FORMAT-INDEPENDENT AUTHENTICATION OF ARBITRARY SCALABLE BIT-STREAMS USING ONE-WAY ACCUMULATORS

Debargha Mukherjee
Hewlett Packard Laboratories, Palo Alto, CA, USA
Email: debargha@hpl.hp.com

ABSTRACT

We propose a mechanism for authentication of general scalable bit-streams, based on quasi-commutative one-way accumulator functions. Such functions allow flexible partitioning between an auxiliary hash computed for removed parts of an original bit-stream and the hash that a receiver can compute from the received bit-stream, and yet allow generation of a common root hash for the original bit-stream against which the authentication may be conducted. Unlike prior work using Merkle hash trees, the number of auxiliary hashes to be transmitted is always one, independent of the actual version of the bit-stream to be authenticated, and the mechanism is independent of the order of hash accumulation. Further, the method readily lends itself to format-independent authentication and adaptation mechanism by use of appropriate standardized metadata.

Index Terms — scalable bit-stream, authentication, one-way accumulator, hash, commutative

1. INTRODUCTION

To improve multimedia content accessibility and to maximize experience commensurate with diverse and dynamic terminal and network capabilities and conditions, as well as individual preferences, it is essential to adapt multimedia content in the delivery path to end consumers. Scalable bit-streams [1][2] are particularly advantageous in this regard, since they enable adaptation by simple bit-stream removal operations, which additionally can be conducted in the encrypted domain by untrusted adaptation engines by use of progressive encryption techniques [3][4]. Further, for sensitive material, a mechanism to enable receivers to verify authenticity [5][6][7] of received content irrespective of the mid-stream adaptations conducted must be supported. Much of the prior work in this area has been focused on JPEG2000 bit-streams [6][7] partly because of the interest generated by standardization of JPSEC [8] or secure JPEG2000. This paper focuses on such authentication within the context of generalized scalable bit-streams, and proposes a new method, that is considerably more generic and elegant, and further eliminates variability in the amount of hash information transmitted. Recently there has been a new line of research on authentication in lossy packet networks. This paper does not address this scenario specifically, but extensions are straightforward.

Another factor is that the set of rich media content formats to be delivered is growing fast. This justifies a drive towards delivery infrastructure components, such as adaptation, encryption/decryption or authentication engines, or modules thereof that use a universal processing model – which do not need frequent upgrades to support new formats and can even support proprietary ones. This is enabled by associating the content with standardized metadata that is small enough to make delivery alongside the content feasible, yet not detailed enough to leak information about the content from a security stand-point. In the spirit of our prior work on format independent adaptation [9][10], encryption [4] and access control [11], we have paid explicit attention to format-independence in our authentication mechanism as well.

In Section 2 we present a general model for scalable bit-streams and a general framework for authentication of such bit-streams. In Section 3 we describe our proposed authentication mechanism, and in Section 4, we show how the authentication operations may be conducted in a fully format-independent manner based on metadata associated with the content.

2. SCALABLE BIT-STREAM AUTHENTICATION

2.1. Modeling Scalability

A scalable bit-stream often scales along multiple dimensions simultaneously. A universal model called the SSM model [9][10] specifies how segments are removed from such a bit-stream to create lower versions. According to a simplified version of this model, the data in a scalable bit-stream is organized in sequential units called *adaptation units* – representing GOP, Frames, Tiles, etc. – each of which is a hypercube with a variable number of dimensions. If there are L dimensions of scalability in each adaptation unit, and the ith

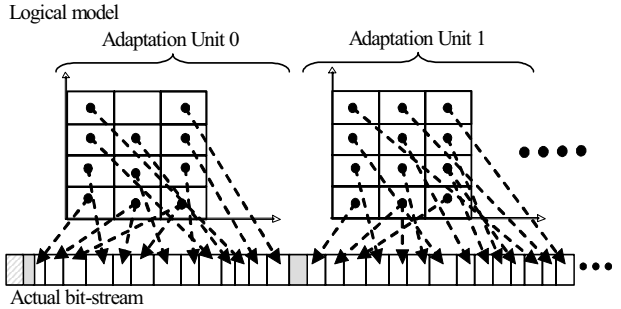

Fig. 1 Bit-stream example showing two adaptation units with 3x4 logical hypercubes in each. Light-shaded and cross-hatched segments correspond respectively to *update* fields and segments that are neither included in logical units nor are update fields.

dimension of the kth adaptation unit contains $l_i[k]$ layers, we say that the data in the kth adaptation unit consists of $l_0[k]\times l_1[k]\times\ldots\times l_{L-1}[k]$ logical data segments $B(i_0, i_1, \ldots, i_{L-1})[k]$, i_j=0,1,…, $l_j[k]$–1, j=0,1,…, L–1, called *logical units* that are arranged in a hypercube. Each logical unit maps to any number of contiguous bit-stream segments located arbitrarily in the actual bit-stream. Fig. 1 illustrates this concept. Note that in addition to bit-stream segments that belong to an atom, there may be segments that do not belong to any.

From this logical hypercube structure [10], we can consider *fully scalable* bit-streams, where each dimension is *incremental* type, meaning layers can be removed only from the outer ends of each dimension to obtain scaled down adapted versions. However, we can also consider other forms of scalability, where one or more atoms are chosen arbitrarily from each dimension. Some other examples of dimension types – *exclusive*, *exhaustive*, *range-type*, etc. [10] – and the resulting scalabilities obtained by combination are shown in Fig. 2 for a single adaptation unit. Note that there may be certain update fields in the bit-stream – such as length of a segment, or number of layers included, or packet index etc. – which need to be updated when segments are dropped in order to create a valid compliant bit-stream. These update fields need special handling because they cannot be used for either progressive encryption as observed in [4], or hash generation for authentication. The *bit-stream support segment* corresponding to a logical unit is thus the concatenation of all contiguous segments belonging to the logical unit in the order they appear in the bit-stream, minus any update fields.

2.2. Authentication strategies

For an arbitrary bit-stream, the conventional authentication strategy is as follows: The owner of the content computes and publishes/transmits a hash using a known hash function for a bit-stream to be distributed. When a receiver receives the content, it can compute the same hash for the received content, and verify authenticity by comparing its computed hash with the published hash. The security of the scheme is derived from the fact that it is impossible for a malicious attacker under reasonable complexity constraints to generate and substitute a fake bit-stream that would yield the same published hash.

Unfortunately, when we have scalable bit-streams that can be adapted mid-stream to handle network and terminal constraints, the above model does not work, because a bit-stream from which layers have been removed, surely would not yield the same hash as the published one. The simplest strategy then is to compute a hash for every possible adapted version of the content and publish/transmit it. The drawback is that for scalable bit-streams with a large number of possible adaptations, it can be messy and expensive to handle and transmit all the possible hashes. A slightly better strategy in cases where the number of possible adaptations is larger than the number of logical units is to have a hash transmitted for every possible logical unit. This is equivalent to the strategy used in [6]. But ultimately, both of these strategies are expensive in terms of bandwidth.

In order to alleviate these problems, Peng et al [7] proposed a scheme based on one-way hash functions and Merkle hash trees employed to generate a root hash in a hierarchical fashion. In this model, transmitted alongside the root hash of the content, is some auxiliary information specific for that version of the content. The auxiliary information essentially represents the contribution of the bitstream segments removed to the root hash. The receiver computes the root hash by combining the hash for the content received with the auxiliary hashes transmitted to obtain the root hash. For the case of a fully scalable bit-stream, the algorithm operates somewhat as shown in Fig. 3 for an exemplary 4×3 logical hypercube bit-stream. Assume $Z(i, j)$ refers to a hash (MD-5, SHA-X etc.) computed from the bit-stream for the bit-stream support segment for logical unit $B(i, j)$. Here h:Z×Z→Z is any one-way hash function. For the specific case as presented in [7], this function is a concatenation of two child hashes followed by a hash computation (MD-5, SHA-X, etc.) on the concatenation. The figure shows how the root hash R is obtained by combining the individual logical unit hashes in a hierarchical fashion. Specifically, first a one way function chain is run for each row backwards, followed by running another function chain vertically backwards to combine the row hashes. If we assume that an adaptation preserves only a 2×2 logical unit, then the logical unit hashes $Z(0,0)$, $Z(0,1)$, $Z(1,0)$, and $Z(1,1)$ would be available from the bit-stream. However, in order to enable verification of the root hash R, the partial hashes R(2,0), R(2,1) and R(0,2) must be transmitted as auxiliary information. Extension to multiple dimensions is obvious.

Some points to note form the above mechanism are as follows: First, the amount of auxiliary information is variable and depends on the actual version transmitted. Second, while the amount of information to be transmitted is usually manageable for the fully scalable case, the situation is not too favorable if the bit-stream

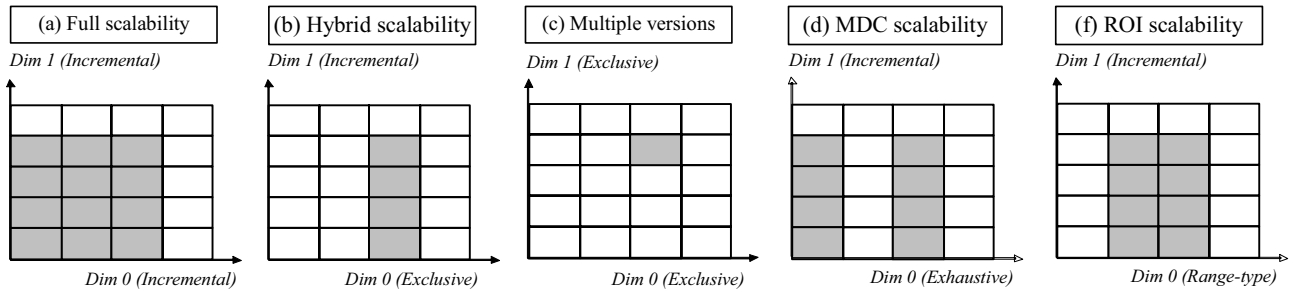

Fig. 2 Various types of dimensions of the logical hypercube model and scalabilities derived from their combinations

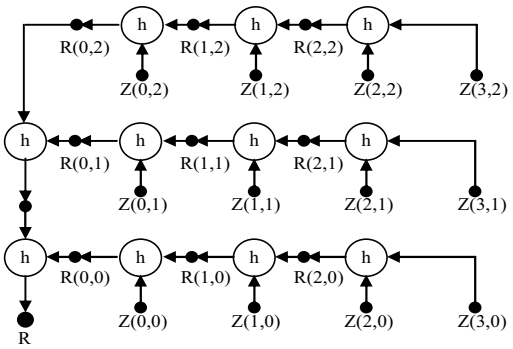

Fig. 3 Root hash generation for fully scalable bit-stream using Merkle hash trees.

supports other scalability types, such as exclusive or range-type. For example, if we consider cropping adaptation of a JPEG2000 bit-stream, where the number of tiles to be removed from all four sides are not known a priori, obtaining a hash tree structure that would result in compact auxiliary information would be quite challenging. Third, the order of computation of the hashes must be conveyed to the receiver in order to enable it to verify authenticity, or a convention for this must be additionally standardized. Further, it must also be conveyed unambiguously to a mid-stream adaptation engine in order to enable it to compute the auxiliary hashes.

In the next section we propose an alternative method where the above problems are alleviated.

3. AUTHENTICATION USING ONE-WAY ACCUMULATORS

Consider a bit-stream with N logical units. Our objective is to obtain a method where any arbitrary partitioning of these N units into M included logical units and N – M excluded logical units, can be authenticated against a root hash, using a single auxiliary hash (see Fig. 4). In order to achieve this objective, we introduce a cryptographic primitive called the *one-way accumulator function* [13]. A one-way accumulator function h:Y×Z→Y is not only one-way: i.e. it is *hard* to obtain y given $h(y, z)$ and z, but it also has satisfies a quasi-commutative property: $h(h(y,z_1),z_2) = h(h(y,z_2),z_1)$. In other words, $h(h(\ldots h(h(y,z_1),z_2),\ldots), z_{n-2}), z_{n-1})$ is independent of the order the z_i's. An example of such a function is the RSA accumulator $A(y, z) = y^z \bmod n$, where n is a very large *rigid* integer, i.e. a product of safe primes. Under these conditions, y cannot be obtained from $A(y, z)$, z and n, in polynomial time.

In our method, we obtain an aggregated hash from the individual logical unit hashes, using such a one-way accumulator function. This aggregated hash becomes the root hash. In particular, the following steps are conducted by the content-owner to obtain a root hash R.

1. Choose n, a large *rigid* integer (product of safe primes).
2. For each logical unit, compute a cryptographic hash by a known method (MD-5, SHA-X, etc.) with the number of bits roughly the same order as n. Denote this hash of logical unit $B(j_0, j_1, \ldots, j_{L-1})[k]$ as $Z(j_0, j_1, \ldots, j_{L-1})[k]$.

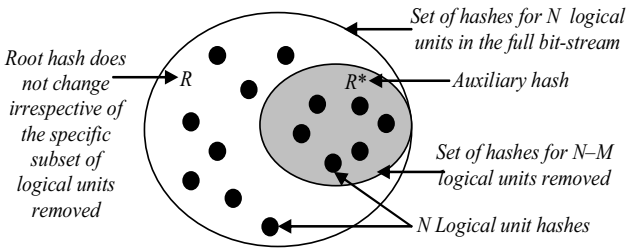

Fig. 4 Flexible partitioning of logical unit hashes

3. Choose an arbitrary large integer R_0. Initialize $R=R_0$.
4. Recursively compute: $R = A(R, Z(j_0, j_1, \ldots, j_{L-1})[k])$, for all atoms $(j_0, j_1, \ldots, j_{L-1})[k]$. That is, for each logical unit $B(j_0, j_1, \ldots, j_{L-1})[k]$, raise R to the hash of the logical unit - $Z(j_0, j_1, \ldots, j_{L-1})[k]$ to obtain the updated R. This is equivalent to computing the aggregated hash R of all the logical units starting from R_0, and its value is the same irrespective of the order of computation of the hashes.

The final result R is the published root hash, along with the integer n.

For each version, which has fewer than the original number of logical units, the auxiliary information R^* is computed as follows:

1. For each logical unit not included in the version, compute the cryptographic hash by the same method (MD-5, SHA-X, etc.) as used for the root hash. Denote this hash of logical unit $B(j_0, j_1, \ldots, j_{L-1})[k]$ as $Z(j_0, j_1, \ldots, j_{L-1})[k]$.
2. Initialize $R^*=R_0$.
3. Recursively compute: $R^* = A(R^*, Z(j_0, j_1, \ldots, j_{L-1})[k]))$, for all atoms $B(j_0, j_1, \ldots, j_{L-1})[k]$ that are not included in the version.

The final value of R^* is the auxiliary information corresponding to the given version. This is communicated to the recipient of a given version of the content along with R and n.

When a receiver receives a version of the content, along with R, n, and R^*, it performs the following steps to verify authenticity:

1. For each logical unit received, compute the cryptographic hash by the same method (MD-5, SHA-X, etc.) as used by the content owner. Denote this hash of logical unit $B(j_0, j_1, \ldots, j_{L-1})[k]$ as $Z(j_0, j_1, \ldots, j_{L-1})[k]$.
2. Initialize $S=R^*$.
3. Recursively compute: $S = A(S, Z(j_0, j_1, \ldots, j_{L-1})[k])$, for all logical units $(j_0, j_1, \ldots, j_{L-1})[k]$ received.
4. Test: If $S=R$, the authentication succeeds, otherwise fails.

Thus, every version of the content is associated with a triple $\{R, n, R^*\}$. For the full version of the content, $R^*=R_0$. But then as more and more of logical units are deleted from the bit-stream in course of adaptations conducted over possibly multiple steps, R^* is updated each time with the hashes of the logical units deleted. The authentication test would still pass because of the quasi-commutative property. Fig. 4 illustrates the concept pictorially.

4. FORMAT-INDEPENDENCE

In this section we show that all functions involved in authentication can be readily supported by format-independent engines. In our previous work [9][10] we have shown that a

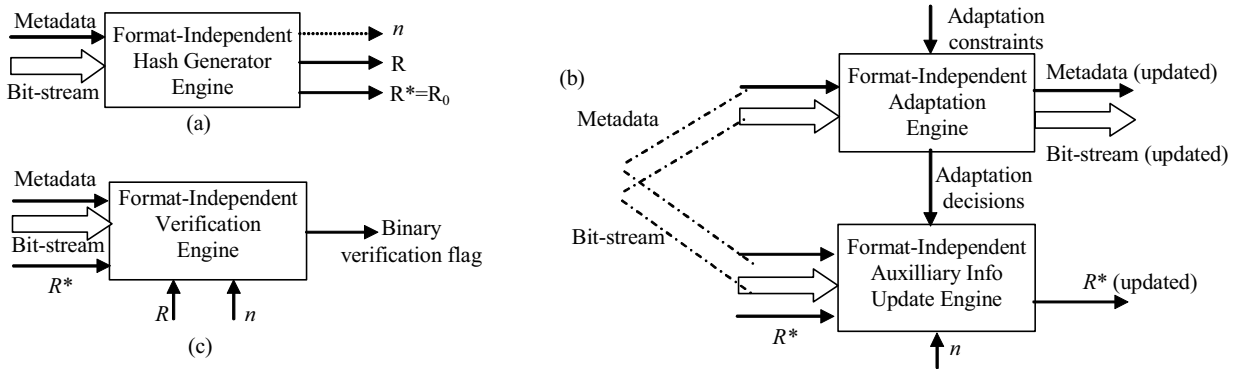

Fig. 5 Format Independent Authentication Components, all driven by metadata

scalable bit-stream can be associated with metadata for adaptation purposes that provides information about the number and types of scalability dimensions, number of layers, and where they lie in the bit-stream. It also provides information on the update fields in the bit-stream, so that they can be removed for hashing purposes. As an example, the (generic) Bit-stream Syntax Description supported in MPEG-21 DIA [12] in conjunction with additional standardization of the bit-stream model, suffices for this purpose. Because the original scalable bit-stream and the associated metadata containing the above information is all that is required to know the number of logical units and their bit-stream support segments (the same mechanism was in fact used in our format-independent encryption work [4]), their hashes can be generated unambiguously and then aggregated to obtain the root hash. In other words, the root hash computation can be performed by a format-independent root hash generator engine as shown in Fig. 5(a). The aggregated hash R, the initial auxiliary hash $R^*=R_0$, and the integer n may be transmitted alongside the bit-stream using additional metadata.

An adaptation engine can next be used to adapt the scalable bi-stream prior to delivery or mid-stream during delivery. It has been shown that this adaptation operation can be conducted in a fully format-independent manner based on the associated metadata and additional metadata representing terminal and network constraints. The adaptation engine includes a decision taking engine that yields decisions regarding which logical units are to be deleted. Based on these decisions and associated metadata, a format-independent auxiliary hash update engine can compute the hashes of the logical units that are removed, and then update the auxiliary hash R^* accordingly. The model is shown in Fig. 5(b). Such adaptations may be conducted in multiple steps.

Finally, at the receiver end, a format-independent verification engine (see Fig. 5(c)) may be used to verify authenticity, based on logical unit hashes computed using the metadata received.

5. REFERENCES

[1] D. S. Taubman, M. W. Marcellin, "JPEG2000: Image Compression Fundamentals, Standards and Practice," Kluwer, Acad. Pubs, 2002.

[2] R. Schaefer, H. Schwartz, D. Marpe, T. Schierl, T. Wiegand, "MCTF and scalability extension of H.264/AVC and its application to video transmission, storage, and surveillance," *Proc. SPIE*, *Visual Communications and Image Processing*, vol. 5960, pp. 243-54, July 2005.

[3] S. J. Wee, J. G. Apostolopoulos, "Secure scalable streaming enabling transcoding without decryption," *Proc. IEEE Int. Conf. Image Processing,* vol. 1, pp. 437–440, Oct. 2001.

[4] D. Mukherjee, H. Wang, A. Said, S. Liu, "Format independent encryption of generalized scalable bit-streams enabling arbitrary secure adaptations," *Proc. IEEE Int. Conf. Ac., Speech and Sig. Proc.*, Philadelphia, March 2005.

[5] B. B. Zhu, M. D. Swanson, S. Li, "Encryption and Authentication for scalable multimedia: Current state of the art and challenges," Proc. SPIE, 2004.

[6] R. Grosbois, P. Gerbelot and T. Ebrahimi, "Authentication and Access Control in the JPEG 2000 Compressed Domain", *Proc. SPIE*, *Applications of Digital Image Processing XXIV*, vol. 4472, pp. 95-104, 2001.

[7] C. Peng, R. H. Deng, Y. Wu, W. Shao, "A flexible and scalable authentication scheme for JPEG2000 image codestreams," *Proc. ACM Int. Conference on Multimedia*, pp. 433-41, Nov. 2003.

[8] F. Dufaux, S. Wee, J. Apostolopoulos, T. Ebrahimi, "JPSEC for secure imaging in JPEG2000," *Proc. SPIE*, *Applications of Digital Image Processing XXVII*, vol. 5558, Aug 2004.

[9] D. Mukherjee and A. Said, "Structured Scalable Meta-formats (SSM) for Digital Item Adaptation," *Proc. SPIE, Internet Imaging IV*, vol. 5018, pp. 148-67, Jan 2003.

[10] D. Mukherjee, A. Said, S. Liu, "A framework for fully format-independent adaptation of scalable bit-streams," *IEEE Trans. Circuits and Systems for Video Technology*, Oct 2005.

[11] D. Mukherjee, M. van der Schaar, "Compact dependent key generation methods for encryption based subscription differentiation for scalable bit-streams," *Proc. IEEE Int. Conf. Image Processing,* Genova, Italy, Oct 2005.

[12] "ISO/IEC 21000-7 FDIS Part 7: Digital Item Adaptation," ISO/IEC JTC 1/SC 29/WG 11/N6168, Dec 2003, Hawaii, USA.

[13] J. Benaloh, M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," *Advances in Cryptology, Proc. EuroCrypt* '93. Lofthus, Norway. May 1993, ed. by T. Heleseth. *Lecture Notes in Computer Science*, ed. by G. Goos and J. Hartmanis. vol. 765, pp. 274-285. Springer-Verlag. New York. 1994.