Regular paper

On a method to authenticate and verify digital streams

Beata J. Wysocki, Yejing Wang, and Reihaneh Safavi-Naini

Abstract — Recently, digital streams have become widely used to make audio, video, and other media available in real-time over the Internet. As with other transmission methods, the recipient needs to have a possibility to verify the source and authenticity of the received information. Several techniques have been proposed to deal with this issue. Most of them are vulnerable to packet losses or they introduce unacceptable computational and/or communication overheads. Some of the graph-based techniques provide immunity to burst losses of certain length. However, these techniques are not immune to the loss of packets containing signatures or occasional burst of lengths greater than the assumed one. In the paper, we propose a modification to one of the graph-based techniques that introduces immunity to the loss of packets containing signatures, without introducing any additional overheads.

Keywords — authentication algorithms, hash chains, signing digital streams, Markov modelling, Gilbert-Elliot channel.

1. Introduction

Streamed data or a stream of data packets is generated by a specialized application on a server machine, and then send in a form of autonomous packets over the Internet (or other packet switched network). The process of splitting a large block of information into the autonomous packets by a server application needs to be distinguished from normal splitting of a large file into packets before sending them through the packet switched network. In the later case, a file to be transmitted is divided into packets by a transport protocol (e.g. TCP [1]). The packets are then transmitted to the receiving host. Because they are numbered, they can be reassembled at the receiver in the desired order, and in case of packets missing or erroneous ones, they can be retransmitted. Only after correctly reassembling the packets (some integrity is checked through the error control coding), the full file is passed by the transport protocol to the application layer of the receiving host. The application can then execute this file (or perform other operation). This is illustrated in Fig. 1 where it is shown that for the application layers in normal data transmission over the Internet, the transmitted file is one contiguous block, not much different from a file read from disk.

In case of streaming, a file is divided into packets at the application layer. These packets are of the size, which would fit into the size of a packet in the underlying packet network. In the case of the Internet, it is usually not more than 1000–1500 bytes [2]. Every packet of a stream is than treated like a full file by the transport layer, and transmitted to the receiving host independently. As soon as such a packet arrives at the transport layer of the receiving host it is passed to the application layer, as there is no need to reassemble anything. For the transport protocol, this packet

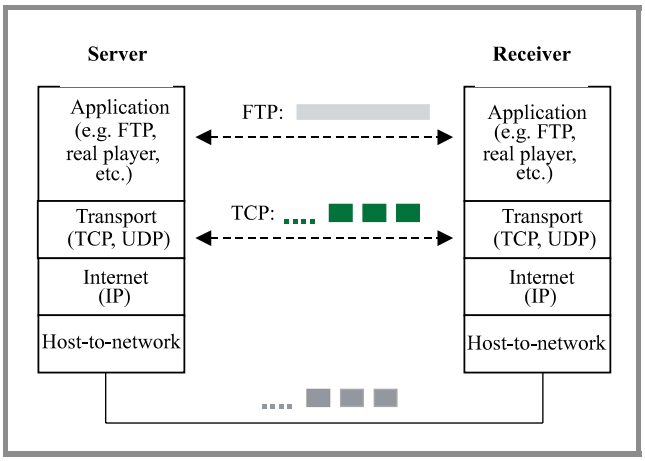

Fig. 1. Transmission of non-streamed data during a file transfer.

constitutes the whole file. The client application can than execute or play the received packet as it comes. It is the application task, to discard packets coming out of order. This is illustrated in Fig. 2.

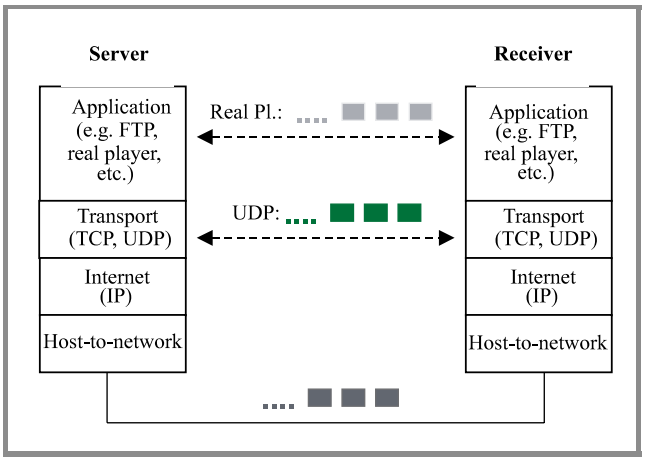

Fig. 2. Transmission of streamed data during a real player session.

As the streaming involves mainly real time applications [3] (e.g. real player) the UDP [1] transport is usually used, which means that there are no retransmissions or integrity checks at the transport layer. This must be catered for by the application. The advantage of UDP over TCP is its speed and lower communication overhead but there is no guarantee for packet delivery or for its correctness.

Another feature of streams, which distinguish them from normal messages, is that the receiver utilizes data and receives at more or less the input rate. Hence, it cannot buffer large amount of unused data [4].

The importance of the need to authenticate streamed data manifests itself in a fact that a recipient would like to have a possibility to verify the source and authenticity of the received information. For example, another feature of streams, which distinguish them from normal messages is important to the listeners of an Internet radio station that the audio stream they receive was really broadcasted by the station they listen to. On the other hand, it is equally important to that station that only the content it broadcasts is attributed to it. Malicious parties should be prevented from injecting commercials or offensive material into the stream.

Moreover, those guarantees must be non-repudiable, which means that a positive verification has to be enough to hold the transmitter responsible for the contents. An answer to those problems is the use of digital signatures. However, the digital signature technology has been developed for single messages, not for a continuous stream of autonomous packets [5–7].

Authentication of a single message or a file transferred from a server to a remote client can be done using one of the standard schemes, like digital signature. The signature or other authentication features are generated by the application and appended in some form to the file. The file is then passed to the transport layer, where it is packetized. The part that contains the signature is not treated in anyway different to the other contents of the message (file). At the remote client machine, the transport layer reassembles the received packets, checks the message (file) integrity and passes the whole message (file) to the application layer, where verification of the message (file) is performed.

For the streamed data, this approach is not possible to use. First of all, the application layer at the server generates autonomous packets itself, and the transport layer considers those packets as separate messages. At the remote client the packets are used as they arrive. This means that they are not assembled back at the application layer. In the optimal solution, they are even not buffered. Hence, it is not possible to have the approach used to authenticate and verify messages (file) used for non-streamed data directly applied to the streamed data. The nature of a digital stream forces the need to authenticate and verify each of the received packets. As a result, each packet must carry some features verifiable by a remote client.

Several different authentication/verification schemes have been proposed in literature, e.g. [5–7, 9, 10, 14]. Most of them can be classified into two major groups:

- The schemes where every packet carries the full information necessary to verify the packet [6, 7], which in some sense resembles signing of every packet individually.
- The schemes where only one packet in the stream (usually the first or last) is signed and the other packets are connected with the signature by a chain of hashes [5, 9, 14].

There are, of course, several modifications to the second group, e.g. [14]. However, all of them suffer from the fact that once the verification chain is broken, there is no means to verify packets incoming after the break. The way to avoid this drawback, and somehow combine the benefits of both groups has been proposed by Golle and Modadugu in [10].

They proposed to divide the stream into sequences of N packets, with each of the sequence being individually verifiable. Hence, a break in the verification chain would result in rejection of a maximum of one sequence of packets but not all packets coming after the break. The technique proposed in [10] is resistant to bursty losses with bursts of up to a predefined length. However, even an isolated single packet loss, when the packet containing a signature is involved, results in a loss of the whole sequence of packets. In this paper, we propose a simple but efficient way to mitigate this problem.

The paper is organized as follows. In Section 2, we briefly introduce some of the techniques proposed to authenticate digital streams, concentrating on the method proposed in [10]. Section 3 describes the Gilbert-Elliott channel model [15, 16] and shows the level of vulnerability of Golle and Modadugu method to the loss of signed packets. In Section 4, we introduce our modification and calculate the probability that the modified authentication chain can be broken for a given bursty channel, while Section 5 concludes the paper.

2. Some of the proposed schemes to authenticate digital streams

2.1. Block signatures

Signing every packet of the data stream can be considered as a direct extension of the message signing technique into authentication of digital streams. This approach, however, involves unacceptable overheads, both a computational and a communication one. Some reduction in those overheads without compromising the benefits of having every packet independently verifiable has been proposed in [6].

One of the stream signing techniques proposed by Wong and Lam in [6] is a star chaining technique. During the authentication phase, a block of "m" packets is formed and hash functions are calculated for every packet in a block, and then for a sequence of these individual hashes a block hash is calculated. After that, the block hash is signed using a standard digital signature algorithm. For packets to be individually verifiable, each packet needs the full authentication information, called a packet signature. The packet signature consists of the block signature, the packet number in the block, and the hashes of all other packets in the block. The authentication process and the transmission of the resulting packet block are illustrated in Fig. 3.

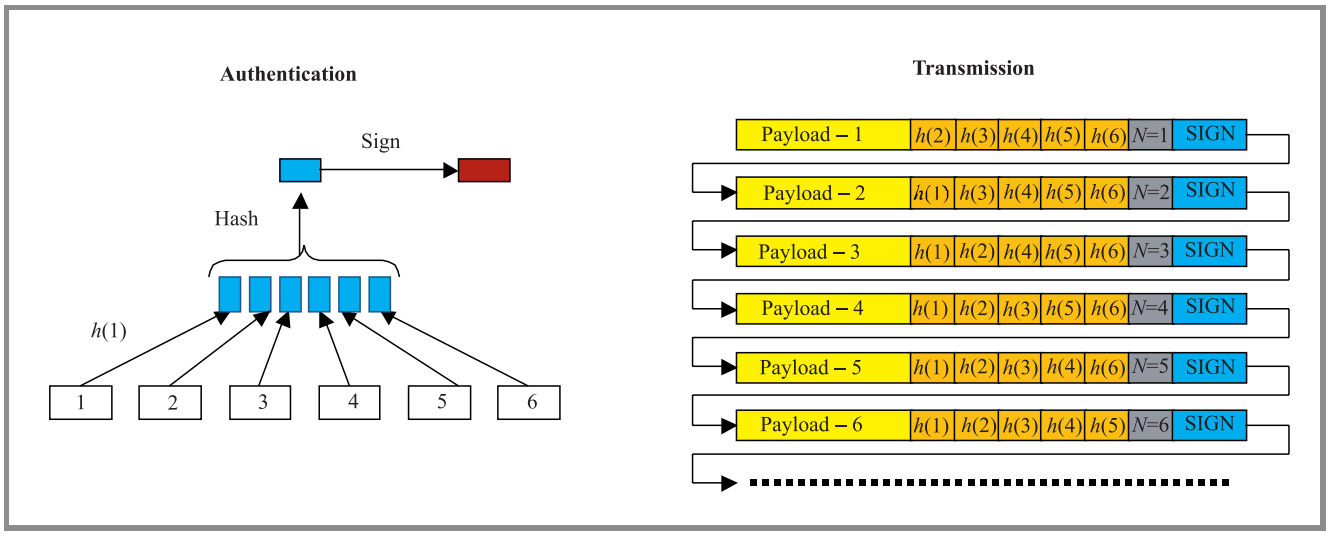

Fig. 3. Authentication process in star chaining scheme [13], and transmission of the resulting packet block.

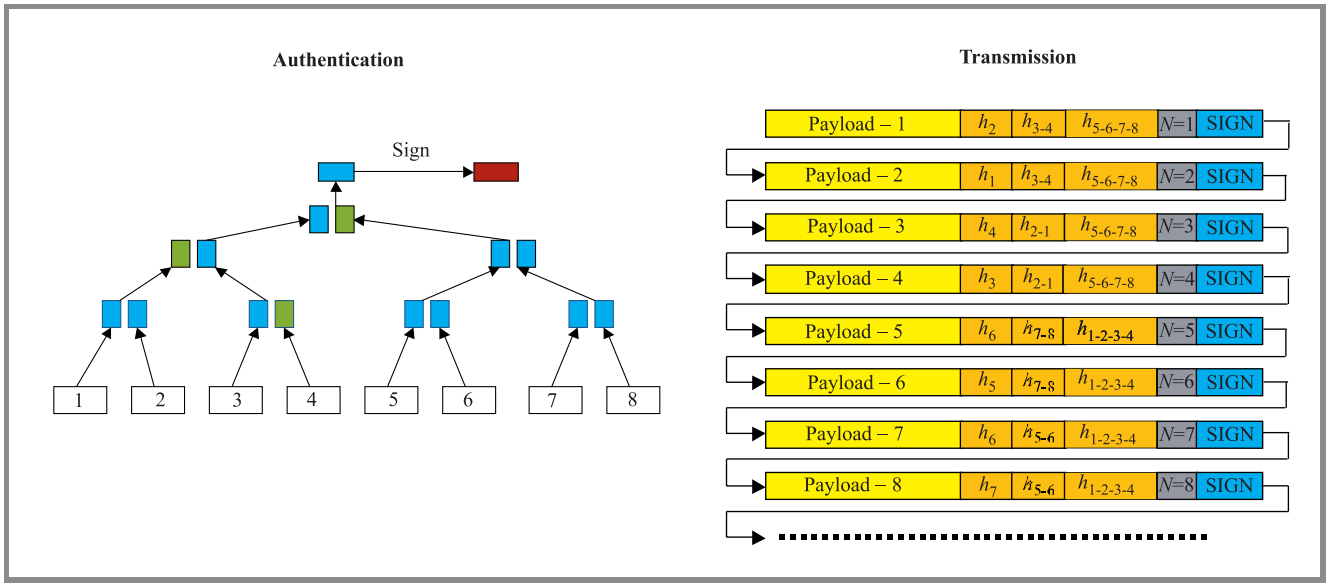

Fig. 4. Tree chain authentication process [13], and transmission of the resulting packet block.

The verification procedure is as follows:

- For the first packet received from the block, the verifier calculates the packet's hash.
- Based on the calculated packet hash, the packet number, and the hashes of other packets in the block contained in the packet signature, the verifier computes the block hash.
- Using the calculated block hash, receiver verifies the block signature, if it is correct, the packet is accepted, if not it is rejected.

For all other packets from the block, the verifier needs only to calculate the new packet hash and compare it to the hash contained in the packet from this block previously positively verified. If they agree, the new packet is verified.

Another scheme proposed by Wong and Lam in [6], is a generalization of their star chaining technique. In this generalized scheme, the block hash is computed as a root node of an authentication tree (see Fig. 4). In such a tree, the packets' hashes are the leaf nodes of the second-degree authentication tree, with other nodes of the tree computed as hashes of their children. For example, in Fig. 4, the parent of leaves D_3 and D_4 is $D_{3-4} = h(D_3, D_4)$. The block signature is calculated on the block hash.

In tree chaining, a packet signature (packet overhead) consists of:

- The block signature.
- The packet number in a block.
- Siblings of each node in the packet's path to the root.

From Fig. 4, it is visible that the communication overhead can be reduced compared to the star chaining scheme. However, this is paid by the increase in computational overhead.

During the verification process the packet's path to the root is verified, i.e. all nodes on that path. The procedure is similar to that for the star-chaining scheme.
The main advantage of the both schemes is the ability to independently verify each of the received packets, so there is no problems with packets lost or tampered with that can be discarded independently. On the other hand, they both involve high computational overhead and quite high communication overhead [10].

2.2. Hash chains

The simplest scheme using a hash chain was proposed by Gennaro and Rohatgi in [5]. It involved the use of just one full digital signature for the whole stream and hashes for each block of "c" packets. The receiver required a buffer of size "c". The receiver first received the signature of the 20-byte hash of the first block and the hash itself. After verifying the signature, the first hash should be verified. He then started receiving the first block and calculated the hash for this block. If it matched the verified hash, it could then play the block. Otherwise the whole stream was rejected. The first block carried the hash for the second block, and so on. At the sender, the whole stream had to be known in advance, as the hash for the block "$i+1$" was appended to the block "i". Reducing the block size "c" to a single packet meant no need for a buffer at the receiver side.
The main advantages of the scheme were very low computational and communicational overheads. However, the scheme was very vulnerable to packet losses, and even a single packet loss would result in a break in the verification chain and rejection of all successive packets. Moreover, the scheme was suited for the off-line applications only. Some modification to the scheme was also proposed in [5] for on-line applications. Unfortunately, it involved significant complication of the scheme while maintaining its vulnerability to packet losses.
Miner and Staddon in [14] propose to introduce additional connections in the authentication chain in order to achieve immunity against lost packets. As the result, the verification chain can tolerate even burst losses of up to the predefined number of packets. However, once the chain is broken due to the longer than assumed burst, the verification cannot be continued. Another disadvantage of the scheme is the fact that it is suited for the off-line authentication only, as the sender needs to know all packets in the stream to calculate the desired sequence of hashes
A breakthrough in design of the hash chain based authentication schemes for streamed data has been the technique proposed by Golle and Modadugu in [10]. It follows from the fact that if a collision-resistant hash of packet P_1 is appended to several modifications to the scheme were proposed in literature, and well generalized by packet P_2 before signing P_2, then the signature on P_2 guarantees the authenticity of both P_2 and P_1. In general, a hash h_1 of P_1 is appended to P_2 before calculating a hash h_2 for P_2, and h_2 is appended to P_3 before calculating h_3 for P_3, and so on. The final packet in a sequence, P_n is then signed after appending h_{n-1} to it. If then the sequence of packets $P_1, \ldots, P_n$ is received without tampering or losses, all packets in that sequence can be authenticated. The process of creating such a simple authenticating graph is presented in Fig. 5.

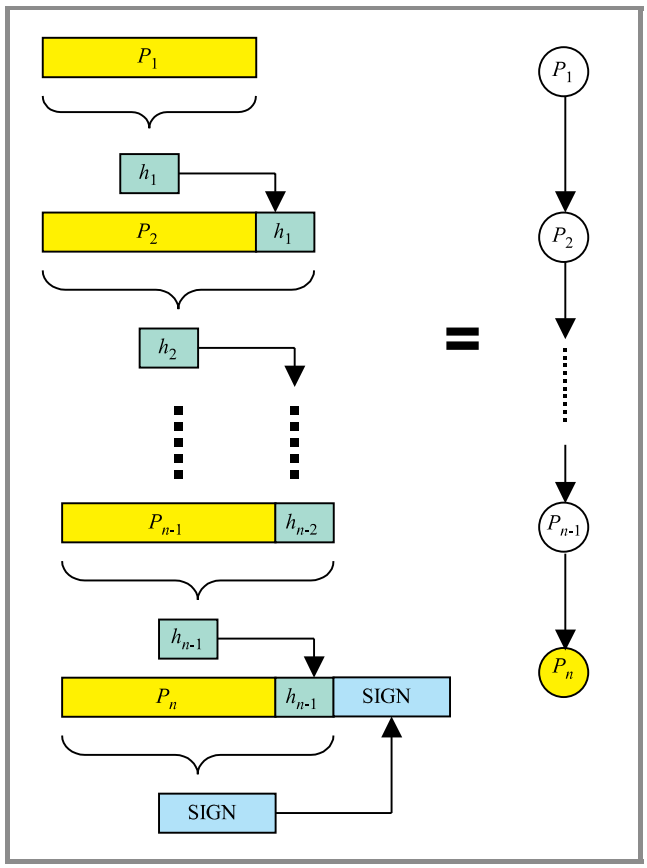

Fig. 5. Basic one-way hash chain of Golle and Modadugu.

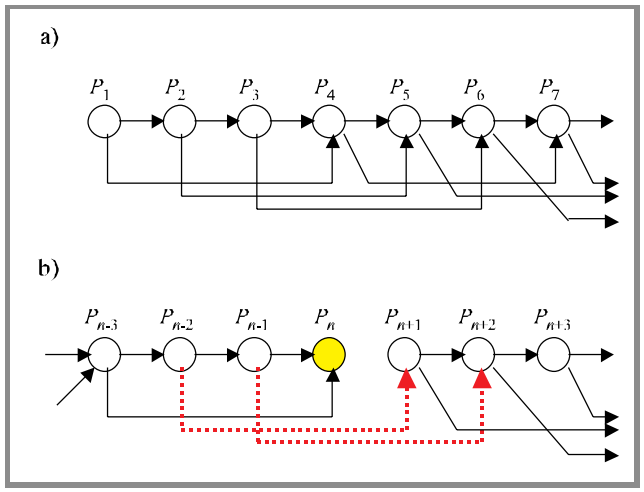

Fig. 6. Explanation of the proposed modification method: (a) the original authentication chain of order 3 [10]; (b) continuation of the chain beyond the signed packet P_n.

The simple chain presented in Fig. 5 can be modified to include supplementary connections to prevent it from being broken in a case of packet losses. Depending on the

type of construction of those additional connections, immunity from burst losses can be achieved. As an example, in Fig. 6a, the chain immune to a loss of two consecutive packets is presented. Several different constructions are presented in [10] and analyzed.

The main advantage of the scheme is the fact that by splitting the stream into smaller sequences of packets and reversing the order of the authentication chain, Golle and Modadugu achieved a scheme suited for on-line applications where a break in the verification chain means rejection of usually just one sequence of packets (maximum two sequences if the break includes a signed packet). Other advantages of the technique are immunity to bursts of up to a given length, low communication overhead, and low computational overhead [10]. The drawbacks of the scheme are the delayed verification and susceptibility to a loss of signed packets.

3. Model of a bursty channel

In his fundamental paper [15], Gilbert introduced a two state Markov chain to model a transmission channel with burst errors. The model has been later refined by Elliott in [16], and is generally known in telecommunications literature as the Gilbert-Elliott channel. The Gilbert-Elliott model has been introduced to analyze channel at bit level. However, we can use the same approach to perform analysis at the packet level.

Gilbert in [15] has shown that a Markov chain with two states can be used to generate bursts. The model is shown in Fig. 7, where the states G and B denote the "good" channel state and the "bad" channel state, respectively. In the "good" state the probability of packet loss approaches zero, while in the "bad" state it can take any arbitrary value greater than zero. In the original model [15], that probability was set to 0.5, as the author dealt with bursts at the bit level, i.e. in the physical channel, where bursts contain good bits interspersed with the errors. In this paper, to simplify considerations, we will assume that at the packet level it is very high, approaching 1. However, more general approach can be taken at the later stage.

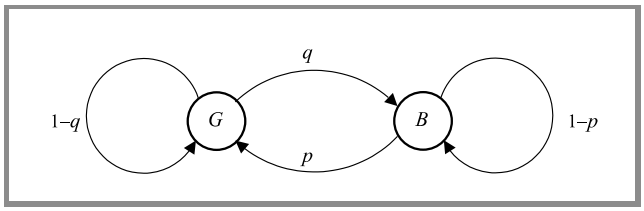

Fig. 7. Gilbert-Elliot channel model.

The model is described by the probability transition matrix $\mathbf{P}_T$ given by:

$$\mathbf{P}_T = \begin{bmatrix} 1-q & q \\ p & 1-p \end{bmatrix} \tag{1}$$

and the corresponding graph is presented in Fig. 7.

For this Markov chain, the stationary probability vector $\mathbf{P}_S = [p_1,\, p_2]$ can be calculated using the formula:

$$\mathbf{P}_S = \mathbf{P}_S \mathbf{P}_T \tag{2}$$

and the normalization condition:

$$p_1 + p_2 = 1\,. \tag{3}$$

From Eqs. (1) and (2), we get a set of two simultaneous equations:

$$\begin{cases} p_1 = p_1(1-q) + p_2 p \\ p_1 + p_2 = 1 \end{cases} \tag{4}$$

that give us:

$$p_1 = \frac{p}{1+p-1+q} = \frac{p}{p+q} \tag{5}$$

and:

$$p_2 = \frac{q}{p+q}\,. \tag{6}$$

The stationary probabilities p_1 and p_2 are the probabilities that at any discrete time instant the channel is in the "good" state or the "bad" state, respectively. For the transmitted packets, it translates on the probabilities that the packet is either received correctly or lost. In addition, the exact values of the probabilities p_{12} and p_{21} fully determine the probabilities of bursts occurrences and their lengths.

Let now denote by $P_g\{M\}$ the probability that out of M signed packets in the stream all packets have been transmitted in "good" state, and that subsequently no such a packet has been lost. Following the above analysis of the Gilbert-Elliott model, we can write that:

$$P_g\{M\} = p_1^M\,. \tag{7}$$

Hence, the probability $P_b\{1\}$ that at least one signed packet is transmitted in the "bad" state and subsequently lost is given by:

$$P_b\{1\} = 1 - P_g\{M\} = 1 - p_1^M\,. \tag{8}$$

To illustrate the problem of the authentication method proposed in [10], let us consider the following example.

Example 1. We consider here the chain construction proposed in [10] that provides immunity against bursts of length $l = 5$ and the stream consists of 50 sequences of 50 packets each. The packets are transmitted through the Gilbert-Elliott channel having the following parameters[1]:

Probability of a transition from G to B: $q = 0.01$

Probability of a transition from B to G: $p = 0.3$

[1]Even though the exact channel parameters q and p are difficult to find for virtual transport channels, i.e. where packet loss is considered, the values used in the example correspond to one of the best channels reported on in [17].

The stationary probabilities p_1 and p_2 can be calculated using Eqs. (5) and (6), and are equal to:

$$p_1 = \frac{0.3}{0.3+0.01} = 0.9677 \text{ and } p_2 = \frac{0.01}{0.3+0.01} = 0.0323$$

and the probability that at least one signature is lost equals to:

$$P_b\{1\} = 1 - 0.9677^{50} = 0.8059\,.$$

The presented example indicates that even for quite a modest stream (50 sequences of 50 packets of 1000 bytes correspond to about 2.5 MB), the probability of loosing at least one signed packet is very high, and with the increased number of sequences it approaches 1.

4. Modification method

Here we propose a simple method to overcome the problem of losing a signed packet without introducing any additional overheads or delays in packet verification, when no signature is lost. An extra delay in verification time will be only introduced in the case of a burst containing the signed packet. However, in the original scheme described in [10], this situation would result in the whole considered sequence of packets being rejected.

The proposed method is based on extending the authentication chain of [10] beyond the packet containing the signature, i.e. the packet, on which the authentication graph should end for the given sequence. In other words, we propose to consider the signed packet in a similar way to any other packet, with the exception that no hash is generated for it. Suppose a stream is divided into sequences $S_1, S_2, \ldots$, where each S_k consists of n packets. Let

$$S_k = \{P_{(k-1)n+1}, P_{(k-1)n+2}, \ldots, P_{kn}\}\,. \qquad (9)$$

For each sequence Golle and Modadugu [10] proposed an optimal authentication chain to provide resistance for bursty loss of packets with the assumption that the signed packet P_{kn} is not lost. We modify their authentication chain by introducing extra edges between S_k and S_{k+1} and removing the assumption of no signed packet lost.

Our authentication chain is defined as follows. Let a be a positive integer. The hash of a packet P_i is appended to two packets P_{i+1} and P_{i+a} for all i, $(k-1)n < i < kn$, and for all $k \geq 1$. This authentication chain sustains bursts of $a-1$, packets, even including the packet with signature. If the signature packet is lost, the authentication has a delay of $2n$.

An example application of the proposed solution together with the authentication chain of [10] is given in Fig. 6. The diagram of Fig. 6a shows the original authentication chain, while the diagram of Fig.6b presents what happens around the signed packet P_n. As the authors of [10] point out, their authentication chain (Fig. 6a) is immune to bursts of 2 packets, but the signed packet P_n must be delivered correctly. In our case, if the burst of 2 packets contains the signed packet, e.g. packets P_{n-1} and P_n are lost, the authentication chain is not broken, and verification can be performed after receiving the next signed packet, i.e. P_{2n}. Moreover, if the bursts are no longer than the length for which the original chain has been designed, any received signature can be used to verify all previously received packets, no matter how many signed packets have been lost.

There are no any additional computational overheads involved, and there is almost no (the hashes from the previous sequence packets are now attached to the current sequence packets, as in Fig. 6b) additional communication overhead involved in the proposed scheme compared to the original scheme described in [10]. All chain constructions proposed in [10] can be used together with the proposed method. This does not seem to be the case of constructions proposed in [14], as the signatures are there transmitted with the first packet in sequences. However, this requires some further investigations.

Let us now analyze the probability that the authentication chain constructed in accordance with our modification will be broken. This can only happen when the burst containing more than b packets occur, while the construction is immune to bursts of no more than b packets.

The burst of length $l = b$ occurs when channels changes state from G to B, stays there for b consecutive steps and returns to G afterwards. Because the verification chain is resistant to burst of length $l = b$, the break of the verification chain occurs at the state $s^{(n)} = B$ if, $s^{(n-1)} = B$, $s^{(n-2)} = B$, $\ldots$, $s^{(n-b)} = B$, and $s^{(n-b-1)} = G$. Hence, we can model the verification chain breaking process by a sequential stochastic machine of $b+2$ states. We denote the states of this machine by $\sigma_1, \sigma_2, \ldots, \sigma_{b+2}$.

The state σ_1 occurs when the channel is in the "good" state and the state σ_{b+2} when the verification chain is broken. The states $\sigma_2, \sigma_3, \ldots, \sigma_{b+1}$ are the states corresponding to the bursts of length $1, 2, \ldots, b$, respectively. Using the original Gilbert-Elliot model, we can find the respective transition probabilities. They are given by the transition probability matrix:

$$\mathbf{P}_{\sigma\sigma} = \begin{bmatrix} 1-q & q & 0 & 0 & 0 & \cdots & 0 \\ p & 0 & 1-p & 0 & 0 & \cdots & 0 \\ p & 0 & 0 & 1-p & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p & 0 & 0 & 0 & 0 & \cdots & 1-p \\ p & 0 & 0 & 0 & 0 & \cdots & 1-p \end{bmatrix}. \qquad (10)$$

The corresponding graph is given in Fig. 8.

As the resulting $(b+2)$ – state Markov chain is a stationary one, (the matrix $\mathbf{P}_{\sigma\sigma}$ does not depend on the number of a currently transmitted packet) we can find the vector of stationary probabilities $\mathbf{P}_\sigma$ by solving the set of $(b+2)$ linear equations:

$$\begin{cases} \mathbf{P}_\sigma = \mathbf{P}_\sigma \mathbf{P}_{\sigma\sigma} \\ \sum_{i=1}^{b+2} p_{\sigma i} = 1 \end{cases}. \qquad (11)$$

The probability $p_{\sigma(b+2)}$ is the stationary probability that during the transmission of a packet, the verification chain is broken.

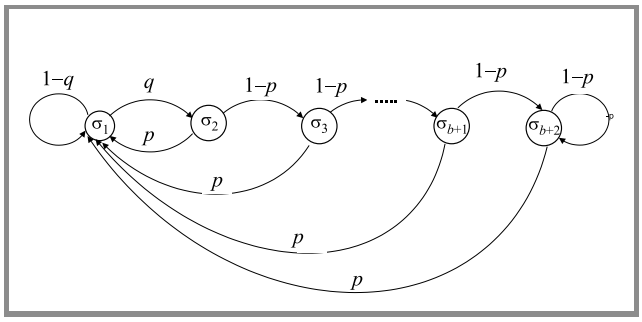

Fig. 8. Graph of the Markov chain given by matrix (9).

Having found the probability $p_{\sigma(b+2)}$, we can calculate the probability $P_K\{break\}$ that during the transmission of K packets the verification chain is broken at least once. Those K packets are assumed as transmitted in the steady state, i.e. they do not include first b packets in the stream. It is given by:

$$P_K\{break\} = 1 - (1 - p_{\sigma(b+2)})^K . \tag{12}$$

This will be illustrated by the following example.

Example 2. Let us consider here the same stream transmitted as in Example 1 through the same channel. We will find the probabilities that the verification chain will be broken during the transmission of a single sequence and the whole stream of 50 sequences. We compare the results for the cases when the authentication chain is created to be immune against bursts of lengths 3, 5, and 8.

To find answers to the problem, we first need to construct the relevant transition probability matrices and find the stationary probabilities of the states corresponding to the break in the verification process. While for $b = 3$, this is still a feasible task to find the stationary probabilities manually, as

$$\mathbf{P}_{\sigma\sigma} = \begin{bmatrix} 1-q & q & 0 & 0 & 0 \\ p & 0 & 1-p & 0 & 0 \\ p & 0 & 0 & 1-p & 0 \\ p & 0 & 0 & 0 & 1-p \\ p & 0 & 0 & 0 & 1-p \end{bmatrix} =$$

$$= \begin{bmatrix} 0.99 & 0.01 & 0 & 0 & 0 \\ 0.30 & 0 & 0.70 & 0 & 0 \\ 0.30 & 0 & 0 & 0.70 & 0 \\ 0.30 & 0 & 0 & 0 & 0.70 \\ 0.30 & 0 & 0 & 0 & 0.70 \end{bmatrix} . \tag{13}$$

However, it becomes more cumbersome for the cases when $b = 5$ or $b = 8$. Any standard software package can be used to provide solution to the set of linear equations. We used here MATLAB, and the results are given in Table 1.

After that, we can use formula (12) to find the probabilities of breaking the verification stream for a single sequence of 50 packets, and for the whole stream of 50 sequences. The results are given in Table 2.

Table 1
Stationary probabilities of breaking the verification chain for the case considered in Example 2

Verification chain immune against bursts of length b	Stationary probability of a break in the verification chain
3	0.0111
5	0.0054
8	0.0019

Table 2
Probabilities of the verification chain being broken during the transmission of a single sequence of 50 packets and the whole stream of 50 sequences for the Example 2

Verification chain immune against bursts of length b	Probability of breaking verification chain in a single sequence	Probability of breaking verification chain in a whole stream
3	0.4277	1.0000
5	0.2372	1.0000
8	0.0907	0.9913

The results from Example 2 indicate that the stationary probabilities of breaking the verification chain are quite low, as shown in Table 1, and decrease with an increase in the maximum length of burst to which the verification chain is immune to. However, the probabilities of having at least one break in the verification chain for the transmission of whole stream are very high, approaching 1 in all considered cases. This is due to the fact that the considered channel is characterized with the high value of a transition from the "good" to the "bad" state and equal to 0.01. This value is more likely to be experienced in wireless channels. However, with the proliferation of multimedia services to the wireless environment such channels need to be considered while designing authentication algorithms for streamed data.

5. Conclusions

In the paper we presented a modification to the Golle and Modadugu authentication chain [10] that results in a chain resistant to losses of packets containing sequence signatures. The modification is achieved without any additional computational or communication overheads. In addition, we have introduced here a Markov chain method to analyze performance of authentication schemes. It draws from the fact that channel with burst losses can be modeled by a two state Markov process, referred to as the Gilbert-Elliot model. The method can be very useful in analyzing performance of different authentication schemes under realistic channel conditions.

In the future, we expect to perform such an analysis for other hash chain based schemes taking into account multiple bursts. This will of course result in more complicated Markov chains of the verification process.

References

[1] W. Stalings, *Data and Computer Communications*. 6th ed. Upper Saddle River: Prentice Hall, 2000.

[2] J. Lu, "Signal processing for Internet video streaming", *Proc. SPIE, Image and Video Commun. Proc.*, Jan. 2000.

[3] G. Pipkin, "Video on the web", http://www.itc.virginia.edu/atg/techtalks/powerpoint/video/sld001.htm (accessed on 23/11/01).

[4] "Utah education network", http://www.uen.org/technical/html/streamingfaq.html (accessed on 23/11/01).

[5] R. Gennaro and P. Rohatgi, "How to sign digital streams", in *Cypto '97*. Springer-Verlag, 1998, pp. 180–197.

[6] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts", *IEEE/ACM Trans. Netw.*, vol. 7. no. 4, pp. 502–513, 1999.

[7] M. J. Moyer, J. R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications", *IEEE Network*, pp. 12–23, Nov./Dec. 1999.

[8] D. R. Stinson, *Cryptography Theory and Practice*. CRC Press.

[9] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels", in *Proc. IEEE Secur. Priv. Symp.*, May 2000, pp. 56–73.

[10] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss", in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 8–9 Feb. 2001.

[11] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss", http://www.isoc.org/isoc/conferences/ndss/01/2001/papers/golle.pdf (accessed on: 12/02/02).

[12] W. Stalings, *Cryptography and Network Security: Principles and Practice*. 2nd ed. Upper Saddle River: Prentice Hall, 1998.

[13] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[14] P. Golle, "Authenticating streamed data in the presence of random packet loss", power point presentation, http://crypto.stanford.edu/p̃golle/papers/auth.html (accessed on: 10/12/01).

[15] S. Miner and J. Staddon, "Graph-based authentication of digital streams", http://www-cse.ucsd.edu/users/sminer/abstracts/GraphAuth.html (accessed on: 11/01/02).

[16] E. N. Gilbert, "Capacity of burst-noise channel", *Bell Syst. Techn. J.*, pp. 1253–1265, Sept. 1960.

[17] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels", *Bell Syst. Techn. J.*, pp. 1977–1997, Sept. 1963.

[18] J. M. Boyce and R. D. Gaglianello, "Packet loss effects on MPEG video sent over the public Internet", in *ACM Multimedia '98, Electron. Proc.*

Beata Joanna Wysocki graduated from Warsaw University of Technology receiving her M.Eng. degree in electrical engineering in 1991. In 1994, she started the Ph.D. study in the Australian Telecommunications Research Institute at Curtin University of Technology. In March 2000, she was awarded her Ph.D. for the thesis: "Signal Formats for Code Division Multiple Access Wireless Networks". During the Ph.D. studies, she was involved in a research project "Wireless ATM Hub" at the Cooperative Research Centre for Broadband Telecommunications and Networking, and worked as a research assistant at Edith Cowan University, within the ARC funded "CDMA with enhanced protection against frequency selective fading", and "Reliable high rate data transmission over microwave local area networks". Since October 1999 she has been with the Telecommunications & Information Technology Research Institute at the University Wollongong as a research fellow. Her research interests include sequence design for direct sequence (DS) code division multiple access (CDMA) data networks and error control strategies for broadband wireless access (BWA) systems.
e-mail: beata@elec.uow.edu.au
University of Wollongong
Northfields Avenue
Wollongong, NSW 2522, Australia

Yejing Wang received a Ph.D. in computer science in 2001 and is currently a research fellow in the School of Information Technology and Computer Science of the University of Wollongong. Her research interest include cryptography and algebraic coding.
University of Wollongong
Northfields Avenue
Wollongong, NSW 2522, Australia

Reihaneh Safavi-Naini is a Professor of Computer Science in University of Wollongong. She received a Ph.D. in electrical and computer engineering from University of Waterloo in Canada. Her research interests include, cryptography, computer and communication security and digital right management.
University of Wollongong
Northfields Avenue
Wollongong, NSW 2522, Australia