

Enhancing Progressive Encryption for Scalable Video Streams

Viktor Gergely and Gábor Fehér

High Speed Networks Laboratory,
Department of Telecommunication and Media Informatics,
Budapest University of Technology and Economics
H-1117, Magyar Tudósok körútja 2., Budapest, Hungary
{gergely,feher}@tmit.bme.hu

Abstract. The technique called progressive encryption is used in many areas of content security. However, the plain algorithm itself is only applicable in real transmission scenarios where no packet loss occurs, otherwise additional error correction techniques need to be used in order to achieve maximum decodeability of network packets. The cipher-stepping method (CSM) described in this article adds error correction to progressive encryption in the case where stream ciphers are used to encrypt stream data. It is also explained how the CSM method along with progressive encryption can be used in the encryption of scalable video streams.

Keywords: progressive encryption, scalable video streaming, stream ciphers, wireless networks.

1 Introduction

Scalable video coding has been an active research area for decades. Up to now, several solutions exist for streaming multimedia over networks, especially the Internet. In order to deliver multimedia services over wired and wireless networks, efficient encryption becomes one of the main problems in scalable video streaming. The aim of the Secure Scalable Streaming technology (SSS) [4] is to enable efficient transcoding of encrypted scalable video streams also over wireless networks [3] using progressive encryption. Due to the fact that progressive encryption partially contradicts the requirement of flexible accessibility to the content of the media stream when the media is transmitted over channels that suffer from packet losses, it is desirable to converge better to this requirement by increasing the robustness of progressive encryption.

Section 2 highlights the advantages of stream ciphers that can be exploited to keep the stream cipher synchronized for proper decryption after lost packets, followed by Section 3 introducing the main idea of progressive encryption. Section 4 describes a proposal of an algorithm called cipher-stepping method (CSM) that can be used to keep the state of a stream cipher synchronized with the received packet, hereby enhancing the robustness of progressive encryption.

Robustness of packet decryption is analyzed theoretically and practically based on packet loss measurements on a wireless network, including a comparison of the CSM method to the classical progressive encryption, described in Section 5. The applicability of the CSM method for scalable video streams is presented in Section 6. Section 7 summarizes the advantage and applicability of the proposed algorithm.

2 The Stream Cipher Architecture

Before the demonstration of the CSM method, it is important to understand the main features of the stream cipher architecture. Block ciphers are the most famous types of ciphers used in symmetric encryption. They operate on fixed-length data blocks, i.e. a fixed amount of plaintext can be encrypted with a key, which produces a fixed amount of ciphertext. A typical example of block ciphers is the AES cipher [2] used in a wide range of applications.

On the contrary, stream ciphers operate on streams of arbitrary length, using a secret key and a public initialization vector as input, and output a stream of random-looking symbols, known as keystream [1]. This output is used for encrypting the data stream by applying exclusive-or (XOR) on the data symbols and the keystream, as shown in Fig. 1.

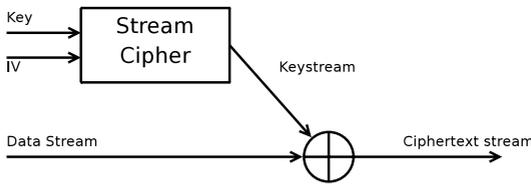


Fig. 1. Encryption using synchronous stream cipher

The stream cipher architecture implies that the state of the cipher is completely independent from the input stream, which can be utilized by the cipher-stepping method described in Section 4. It is important to note that there exist asymmetric stream cipher constructions where this independence cannot be assumed, the article only covers the usage of symmetric stream ciphers.

3 Progressive Encryption

Progressive encryption has been a widely used technique for a long time. The purpose of the technique is to increase data security by encrypting data segments (tiles) continuously without reinitializing the cipher instead of encrypting tiles independently of each other and reinitializing the state of the cipher before encrypting each tile. Examples of usage cover progressive encryption and access

control of JPEG 2000 images [5] or the Secure Scalable Streaming technique [3], [4] used for encryption of scalable video streams.

On the one hand, an advantage of the progressive encryption method is that ciphering a larger amount of data in one session instead of beginning a new session for each tile and ciphering them independently, encryption can be made more secure as it becomes a harder task for the attacker to get access to encrypted information. This is due to lower correlation between the transmitted packets, and solving information related to encryption requires more capacity. On the other hand, progressive encryption enables less flexible accessibility to the data stream, because data packets cannot be decrypted independently of each other.

By using progressive encryption, the time required for encryption of a certain amount of data can be reduced, due to lower number of cipher initializations, which can be rather costly in case of certain stream cipher types. Several new stream cipher types have been introduced [1] as a result of the ECRYPT Stream Cipher Project (eSTREAM) which can possibly outperform conventional stream ciphers i.e. AES in some aspects including speed measured in cycles per byte (CPB) especially for long streams. Therefore, using progressive encryption for these ciphers is essential for acceptable level of security and performance.

4 The Cipher-Stepping Method (CSM)

4.1 Keeping the Cipher State Synchronized

The aim of the cipher-stepping method (CSM) is to keep the state of the stream cipher synchronized with the received packet bytes, even if one or more packet is lost during transmission. By using a cipher for progressive encryption based on the stream cipher architecture described in Section 2, it can be seen that the state of the stream cipher does not depend on the value of the ciphered bytes, only the number of bytes ciphered. Therefore it is desirable to keep a record of the number of bytes ciphered in the actual session to synchronize the cipher to the correct state before decrypting each packet. This can be done by extending the header of each transmitted packet with an extra information containing the number of bytes encrypted in the current session till the current packet. Fig. 2 shows a scenario where packets are extended with the synchronization information.

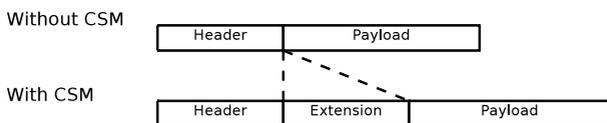


Fig. 2. Header extension for cipher state synchronization

4.2 Decryption Using Cipher Stepping

When decrypting a received packet, first the state of the cipher has to be synchronized with the actual received packet in order to decrypt the packet correctly. By investigating the header extension information in the packet, it can be determined what state the stream cipher should be in for correct decryption. If no packet loss happened after decryption of the previous packet, the number of bytes decrypted with the cipher should be equal to the header information, so the packet can be decrypted using the cipher with no additional operation. If one or more packets are lost before the actual packet, the number of bytes decrypted with the stream cipher is less than the number in the header of the actual packet. In this case, the cipher has to decrypt a number of bytes to reach the state when the received packet can be decrypted. In the CSM method the cipher should decrypt so many bytes of irrelevant information that the number of decrypted bytes is equal to the number of bytes shown in the packet header. As the state of the stream cipher only depends on the amount of information encrypted in the actual session, the bytes used for stepping the cipher to the correct state can be chosen arbitrarily. Provided that no packet disorder occurs during transmission, the cipher can be set to the correct state for each received packet.

4.3 Example Scenario

To summarize the CSM method, the following scenario is taken.

Let us take a session of packets where the first packet is N_1 bytes long, the second is N_2 bytes long, etc. Assume that the second packet is lost over the network. Let C represent the cipher state (the number of decrypted bytes after initialization), and let H_i stand for the header extension information in the i -th packet.

Initially, $C = 0$, because no packets were decrypted so far. After decrypting the first packet, $C = N_1$ holds. The second packet is lost, so after receiving the third packet, the cipher state remains the same as after decrypting the first packet. Before decrypting the third packet, $C = N_1$ and $H_3 = N_1 + N_2$, which implies that $C < H_3$. Therefore, the cipher has to be synchronized before decrypting the third packet using the CSM method, which is done by stepping the cipher with $H_3 - C = N_2$ arbitrary bytes. After synchronization $C = H_3$ holds, so the third packet can be decrypted, and the cipher moves to state $H_3 + N_3 = N_1 + N_2 + N_3$, which is the same as the state of the cipher if the second packet were not lost during transmission.

5 Robustness of Packet Decryption

A self-explanatory measurement of robustness of the cipher-stepping method is comparing the number of decodeable packets while not using any enhancement to the number of decodeable packets using the CSM method. The comparison is made both using theoretical approach and analysing measurements based on

packet transmission over a wireless network. Redundancy solutions that can further increase the robustness of the method itself are over the scope of this article.

5.1 Theoretical Approach

Let us assume that a channel for data transmission has a packet loss probability p for a specific time interval. Let p_i denote the probability of receiving i packets correctly. The number of correctly received packets is represented by probability variable X . Therefore the distribution of X is

$$\Pr(X = i) = p_i \quad (1)$$

where

$$p_0 = p, \quad p_1 = (1-p)p, \quad p_2 = (1-p)^2p, \quad \dots, \quad p_N = (1-p)^N \quad (2)$$

in the case where the cipher-stepping method is not used and N is the number of packets encrypted in a session. The distribution of X is similar to geometrical distribution. The average number of decodeable packets can be derived by calculating the estimated value of this variable:

$$\begin{aligned} \mathbf{E}X &= \sum_{i=0}^N ip_i = \sum_{i=0}^{N-1} i(1-p)^i p + N(1-p)^N = \\ &= \frac{1-p}{p} - \sum_{i=N}^{\infty} i(1-p)^i p + N(1-p)^N = \\ &= \frac{1-p}{p} - N(1-p)^N - \frac{(1-p)^{N+1}}{p} + N(1-p)^N = \\ &= \frac{1-p - (1-p)^{N+1}}{p}. \end{aligned}$$

When using the cipher-stepping method, neither lost packet causes further decodeability problems in other packets, so the distribution of X is reduced to binomial distribution, as shown below:

$$\begin{aligned} \Pr(X = k) &= \Pr(k \text{ correct packets received}) = \\ &= \Pr(N - k \text{ packets dropped out of } N) = \\ &= \binom{N}{N-k} p^{N-k} (1-p)^k = \binom{N}{k} (1-p)^k p^{N-k}. \end{aligned}$$

Therefore X is a binomial probability variable with parameters N , $(1-p)$. The average number of decodeable packets is

$$\mathbf{E}X = N(1-p). \quad (3)$$

5.2 Measurement Results

The robustness of the CSM method was also analyzed using transmission measurements in real time environment. The measurement scenario was sending packets from a wired to a wireless network node through congested channel. The congestion was created by producing CBR background traffic with UDP packets of 2 kilobytes each and a time lag of 2 milliseconds. The analyzed traffic contained UDP packets of length 2 kilobytes with sequence numbering and a time lag of 7 milliseconds. Running the tests with the configuration above resulted in approximately 0.002 probability of packet loss in the analyzed stream over a WLAN channel with 54 Mbit/s bitrate.

The measurement data was derived from identifying lost packets and calculating the number of correctly received packets using progressive encryption with and without the CSM method. For comparison with the theoretical results, the packet loss probability p was estimated by calculating the rate of the number of received packets to the number of all sent packets. Calculating p for shorter periods can lead to adaptively choosing the parameters (header extension overhead and the number of tiles in a session) of the CSM method – assuming that the packet loss probability can be considered constant for the interval of estimation.

The results of the measurement are shown in Fig. 3, where *EX1* and *EX2* mean the estimated values described in Section 5.1, *Sim1* and *Sim2* mean the average number of decodeable packets in the case of using and not using the CSM method, respectively.

The difference between the theoretical and measured values is not significant when using the CSM method, as in this case, a packet loss results in no further undecodeable packets, so the packet loss probability and the number of lost

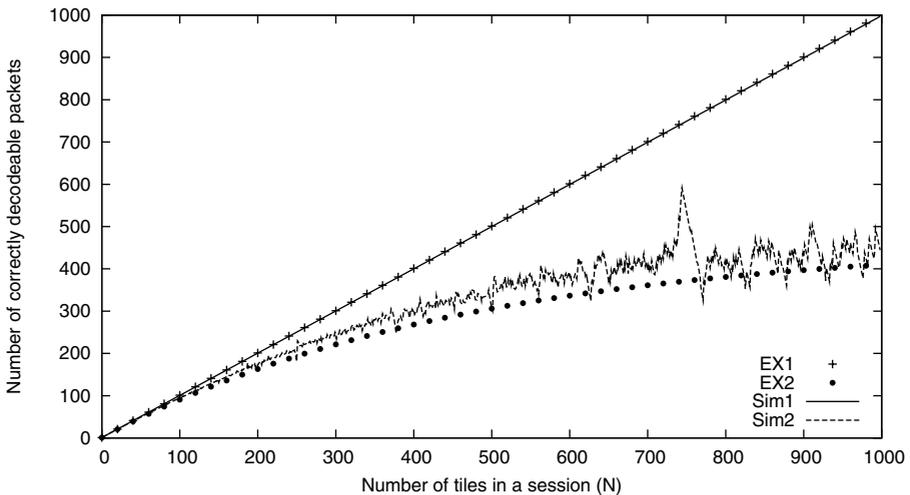


Fig. 3. Measurement results in comparison with theoretical results ($p \approx 0.002$)

packets are strictly correlated. However, when not using the CSM method, the difference of the theoretical and practical results increase if more tiles are encrypted continuously in a session. This effect is due the lower number of samples from which the average number of decodeable packets is calculated when larger sessions are taken. When calculating the number of decodeable packets after a lost packet, all further packets were assumed to be corrupted. In certain application circumstances, when self-synchronizing stream ciphers are used or the state of the stream cipher can be changed adaptively to packet losses, the number of decodeable packets can be increased, even if the CSM method is not used.

Consequently, the measurement results indicate that the CSM method can be very effective in increasing the number of correctly decodeable packets when using progressive encryption with large number of tiles encrypted in a session. In this case large sessions also gain greater security of the transmitted data and produce smaller encryption time by avoiding regular initialization of the stream cipher.

6 Applicability for Scalable Video Streams

For scalable video streams it is an important requirement to have flexible accessibility to the different layers included in the video stream. During encryption, different encryption keys need to be used for the different layers to enable the customization of the video stream in the network nodes without decrypting and re-encrypting the stream. When applying progressive encryption it is desirable to arrange the video packets into different groups by the video layer information and encrypt the packet groups independently in different sessions.

The packets of different layers can differ in packet size, importance, quality or confidentiality, so the length of header extension used in the CSM method can

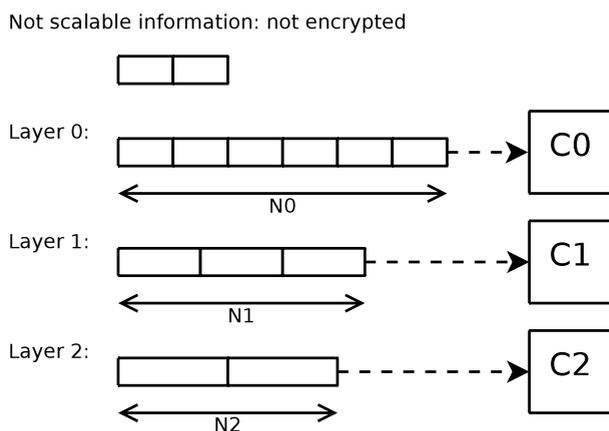


Fig. 4. Example configuration of CSM for scalable video streams

be arbitrarily chosen for each group, making the CSM method and progressive encryption flexible to adapt to the needs of specific applications. Fig. 4 shows an example of the CSM method parameters for each layer group of a scalable video stream.

7 Conclusion

In this paper, a method was introduced which can increase robustness of progressive encryption by utilizing the property of stream ciphers that the cipher state depends only on the amount of encrypted information. The synchronization of the cipher can be maintained by a relatively small overhead introduced by a header extension of the transmitted packets, making decryption resistant to packet losses over a network. Theoretical analysis and wireless network measurements confirm gain increase of the CSM method for large sessions in progressive encryption, which additionally enhances the security of the transmitted content. The proposed method can be easily adapted to encryption of scalable video streams, making the method practically useful.

Acknowledgement

This paper has been (partially) supported by HSNLab, Budapest University of Technology and Economics (<http://www.hsnlab.hu>) and the European Union through the OPTIMIX FP7 ICT project (<http://www.ict-optimix.eu>).

References

1. Bjørstad, T.E.: An Introduction to New Stream Cipher Designs. In: 25th Chaos Communication Congress (December 2008)
2. Schneier, B.: Applied Cryptography, 2nd edn. John Wiley & Sons, Inc., Chichester (1995)
3. Wee, S.J., Apostolopoulos, J.G.: Secure Scalable Video Streaming for Wireless Networks. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, Utah (May 2001)
4. Wee, S.J., Apostolopoulos, J.G.: Secure Scalable Streaming Enabling Transcoding Without Decryption. In: IEEE International Conference on Image Processing (ICIP) 2001, Thessaloniki, Greece, October 7–10 (2001)
5. Haggag, A., Ghoneim, M., Lu, J., Yahagi, T.: Progressive Encryption and Controlled Access Scheme for JPEG 2000 Encoded Images. In: ISPACS 2006. International Symposium on Intelligent Signal Processing and Communications (December 2006)