Secure Transcoding of Internet Content

Yuan-Chi Chang, Richard Han, Chung-Sheng Li, and John R. Smith
IBM Thomas J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532 USA

ABSTRACT

In this paper, we introduce a secure transcoding framework that enables network intermediaries such as proxies to transcode multimedia data without violating end-to-end security guarantees. In our approach, an encoder decomposes a data stream at the source into multiple streams, encrypts each stream independently, and annotates each stream with clear-text metadata. An intermediary performs transcoding by prioritizing the data streams based only on the clear-text metadata, and then dropping lower priority streams. The destination can then decrypt the remaining received streams and recombine them into the transcoded output stream. Our solution offers true end-to-end security since there is no decryption and re-encryption of the data stream midway. As a result, the proxy/intermediary may employ compression-based transcoding of encrypted multimedia data to improve speed of delivery over slow access links without having to decrypt the data.

1 INTRODUCTION

Transcoding of media and Web content has received much attention recently because of the increasing popularity of non-PC devices. We observe that the lack of end-to-end security support in the conventional transcoding solutions can potentially impede its role in e-commerce. In this paper, we outline the design of a secure transcoding framework which allows transcoding and security to co-exist.

Transcoding often refers to the process of transforming multimedia text, images, audio and/or video from the original format in which the multimedia was encoded into a possibly different format and/or quality. There are several objectives to applying transcoding to multimedia content. The first objective is to reduce the download delay of media content over low-bandwidth access links such as modem links and wireless access links [Liljeberg95, Smith98a]. The second objective is to resolve the mismatches between the decoding format supported by a client device and the encoding format employed by a provider of multimedia content. An example of the latter objective is the adaptation of content to computationally constrained or limited-display client devices such as cellphones and PDAs. These objectives have motivated much research and product development in the field of transcoding lately.

The transcoding function typically resides within an intermediary or proxy that is placed between the content provider's Web server and the client device's Web browser. It was observed, however, that the placement of the transcoding function in an intermediate proxy introduces a security problem [Haskell98].

In Fig. 1, a transcoding proxy is introduced as an intermediary between the content provider and the client device. The standard approach to transcoding at a proxy requires that the proxy first decrypt the encrypted data (encrypted by the content provider) before transcoding can be applied. In Fig. 1, the transcoding proxy first decrypts the data, then decompresses the data, then applies a compression algorithm to re-compress the data thereby changing the size of the data and/or its format, and finally re-encrypts the transcoded data for transmission to the client device. The client side decrypt the data again and decompress the data using the new compression algorithm. Once the data has been decrypted in the transcoding proxy and before it is encrypted again, an observer can eavesdrop on the unencrypted data. For example, Fig. 1 shows how the unscrambled image can be viewed at the transcoding proxy. This unscrambled condition may violate the end-to-end security guarantee of privacy implicit in the use of encryption, in which only the sender and receiver are supposed to be able to access the data in its unscrambled state. Though it is possible that in certain cases transcoding proxies may be entities trusted by the sender and receiver to decrypt the data, in general not all transcoding proxies will be trusted.

Our solution to this serious security problem introduced by a transcoding proxy is based on the premise that a content provider first subdivides multimedia content into multiple components. Each of these components may then be independently encrypted. A transcoding proxy downstream of the content provider selectively "filters" or "drops" some of the encrypted components. No media processing functions are performed on those components. Selective filtering achieves compression-based transcoding of the content, improves the speed of content delivery over slow access links, and minimizes the latency incurred in the transcoding process, all without having to decrypt any of the components of the content.

This paper outlines a secure transcoding framework that specifically addresses the aforementioned issue. We discuss the architecture, multimedia decomposition, and the deployment of secure transcoding on SSL.

2 ARCHITECTURE

The architecture introduced in this section enables transcoding (i.e. compression of data) on encrypted data without requiring decryption of the data. An encoder at the content provider/source decomposes the data into multiple components, which are then independently compressed, encrypted, and annotated with clear-text metadata. A secure transcoding proxy inspects the clear-text metadata of each component in order to determine which of the lowest priority encrypted components to drop. The decoder at the client will reconstruct the transcoded data from the remaining still-encrypted components. As shown in Fig. 2, the content provider, e.g. Web/video server, begins by generating multiple components from an existing multimedia object. Next, each individual component's data is passed through a compression algorithm, "C" in Fig. 2. The content provider also annotates each component with *metadata*. This metadata contains labels that identify components and/or describe the importance of components.

The content provider generates two versions of a metadata header, a version upon which encryption will be performed as well as a second version that will stay non-encrypted, i.e. remain in clear text. The two versions of the metadata header are later used by the client device to detect tampering. In Fig. 2, "M" denotes the generation of the metadata header subject to encryption, while "H" denotes the generation of the clear-text metadata header. The metadata header to be

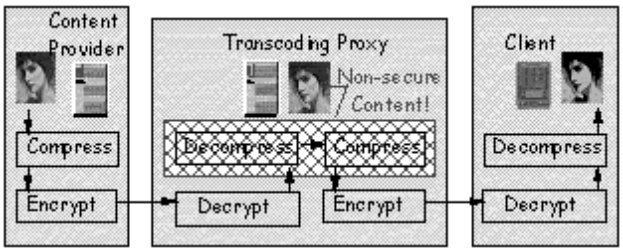

Figure 1. Traditional transcoding at an intermediary decrypts data before transcoding and re-encrypting data, thereby violating end-to-end security.

encrypted for component 1 is labeled "Metadata 1", the clear-text version of the metadata header for component 1 is labeled "Metadata 1B".

In Fig. 2, a simple message is assembled as follows. First, the metadata header 1 is appended to the compressed data of component 1 and this collection is encrypted by the operation "E". Second, the clear-text metadata header 1B is appended to the encrypted collection consisting of the metadata header and compressed component data, as denoted by the operation "A". The output of the second appending operation "A" is an assembled message 1.

At the transcoding proxy, the multiple messages representing the various components of the multimedia object are processed. The transcoding proxy extracts the clear-text metadata header of each assembled message. Using the information provided in the metadata header of each message, the transcoding proxy determines which encrypted components or component portions to selectively drop or substitute.

In Fig. 2, the transcoding proxy receives two components. These components are demultiplexed, and their metadata headers are extracted, as denoted by "A^{-1} ". In this example, the transcoding proxy drops component 2, and forwards the remaining component 1 on towards its destination, namely the client device. Reassembly of the remaining messages is also shown, e.g. metadata headers are joined back with the respective payloads with which they arrived if necessary. In general, there may be K messages, and the proxy may drop $L<=K$ of these messages and may modify the remaining K-L messages, either by dropping or substituting message portions.

The process of selectively dropping or filtering or substituting encrypted annotated components by a transcoding proxy achieves secure transcoding because the size of the multimedia object has been compressed

by an intermediary without having to decrypt any of the data, i.e. components, representing the object.

The decoding process at the client device consists of reconstructing a transcoded version of the original multimedia object from the *K-L* remaining messages forwarded by the transcoding proxy.

In Fig. 2, for each component the client device extracts (A^{-1}) the clear-text metadata header, decrypts (E^{-1}) both the encrypted metadata header and the encrypted component data, extracts (A^{-1}) the decrypted metadata header, compares the two metadata headers for signs of tampering, and finally decompresses (C^{-1}) the component data if no tampering has been found.

3 MULTIMEDIA DATA DECOMPOSITION

There are many ways to decompose data in various modalities: text, image, video, audio or a combination of the above. In this section, the decomposition of image modality is described as an example.

The quality of an image may be controlled through its spatial size, color resolution, and lossy compression. Image decomposition may be achieved through a combination of the following techniques.

1. B/W vs. color – the luminance component of a color image can be extracted and forwarded separately as a black and white image.
2. Spatial size – changing the size of an image can be achieved through wavelet transforms or other downsampling techniques.
3. Lossy compression – it is observed that visual quality of images more strongly correlates with low frequency components than high frequency components.

The above illustrated techniques can be combined to enrich the granules of decomposition. It is noted that the granules may have different sizes after compression and thus one can not determine bandwidth usage by simply counting the number of granules.

4 TRANSCODING SERVICE ON SECURITY PROTOCOLS

Decomposed components are carried by secure networking transport from the sender to the receivers. Secure Socket Layer (SSL) developed by Netscape is

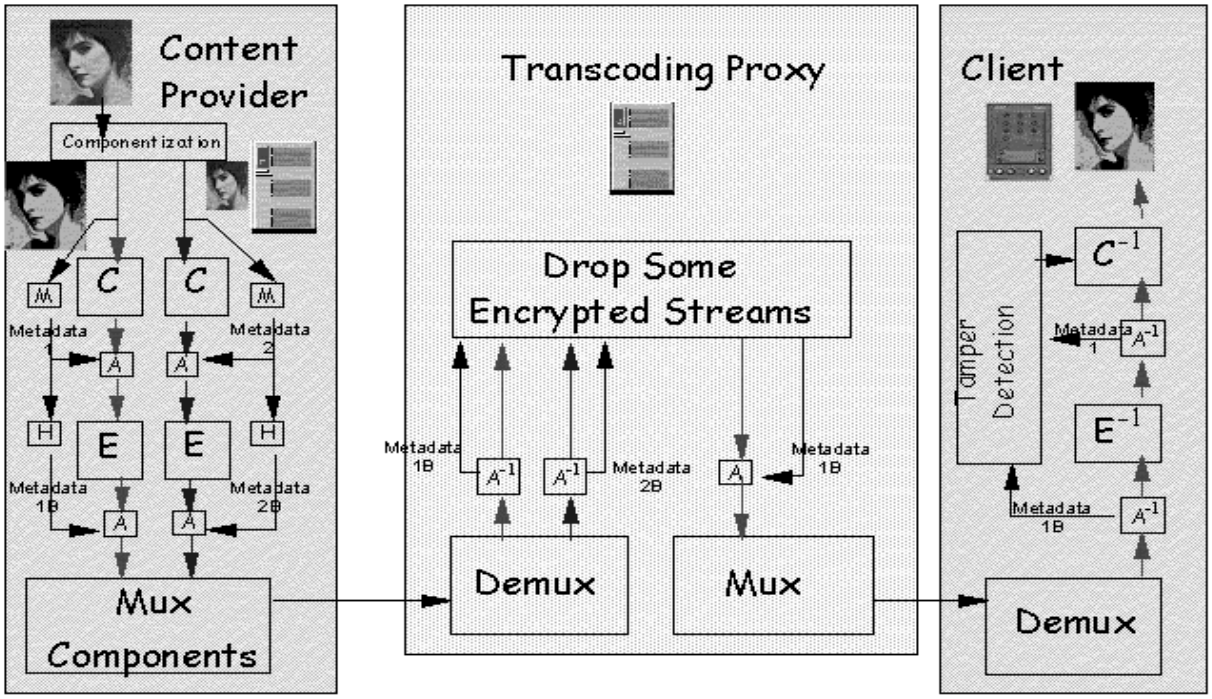

Figure 2. Secure transcoding architecture: (1) the encoder decomposes data into multiple components, encrypts each component independently and adds clear-text metadata to each component; (2) a transcoding proxy drops some encrypted components based only on clear-text metadata; (3) the decoder receives the remaining components, decrypts them, then reassembles the transcoded data.

the most widely used secure transport protocol on the Internet [Freier96]. It provides a secure communication channel between two networking applications that is safe from eavesdropping, tampering, or message forgery.

The transcoding implementation on SSL is illustrated in Fig. 3. Session managers at the server and client sides manages multiple SSL connections, each of which carries one data component. At the beginning of the application session, the client manager requests the server manager for the number of components available. The client manager then sequentially establishes SSL connections to receive all the components. In this figure, four components are available initially. The transcoding proxy in the middle decides that based on client device and connection profiles, only two SSL connections can be supported. Assume that the dependency among data components is simply ordered and self-contained. In other words, the 2nd connection depends on the 1st; the 3rd depends on the 1st and 2nd, and so on. The transcoding proxy then drops IP packets in the 3rd and 4th connections and relays those from the first two. This would cause the TCP sessions of the 3rd and 4th SSL connections to time out and achieve the goals of filtering out unwanted components.

5 CONCLUSION

In this paper we presented a framework for secure transcoding for multimedia content on the Internet. We observed that conventional transcoding solutions, while necessary for matching server data to client preferred formats, prevent fully end-to-end encryption. We outlined the architecture, data decomposition, and security protocol necessary to support our new transcoding proposal. While much is involved in changing the existing data representations to be decomposable, we feel this is necessary and probably the only scalable solution capable of providing end-to-end security without losing the benefit of transcoding.

6 REFERENCES

[Freier96] A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL Protocol Version 3.0," IETF Internet Draft, Nov. 1996.

[Han99] R. Han, "Factoring a Mobile Client's Effective Processing Speed Into the Image Transcoding Decision," *ACM International Workshop On Wireless Mobile Multimedia (WOWMOM)*, 1999, pp. 91-98.

[Han2000] R. Han, J. Smith, "Transcoding of the Internet's Multimedia Content For Universal Access," *Multimedia Communications: Directions and Innovations*, Academic Press, 2000, Chapter 15.

[Haskell98] P. Haskell, D. Messerschmitt, L. Yun, "Architectural Principles for Multimedia Networks," *Wireless Communications: Signal Processing Perspectives*, Prentice Hall, 1998, pp. 229-281.

[Liljeberg95] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, K. Raatikainen, "Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach," Second International Workshop on Services in Distributed and Networked Environments, 1995, pp. 132-139.

[Smith98a] J. Smith; R. Mohan; C. Li, "Content-based Transcoding of Images in the Internet," *Proceedings of the International Conference on Image Processing (ICIP)*, vol. 3, 1998, pp. 7-11.

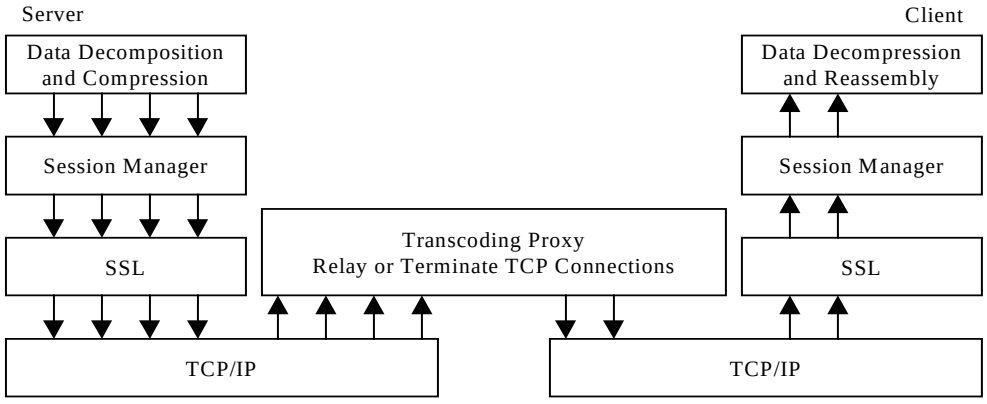

Figure 3. Transcoding proxy on SSL relays or terminates TCP connections.