheme: fraction of received packets that can be verified.

$k = 1$ results in high sensitivity to loss, $k = 2$ is suitable for reasonable loss ratios ($< 10\%$), and $k = 3$ becomes the preferred choice as the loss ratio grows higher. Notice that lower k values are always preferred, since the amount of communication overhead is directly proportional to k.

4) Communication Overhead: We measure the communication overhead as the additional bandwidth a receiver has to consume to receive the authentication information, averaged over all receivers. The overhead for the three video streams, when not employing the overhead reduction algorithm, is shown with the green bars in Fig. 13(a).

To reduce this overhead, we run the proposed algorithm for grouping truncation units of each frame. The number of truncation units in a frame can vary from frame to frame, especially for frames at different temporal layers. Fig. 12(b) shows the average number of truncation units in the frames of each temporal layer. As expected, frames of lower temporal layers have many more packets—as these frames are used for prediction of more other frames, they are encoded with lower quantization parameters to provide stronger prediction signals. There are at least a few dozens of truncation units in each frame, which we optimally group using the proposed overhead reduction algorithm. We assume a multi-modal Gaussian distribution for modeling user bandwidth in a heterogenous environment. In this distribution, shown in Fig. 12(a), three major concentrations of user bandwidths are assumed at 512 kbps, 4 Mbps, and 8 Mbps. By applying the proposed algorithm, the average overhead is considerably reduced for streams as shown in Fig. 13(a) and (b). The saving is higher in the third stream, which has a higher bitrate and a higher number of truncation units per frame. More truncation units per frame provides opportunity for more efficient grouping. We note that packet losses can impact the efficiency of our grouping algorithm, as discussed in Section V-B, though even in the presence of losses, the algorithm still benefits us in reducing the overhead. The gain, however, is reduced from 50% to around 20% and 10% when a loss of ratio 5% and 10% is introduced. For higher loss ratios, although this benefit diminishes to less than 10%, the algorithm never results in an overhead worse than that of not grouping.

As discussed earlier, on one hand, the grouping process reduces the communication overhead by reducing the number of hash values needed. On the other, the grouping itself may cause an overhead, because it omits some truncation possibilities and

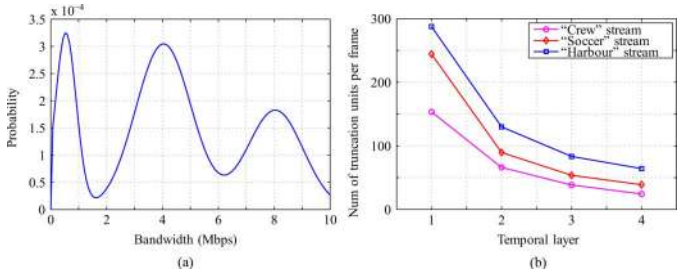

Fig. 12. User bandwidth distribution and the number of truncation units per frame at different temporal levels. (a) Distribution of user bandwidth. (b) Number of truncation units per frame.

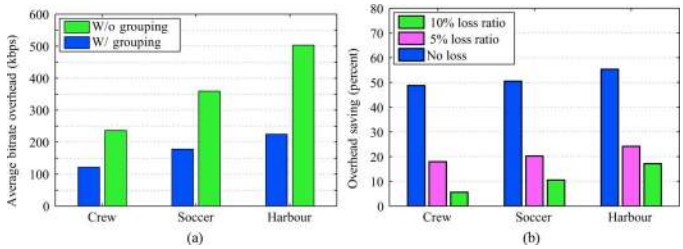

Fig. 13. Efficiency of the overhead reduction algorithm. (a) Communication overhead. (b) Saving in communication overhead.

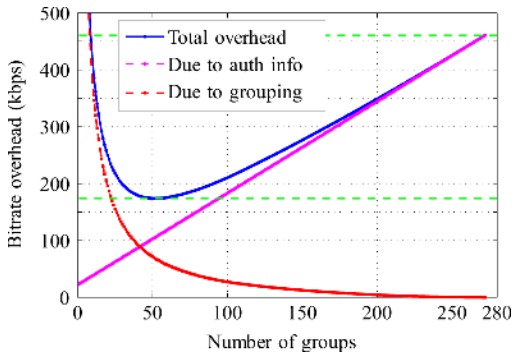

Fig. 14. Deciding the best number of groups.

makes some users receive a lower bitrate video than they would have received if there was no grouping. Fig. 14 depicts this tradeoff for an average frame in the temporal base layer, and shows how the obtained optimal grouping results in a significantly lower overhead than the two ends of the tradeoff.

VII. Conclusion

In this paper, we have studied the problem of authenticating modern scalable video streams. These streams offer a high degree of flexibility for extracting different substreams, without significantly decreasing the coding efficiency. However, they are not supported by current scalable video authentication schemes in the literature. We developed an authentication scheme for these streams that enables verification of all possible substreams, and we analytically proved its security. We have implemented this scheme for H.264/SVC video streams as an open source library called *svcAuth*, which can be employed as a transparent add-on component by any streaming application. We designed an additional algorithm for minimizing the communication overhead, which can become non-negligible

for highly flexible streams. The overhead could be reduced by the proposed algorithm by more than 50% in our experiments. The overhead reduction algorithm can also be used with other authentication schemes in the literature, which are designed for traditional scalable videos, to optimize their overhead. We conducted a simulation study with real video traces, which shows that our authentication scheme is robust against reasonable packet loss rates ($< 20\%$), has low communication overhead, incurs negligible computational cost, adds only a short (1–2 s) delay, and requires no significant buffering (< 2 MB) by receives.

REFERENCES

[1] Streaming Media, IPTV, and Broadband Transport: Telecommunications Carriers and Entertainment Services 2006–2011, T. I. R. Corp., Apr. 2006. [Online]. Available: http://www.insight-corp.com/execsummaries/iptv06execsum.pdf.

[2] Global IPTV Market Analysis (2006–2010), Research Rep., R. I. R. Solutions, 2006. [Online]. Available: http://www.rncos.com/Report/IM063.htm.

[3] M. Wien, H. Schwarz, and T. Oelbaum, "Performance analysis of SVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1194–1203, Sep. 2007.

[4] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[5] S. Wenger, Y. Wang, and T. Schier, "Transport and signaling of SVC in IP networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1164–1173, Sep. 2007.

[6] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile video transmission using scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1204–1217, Sep. 2007.

[7] P. Atrey, W. Yan, and M. Kankanhalli, "A scalable signature scheme for video authentication," *Multimedia Tools Appl.*, vol. 34, pp. 107–135, Jul. 2007.

[8] C. Liang, A. Li, and X. Niu, "Video authentication and tamper detection based on cloud model," in *Proc. Conf. Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP'07)*, Splendor Kaohsiung, Taiwan, Nov. 2007, vol. 1, pp. 225–228.

[9] S. Park and S. Shin, "Combined scheme of encryption and watermarking in H.264/scalable video coding (SVC)," in *New Directions in Intelligent Interactive Multimedia*, ser. Studies in Computational Intelligence. New York: Springer, 2008, vol. 142, pp. 351–361.

[10] R. Iqbal, S. Shirmohammadi, A. El-Saddik, and J. Zhao, "Compressed-domain video processing for adaptation, encryption, and authentication," *IEEE Multimedia*, vol. 15, no. 2, pp. 38–50, Apr. 2008.

[11] D. Skraparlis, "Design of an efficient authentication method for modern image and video," *IEEE Trans. Consum. Electron.*, vol. 49, no. 2, pp. 417–426, May 2003.

[12] H. Yu, "Scalable streaming media authentication," in *Proc. IEEE Int. Conf. Communications (ICC'04)*, Paris, France, Jun. 2004, vol. 4, pp. 1912–1916.

[13] J. Park, E. Chong, and H. Siegel, "Efficient multicast stream authentication using erasure codes," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 2, pp. 258–285, May 2003.

[14] A. Pannetrat and R. Molva, "Efficient multicast packet authentication," in *Proc. Network and Distributed System Security Symp. (NDSS'03)*, San Diego, CA, Feb. 2003.

[15] R. Kaced and J. Moissinac, "Multimedia content authentication for proxy-side adaptation," in *Proc. Int. Conf. Digital Telecommunications (ICDT'06)*, Cote d'Azur, France, Aug. 2006.

[16] C. Gentry, A. Hevia, R. Jain, T. Kawahara, and Z. Ramzan, "End-to-end security in the presence of intelligent data adapting proxies: The case of authenticating transcoded streaming media," *IEEE J. Select. Areas Commun.*, vol. 23, no. 2, pp. 464–473, Feb. 2005.

[17] Y. Wu and R. Deng, "Scalable authentication of MPEG-4 streams," *IEEE Trans. Multimedia*, vol. 8, no. 1, pp. 152–161, Feb. 2006.

[18] J. Apostolopoulos, "Architectural principles for secure streaming & secure adaptation in the developing scalable video coding (SVC) standard," in *Proc. IEEE Int. Conf. Image Processing (ICIP'06)*, Atlanta, GA, Oct. 2006, pp. 729–732.

[19] M. Hefeeda and K. Mokhtarian, "Authentication schemes for multimedia streams: Quantitative analysis and comparison," in *ACM Trans. Multimedia Comput., Commun., Appl.*, 2009. [Online]. Available: http://nsl.cs.sfu.ca/papers/TOMCCAP09.pdf.

[20] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[21] I. Amonou, N. Cammas, S. Kervadec, and S. Pateux, "Optimized rate-distortion extraction with quality layers in the scalable extension of H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1186–1193, Sep. 2007.

[22] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 645–656, Jul. 2003.

[23] K. Mokhtarian and M. Hefeeda, "End-to-end secure delivery of scalable video streams," in *Proc. Int. Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'09)*, Williamsburg, VA, Jun. 2009, pp. 79–84.

[24] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien, "Joint draft 11 of SVC amendment," in *Joint Video Team (JVT), Doc. JVT-X201*, Jul. 2007.

[25] Y. Zhang and N. Duffield, "On the constancy of Internet path properties," in *Proc. ACM SIGCOMM Workshop Internet Measurement*, San Francisco, CA, Nov. 2001.

Kianoosh Mokhtarian (S'09) received the B.Sc. degree in software engineering from Sharif University of Technology, Tehran, Iran, in 2007 and the M.Sc. degree in computing science from Simon Fraser University, Surrey, BC, Canada, in 2009.

He is currently a Software Engineer in the telecommunications industry at Mobidia Inc., Richmond, BC, Canada. His research interests include peer-to-peer systems, multimedia networking, and security.

Mohamed Hefeeda (S'01–M'04–SM'09) received the M.Sc. and B.Sc. degrees from Mansoura University, Egypt, in 1997 and 1994, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 2004.

He is an Assistant Professor in the School of Computing Science, Simon Fraser University, Surrey, BC, Canada, where he leads the Network Systems Lab. His research interests include multimedia networking over wired and wireless networks, peer-to-peer systems, network security, and wireless sensor networks. In addition to publications, he and his students develop actual systems, such as PROMISE, pCache, svcAuth, pCDN, and mobile TV testbed, and contribute the source code to the research community.

Prof. Hefeeda's paper on the hardness of optimally broadcasting multiple video streams with different bit rates won the Best Paper Award in the IEEE Innovations 2008 conference. The mobile TV testbed software developed by his group won the Best Technical Demo Award in the ACM Multimedia 2008 conference. He serves as the Preservation Editor of the *ACM Special Interest Group on Multimedia (SIGMM) Web Magazine*. He has served as the program chair of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2010) and as the vice chair of the Distributed Multimedia track in the International Conference on Embedded and Multimedia Computing (EMC 2010). In addition, he has served on many technical program committees of major conferences in his research areas, including ACM Multimedia, ACM Multimedia Systems, ACM/SPIE Multimedia Computing and Networking (MMCN), IEEE Conference on Network Protocols (ICNP), and IEEE Conference on Communications (ICC). He also has served on the editorial boards of the *Journal of Multimedia* and the *International Journal of Advanced Media and Communication*. He is a member of the ACM Special Interest Groups on Data Communications (SIGCOMM) and Multimedia (SIGMM).