國立交通大學

資訊科學與工程研究所

碩 士 論 文

安全且快速訊息驗證的廣播認證機制

Secure and fast message verification scheme for multicast stream authentication

研 究 生：陳證傑

指導教授：曾文貴　教授

中 華 民 國　九 十 八 年 六 月

安全且快速訊息驗證的廣播認證機制

Secure and fast message verification scheme for multicast stream authentication

研 究 生：陳證傑　　Student：Cheng-Chieh Chen

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 系

碩 士 論 文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

安全且快速訊息驗證的廣播認證機制

學生：陳證傑　　指導教授：曾文貴　博士

資訊科學與工程研究所

國立交通大學

摘要

在網際網路上，一台伺服端想廣播大型的資料，如電影或即時新聞，給多個使用者；使用者則希望確認所收到的資料確實是由伺服端所發出。在這樣的情形下，為了確保整體傳輸效能，只能使用不可靠的 UDP 通訊協定，因而導致在傳送資料的過程中，可能會有部分封包遺失，增加使用者驗證封包的困難。由於資料亦可能遭到攻擊者的竄改，直覺的做法是伺服端對每個資料做數位簽章，使用者以伺服端的公開金鑰對封包做認證。但是數位簽章的簽名與驗證皆要花費相當長的計算時間，如何設計出安全且有效率的廣播認證機制，便成為一項有趣且實際的研究議題。

在本篇的方法中，由於接收端的計算資源可能不夠或者不是全部用在對資料的認證計算，對於來不及做驗證的資料會暫時存在記憶體，若是可用記憶體不足，接收端會將接下來的封包丟棄，造成不必要的封包遺失。若是攻擊者傳送大量偽造封包給接收端，則接收端的計算與儲存資源會更加不足，封包的遺失也會更嚴重。我們提出的方法除了使接收端能驗證資料的正確性，還能以花費少量的計算來對封包做驗證，進而減輕接收端的計算與儲存負擔。

Secure and Fast Message Verification Scheme for Multicast Stream Authentication

Student : Cheng-Chieh Chen Advisor : Dr. Wen-Guey Tzeng

Department of Computer Science

National Chiao Tung University

Abstract

We propose two schemes for the multicast stream authentication problem. In this paper, we focus on the computational overhead of receiver. From the point of view of the receiver, receiver may execute many programs simultaneously. Thus, the computational resource is not totally used for packet authentication. If the computational resource is not sufficient, the buffer may overflow and the receiver drops the incoming packets. Moreover, an attacker may forge many invalid packets and send them to receiver to exhaust the computational resource of receiver. With this attack, the problem of buffer overflow becomes more serious. In this paper, we propose two schemes that reduce the computational cost of receiver. Our schemes also achieve data integrity, non-repudiation, individual packet authentication, robust to packet loss and reasonable storage overhead. The first scheme is suitable when the packet loss rate is low. However, the loss rate of network may be high, we the second scheme which is more efficient than the first scheme. Our schemes are robust against the injection attack.

Keywords: Multicast stream authentication, injection attack

誌　　謝

首先感謝我的指導老師曾文貴教授，在我碩士班兩年間的學習過程中，帶領我深入密碼學的領域。老師認真積極的教學態度，使我受益良多。另外我要感謝口試委員，交大黃育綸教授、交大蔡錫鈞教授與清大孫宏民教授，在論文上給我許多建議與指導，讓我的論文更加完善。除此之外，我也要感謝實驗室學姐林孝盈、實驗室學長朱成康、林煥宗、沈宣佐與曾智揚給我的指導，以及實驗室學弟和其他的碩士班同學的幫忙。

最後，我要感謝我的家人以及清交伙食團，不論在精神或物質上都給我極大的支持，讓我在無後顧之憂的情況下可以順利完成學業。在此，謹以此文獻給所有我想要感謝的人。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

A trusted server wants to transmit a large file such as movie over the Internet to many receivers simultaneously. The file is divided into many small messages and sent out as packets. On getting a message, the receiver has to process it in real-time. However, the receiver doesn't know whether the message is actually produced by the trusted server or forged by a malicious attacker. Thus, every packet is composed of original message and authentication information such as digital signature. It is required for a receiver to have an efficient and secure way to check the validity of received messages.

Unicast is not a choice when a sender wants to transmit the same data to many receivers at the same time. Multicast is an efficient protocol for delivering packets to many receivers that belong to the same multicast group simultaneously. This protocol is suitable for some real-time applications such as stock quote or live video broadcast. In these time-sensitive applications, it is impractical to use reliable transmission protocol, because the packet is retransmitted whenever the packet loss occurs in one of these receivers. This slows down the transmission rate at sender side. Therefore, the UDP is used in real-time multicast. However, as the UDP is unreliable, some packets may lose at receiver side. The packet loss incurs some inconvenience when the receiver verifies the received packets. Furthermore,

it is well known that the Internet is insecure. The attacker can eavesdrop, inject, delay, replay and capture packets. The basic requirement in multicast authentication is that the receiver can make sure that whenever he receives a packet, he can decide whether this packet is produced by the trusted server or by the malicious attacker.

1.2 Trivial solutions

The packets transmitted over the Internet may be modified by the attacker. There are some cryptographic primitives such as MAC [10] (message authentication code) and digital signature scheme. The MAC provides data integrity and authenticity. Data integrity means a receiver can make sure the received data has not been changed. Authenticity means an authorized receiver can make sure the packet is coming from the trusted sender. The digital signature provides data integrity and non-repudiation. Non-repudiation means if a signer has signed a message, he can't deny this signature. In MAC, two parties R and S share the same secret key and use it to compute the message digest. When S sends the message along with the digest to R, R can make sure this message is coming from S because only the two parties know the secret key. Besides, the MAC is a symmetric primitive, the computational and communication cost is low. However, this solution is not applicable in the multicast authentication situation. Since all receivers can access the same secret key, any one of the receivers can compute valid message digest and claim that the message is produced by the sender. Another naive way is that the sender uses a digital signature scheme such as RSA to sign each message using his private key. The receiver verifies the messages using the public key of the sender. Digital signature scheme is believed to be

secure. This solution provides adequate authentication, but the computational cost of digital signature is too expensive to make this solution impractical.

1.3 Design issues

There are many additional requirements when designing the algorithm for the multicast stream authentication problem.

- Low computational overhead: Since the packets must be sent and verified as fast as possible.
- Low communication overhead: For receivers to authenticate a message, a packet is composed of message and authentication information. The size of authentication information should be as small as possible to avoid network congestion.
- Low storage overhead: The buffer or unused memory of receiver is limited. When the buffer is full of packets, the receiver drops the incoming packets, causing unnecessary packet loss.
- Robust to packet loss: When packet loss occurs, the receiver can still verify the incoming packets.
- Individual packet authentication: If the authentication of one packet P_1 depends on another packet P_2 received later, the receiver must keep P_1. If P_2 is lost, P_1 is useless. Thus, if a scheme achieves individual packet authentication, the receiver can verify a packet immediately.

Our contribution. In this paper, we focus on the computational cost of receiver. We propose two schemes that reduce the computational cost of receiver.

These schemes also achieve data integrity, non-repudiation, individual packet authentication, robust to packet loss and reasonable storage overhead. From the point of view of the receiver, receiver may execute many programs simultaneously. Thus, the computational resource of the receiver is not totally used for packet authentication. As a result, there may be some packets that can't be checked immediately. These packets are temporarily stored in the buffer of receiver. If the computational resource is not sufficient, the buffer may overflow and the receiver drops the incoming packets. Moreover, a type of denial of service attack is called the injection attack or the pollution attack [1]. The attacker forges many invalid packets and sends them to receiver to exhaust the computational resource of receiver. With this attack, the problem of buffer overflow becomes more serious. From the point of view of the sender, the sender just broadcasts packets. The sender doesn't receive any packet. There is no denial of service attack at sender side. And it is reasonable to assume that the computational power of sender is stronger than the receiver

The rest of paper is organized as follows: In Chapter 2, we introduce some related works. In Chapter 3, we give some background information for this paper. In Chapter 4, we propose our two schemes. In Chapter 5, we give the simulation results. In Chapter 6, we give the conclusion and future work of this paper.

Chapter 2

Related work

Previous works on multicast stream authentication problem are roughly divided into two classes. The first class of schemes is called the signature amortization schemes, and the second class of schemes is called the MAC-based schemes. We introduce some signature amortization schemes in section 3.1 and then describe some MAC-based schemes in section 3.2.

2.1 Signature amortization schemes

The signature amortization schemes compute single signature over several packets and reduce the computational overhead greatly. Hash chain [2], hash tree [3], and SAIDA [4] are examples of signature amortization schemes.

2.1.1 Hash chain

Early work on signature amortization was done by Gennaro and Rohatgi [2]. Let || denotes string concatenation, and H(.) denotes a collision resistant hash function. Assume the sender knows the whole stream. The stream is finite and can be divided into n messages. We denote the stream as $S = M_1||M_2|| \dots ||M_n$. Each packet P_i transmitted over the network is composed of the message M_i representing the original data of stream and the hash of the next packet. The

construction of the packets works backwards as follow:

$$P_i = \begin{cases} H(P_{i+1})||Sign_{SK}\big(H(P_{i+1})\big), i = 0 \\ M_i||H(P_{i+1}), 1 \le i \le n-1 \\ M_i, i = n \end{cases}$$

If P_i is a valid packet, then the receiver has the valid hash value of P_{i+1}. When the receiver gets P_{i+1}, it takes just one hash operation to verify P_{i+1}. The computational cost of receiver is very low.

This approach has constant communication and computational overhead. We use the hash chain approach as part of our scheme. However, it doesn't achieve individual packet authentication. When a packet loss occurs, the receiver doesn't get the hash of the next packet. The receiver can't authenticate the remaining packets.

2.1.2 Merkle hash tree

Wong and Lam [3] used the Merkle hash tree to deal with the problem of multicast stream authentication. A Merkle hash tree is a binary hash tree that takes *n* leaves as input. A leaf is the hash value of a message. The internal node of a Merkle hash tree is defined as the hash value of the concatenation of its two children. The Merkle hash tree is used to compute a single hash digest over several messages. Figure 2-1 shows a Merkle hash tree when *n*=8.

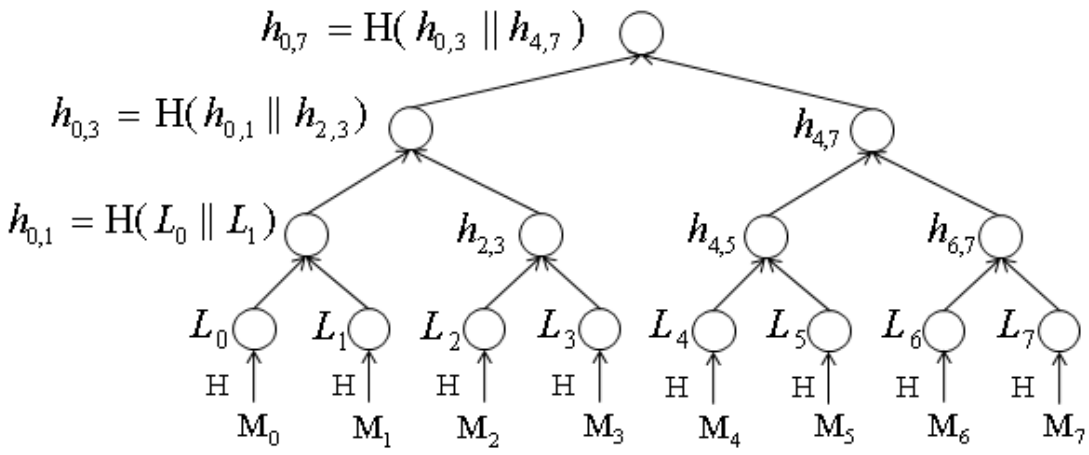

Figure 2-1: Merkle hash tree.

We define authentication path AP(i) as the concatenation of the $\log_2 n$ hash values needed to reconstruct the path from L_i to the root of the tree. For example, in Figure 2-1 we have AP(0)= $L_1||h_{2,3}||h_{4,7}$. Using AP(0), we can reconstruct the root as $h_{0,7} = \mathrm{H}(\mathrm{H}(\mathrm{H}(L_0|| L_1)||h_{2,3})||h_{4,7})$. By the property of second-preimage resistant of hash function, the attacker cannot change the value of any node of the tree. Thus, each message with its authentication path is individually verifiable. In Wong and Lam's scheme, the stream can be finite or infinite and can be divided into blocks of n messages. The stream can be denoted as $S= (B_0||B_1|| \ldots)$. Each block consists of n messages. The i-th block is $B_i = (M_{in}||M_{in+1}||\ldots||M_{in+n-1})$. We focus on the first block B_0. A packet P_i is composed of the message M_i, the authentication path AP(i) and the signature of the root of Merkle hash tree. Each authentication path of a packet is unique and an attacker can't change it. Thus, this scheme achieves individual authentication. However, assume the receiver doesn't store the authentication path, it takes $\log_2 n$ hash operations to verify a packet. Even if the receiver stores the authentication path, the computational cost of receiver can be reduced. In our paper, we combine the hash chain scheme and the Merkle hash tree scheme to obtain schemes that achieve low computational cost at

receiver side and individual packet authentication. In our paper, we call this classical Merkle hash tree scheme the scheme-0.

2.1.3 SAIDA

SAIDA (Signature Amortization using the Information Dispersal Algorithm) [4] [11] is a scheme based on erasure code [12][13][14][15]. The erasure code is composed of an encoder and a decoder. The (n, t)-encoder takes a block of n messages as input and output n symbols. When the receiver get at least t of these symbols, the receiver can decode these symbols back to the original block of messages. Thus, erasure code is robust to any pattern of loss among the symbols as long as the receiver gets sufficient number of symbols. We describe the construction of SAIDA as follow. Assume the stream is also divided into blocks of n messages and can be finite or infinite. We also focus on the construction of the first block. The sender computes $H_0 = H(M_0)||H(M_1)||\ \ldots\ ||\ H(M_{n-1})$, the concatenation of the hash values of all messages of this block. In the second step, the sender computes the signature of the hash of H_0, i.e. S_0=Sign(SK, $H(H_0)$). Then the erasure encoder encodes $H_0||S_0$ into n symbols $(s_0, s_1, \ldots, s_{n-1})$. Finally, the sender appends the symbols to each packet and outputs n packets such that $P_i = M_i||s_i$ for i=0, 1, … , n-1. This scheme is robust to packet loss. Unfortunately, if the attacker injects forged packets into the communication channel, the receiver will decode an invalid H_0 that can't pass the verification algorithm of digital signature. If the receiver tries to reconstruct a valid H_0 by decoding all possible combinations of received symbols, it is computationally expensive for him to do this.

2.2 MAC based schemes

The MAC-based schemes use the symmetric primitives to design an efficient scheme that achieves asymmetry.

2.2.1 Timed Efficient Stream Loss-Tolerant Authentication (TESLA)

In [5], Perrig *et al* proposed the simplest TESLA protocol. Let F and F' be two pseudo random functions. Assume $S = M_1 || M_2 || M_3 ...$ is the stream. The sender generates every packet as follows:

$$P_i = M_i||F(K_{i+1})||K_{i-1}||MAC\ (K_i'\ , M_i||F(K_{i+1})||K_{i-1})$$

$$K_i' = F'(K_i)\ ,\ K_i \text{ is a random number}$$

The receiver uses F' to compute the keys of MAC. To authenticate a packet P_i, assume that the packet P_{i-1} has been authenticated. When receiving P_{i+1}, the receiver checks K_i contained in P_{i+1} using the value $F(K_i)$ contained in P_{i-1} first. Then the receiver computes $K_i' = F'(K_i)$ as the MAC key of P_i. Finally, the receiver checks the validity of P_i using the MAC. If all verifications are successful, P_i is a valid packet. The verification process can be illustrated in Figure 2-2.

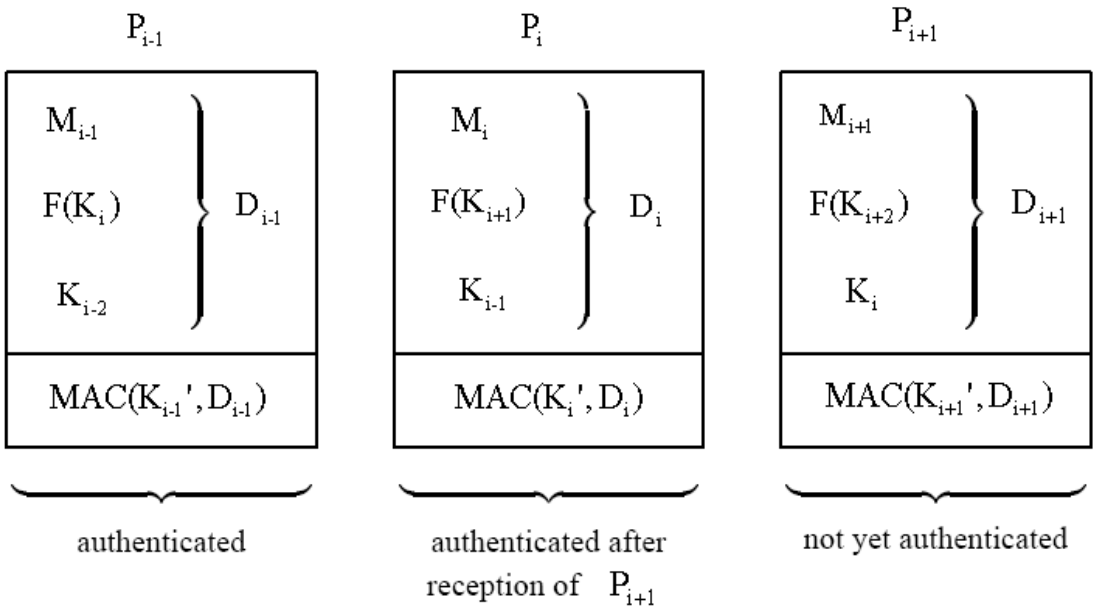

Figure 2-2: TESLA.

The simplest TESLA scheme has two shortcomings. First, if P_{i+1} is sent before the receiver gets P_i. An attacker can intercept P_i and P_{i+1}, and then he can use K_i disclosed in P_{i+1} to forge an invalid P_i without the detection of the receiver. To avoid this type of attack, when sending a packet P_i, the sender must wait until all receivers get P_i, then he sends the next packet P_{i+1}. This slows down the transmission rate. Second, this scheme can't tolerate a single packet loss. If the receiver doesn't get P_{i-1}, he can't authenticates P_i even if he get P_{i+1} because he doesn't know whether K_i is valid or not.

2.2.2 Modified TESLA

In order to avoid the second shortcoming described above, Perrig *et al* proposed the modified TESLA protocol. Let $F^2(x) = F(F(x))$ denotes two consecutive operations of F on input x. In modified TESLA, the sender chooses a number n and a MAC key K_n randomly first. Then he computes a sequence of n

MAC keys $K_{n-1}, K_{n-2}, \ldots, K_0$ where $K_i = F^{n-i}(K_n)$ for i=0, 1, … , n-1. We get a key chain. Figure 2-3 shows the generation of key chain and the construction of modified TESLA.

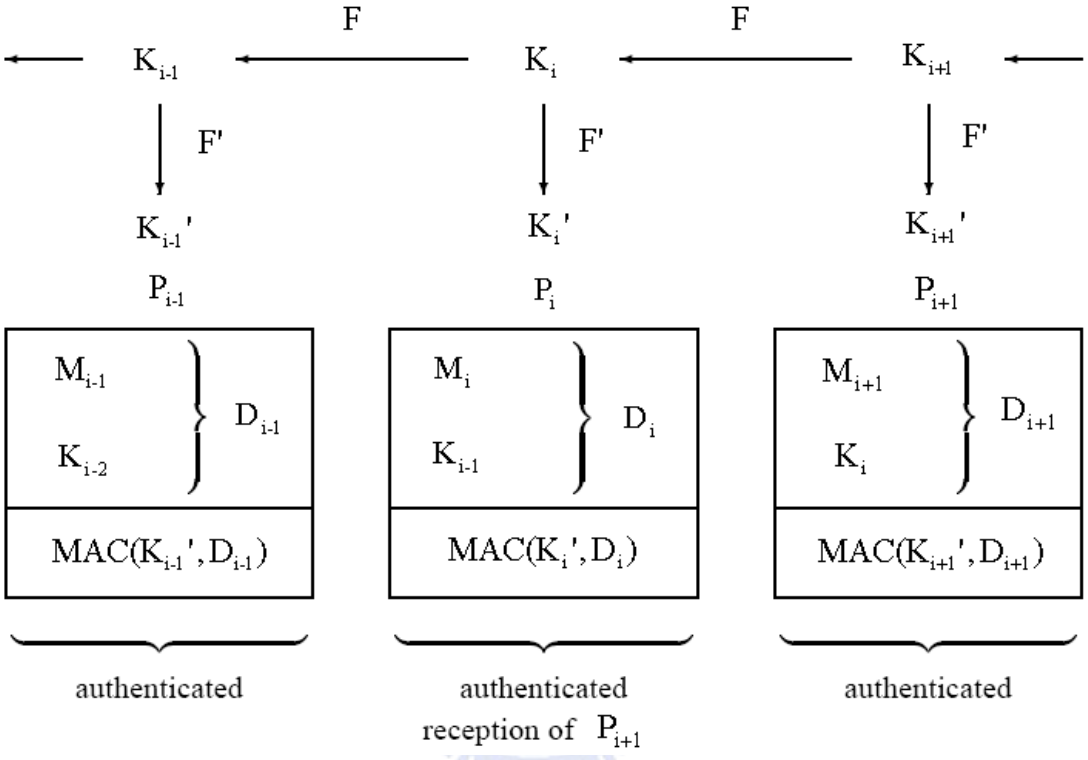

Figure 2-3: Modified TESLA.

The key chain can tolerate any pattern of packet loss because when a receiver gets the latest MAC key K_i, he can compute all keys from K_{i-1} to K_0. Since F is a one way function, it is hard for an attacker to compute K_{i+1} given K_i.

To deal with the first problem mentioned above, the sender discloses K_i in P_{i+d} rather than P_{i+1}. However, before the receiver gets P_{i+d}, he can't authenticates P_i. He has to store d packets in the process of verification. When the attacker injects many invalid packets, the problem of buffer overflow is serious.

From the schemes mentioned above, designing a scheme that achieves all requirements is hard. Although in Wong and Lam's scheme, the communication

overhead is high, it has many outstanding properties:

- Each packet is individually verifiable.
- It is robust to any pattern to packet loss.
- It is robust to injection attack.
- It is robust to delay and capture attack.

It is clear that only the Merkle hash tree scheme meets these properties. We think that the Merkle hash tree scheme is one of the best schemes for multicast scheme authentication. However, we can improve the computational cost of receiver. Thus, we design our schemes based on Wong and Lam's scheme and the hash chain scheme.

Chapter 3

Preliminaries

3.1 Collision resistant hash function

A collision resistant hash function H is a function which maps a message M of arbitrary size to a message digest of fixed size. H has the following properties:

- Given an input x, it is efficient to compute H(x).
- Given a hash value y, it is computationally infeasible to find x such that H(x)=y.
- Given an input x, it is computationally infeasible to find another x' such that $x' \neq x$ and $H(x') = H(x)$.
- H is deterministic.

Because of the collision resistant property, the modification of the message will result in a different hash value.

3.2 Packet loss model

In our simulation, we use two different packet loss models to simulate the performance of scheme-0 and our two schemes. One is the uniform packet loss model. The other is introduced by Gillbert[6] and Elliott [7], which is called the two-state Markov Chain (2-MC) loss model[9][11]. According to [8, 9], Paxson

and Yajnik *et al* showed that the pattern of loss in the Internet is bursty. To accurately simulate the loss pattern in the Internet, we use this model as part of our simulation result. The 2-MC model has two possible states, one is the “good state” and the other is the “bad state”. If the packet is lost, the transmission channel is in “bad state”, otherwise the channel is in “good state”. Figure 3-1 shows the 2-MC model.

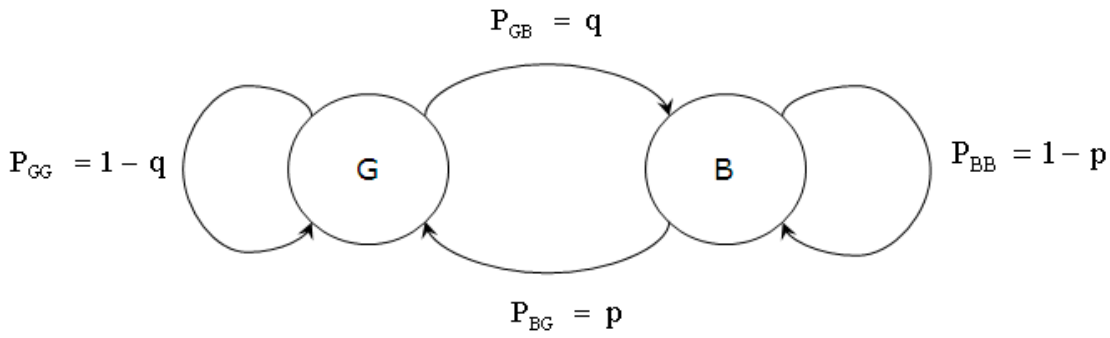

Figure 3-1: 2-MC model.

There are four transition probabilities (P_{GG}, P_{GB}, P_{BG} and P_{BB}). The stationary probabilities of good state and bad state are denoted as π_G and $\pi_B = 1 - \pi_G$ respectively. The probability transition matrix M is:

$$M = \begin{bmatrix} 1-q & q \\ p & 1-p \end{bmatrix}$$

The stationary probability vector is $V = [\pi_G, \pi_B]$ such that V=VM. We have

$$\begin{cases} \pi_G(1-q) + \pi_B p = \pi_G \\ \pi_G + \pi_B = 1 \end{cases}$$

Thus we get:

$$\pi_G = \frac{\mathrm{p}}{\mathrm{p+q}},\ \pi_B = \frac{\mathrm{q}}{\mathrm{p+q}}$$

The average probability of packet loss can be defined as $\mathrm{P_{avg}} = \pi_G \mathrm{P_{GB}} + \pi_B \mathrm{P_{BB}}$.

In [9], the parameter p is measured as 0.0496. The parameter 1-q is measured as 0.0513. If (1-q)>p, the packet loss is burstier than predicted by the uniform packet loss model and the 2-MC model is more accurate.

Chapter 4

Proposed schemes

We propose two schemes that are robust to packet loss and achieve lower computational overhead at receiver side. In section 4.1, we propose the first scheme called scheme-1 which is the simplest one. However, when the packet loss probability is not low, the computational cost of receiver also increases. In section 4.2, we propose a scheme called scheme-2 that is more efficient than scheme-1 when the packet loss of network is not low.

4.1 Scheme-1

We review some notations. The hash value of message M is H(M). The stream S can be divided into blocks, each block consists of n messages.

$$S = (B_0||B_1|| \dots)$$

Each block is denoted as $B_b = (M_{b\times n}||M_{b\times n+1}||\dots||M_{b\times n+n-1})$. The stream can also be finite or infinite. Because the same construction is performed on every block, we focus on the construction and verification of the first block ($B_0 = (M_0||M_1|| \dots ||M_{n-1})$). The construction of scheme-1 consists of three steps. In the first step, the sender constructs a hash chain as follow:

$$M_i' = \begin{cases} M_i||H(M_{i+1}'), 0 \le i \le n-2 \\ M_i, i = n-1 \end{cases}$$

In the second step, the sender uses M_i' to build a Merkle hash tree. For example, if n=8, the tree structure of scheme-1 is shown in Figure 4-1. Note that $L_i = H(M_i')$. AP(i) is the $\log_2 n$ hash values needed to reconstruct the path from L_i to the root of the tree.

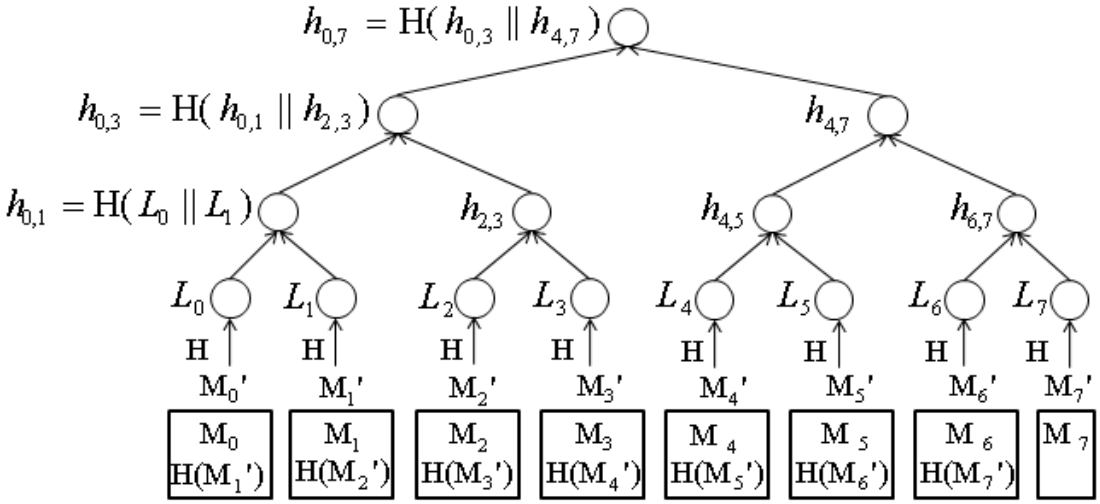

Figure 4-1: Tree structure of scheme-1.

In the third step, the sender signs the root of the tree. The signature of the root is denoted as Sign. We illustrate the content of the packets of the first block in Figure 4-2. Each packet is composed of M_i', authentication path and the signature of root. M_i' is composed of original message M_i and $H(M_{i+1}')$. When M_i' is verified, M_i is verified.

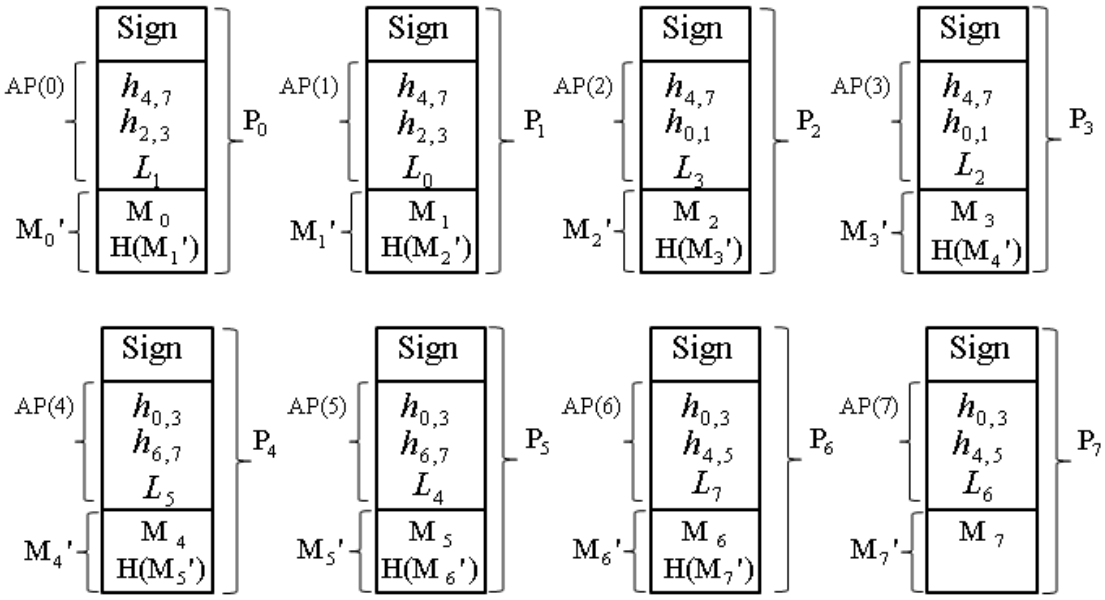

Figure 4-2: Packet format of scheme-1.

The verification is easy. Assume each packet has sequence number and block number. Each packet in block B_b has block number b. The i-th packet of a block has sequence number i-1. The sequence number and block number are part of the original message M_i and can be used to point out the position at the stream. For example, in Figure 4-2, the block number of P_2 is 0 and the sequence number is 2. In the first block, when getting the first packet P_0, the receiver uses AP(0) and Sign to verify M_0'. The receiver has valid $H(M_1')$. Whenever P_i is verified, the receiver records the sequence number *i* of P_i as the latest valid sequence number and updates the hash chain information as $H(M_{i+1}')$. If the next received packet is P_{i+1}, the receiver uses $H(M_{i+1}')$ to verify M_{i+1}'. The receiver doesn't have to verify M_{i+1}' by using AP(i+1). It takes just one hash operation to verify a message. In the other case, if the receiver doesn't receive P_i, he doesn't have $H(M_{i+1}')$ and the hash chain is broken. When receiving P_{i+1}, the receiver has to use AP(i+1) of P_{i+1} to verify M_{i+1}'. If the receiver doesn't cache the authentication path information, it takes $\log_2 n$ hash operations to verify a

message. Every packet can be verified individually. When the packet loss probability is very low, the computational cost of receiver is low.

In scheme-1, the communicational overhead of every packet is $1+\log_2 n$ hash values. When the packet loss probability increases, the verification time of scheme-1 goes up. In some situation like stock quotes, even when the packet loss probability is not low, the receiver still wants to receive the packet because of the importance of packets. We design a scheme that is also efficient and robust against injection attack. So we propose scheme-2 that is more efficient and robuster to injection attack than scheme-1.

4.2 Scheme-2

The construction of scheme-2 is based on scheme-1. Before the description of scheme-2, we define a term: sub-tree. The tree structure of scheme-1 can be roughly illustrated in Figure 4-3. There are many smaller Merkle hash trees T_j. Every T_j is called sub-tree. Every sub-tree has λ leaves. Note that $\lambda = 2^x$ and x is a non-zero integer. The root of T_j is $h_{\lambda\times j,\lambda\times j+\lambda-1}$. λ can be greater than the average consecutive loss length of the network.

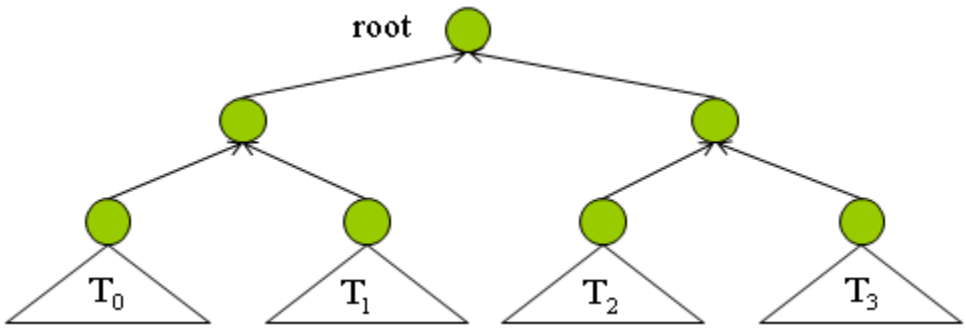

Figure 4-3: Tree structure of scheme-1.

The construction of scheme-2 is described as follow. Assume a block is

composed of n messages and every sub-tree has λ leaves. In the first step, the sender constructs a lower hash chain:

$$M_i' = \begin{cases} M_i || H(M_{i+1}'), 0 \le i \le n-2 \\ M_i, i = n-1 \end{cases}$$

In the second step, the sender constructs many sub-trees T_j using $(M_{j\times\lambda}', M_{j\times\lambda+1}', \ldots, M_{j\times\lambda+\lambda-1}')$ for j=0, 1…, (n/λ)-1. In the third step, the sender constructs a upper hash chain backwards:

$$M_j'' = \begin{cases} h_{\lambda\times j,\lambda\times j+\lambda-1} || H(M_{j+1}''), 0 \le j \le \left(\frac{n}{\lambda}\right) - 2 \\ h_{\lambda\times j,\lambda\times j+\lambda-1}, j = \left(\frac{n}{\lambda}\right) - 1 \end{cases}$$

In the fourth step, the sender constructs a upper Merkle hash tree using M_j'' for j=0, 1, …, (n/λ)-1, and then signs the root of the upper Merkle hash tree.

We illustrate an example. Assume every sub-tree has eight leaves and a block has 32 messages. Every root of sub-tree can be treated as an original message in scheme-1. We replace M_j with $h_{8\times j,8\times j+7}$ for j=0, 1, 2, 3 in scheme-1 and get Figure 4-4. The complete tree structure of scheme-2 is illustrated in Figure 4-5. M_j'' can be verified by using the authentication path for upper Merkle hash tree. Every original message M_i in M_i' can be verified by using the authentication path for sub-tree. Scheme-2 achieves individual packet authentication. We can use the upper and lower hash chains to reduce the computational overhead of receiver. The packet format of scheme-2 is illustrated in Figure 4-6.

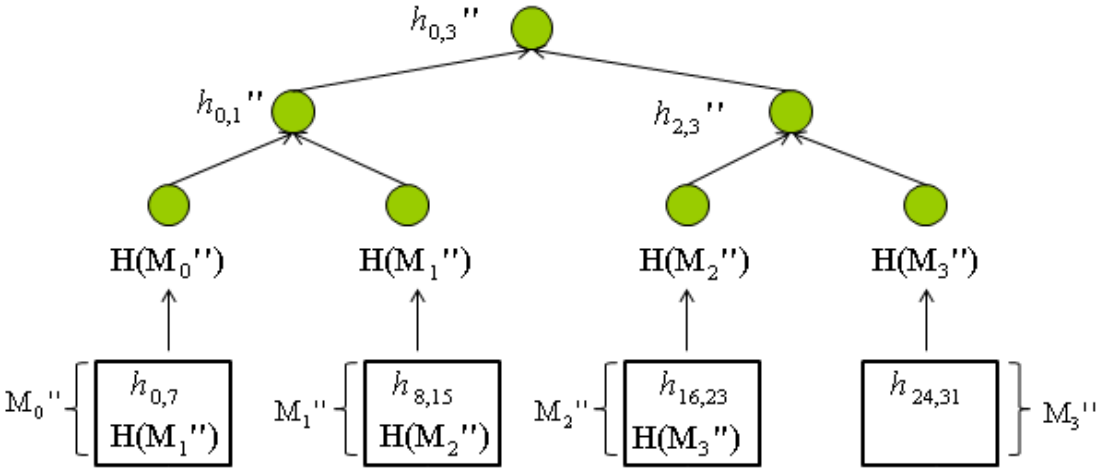

Figure 4-4: Upper part of tree structure of scheme-2.

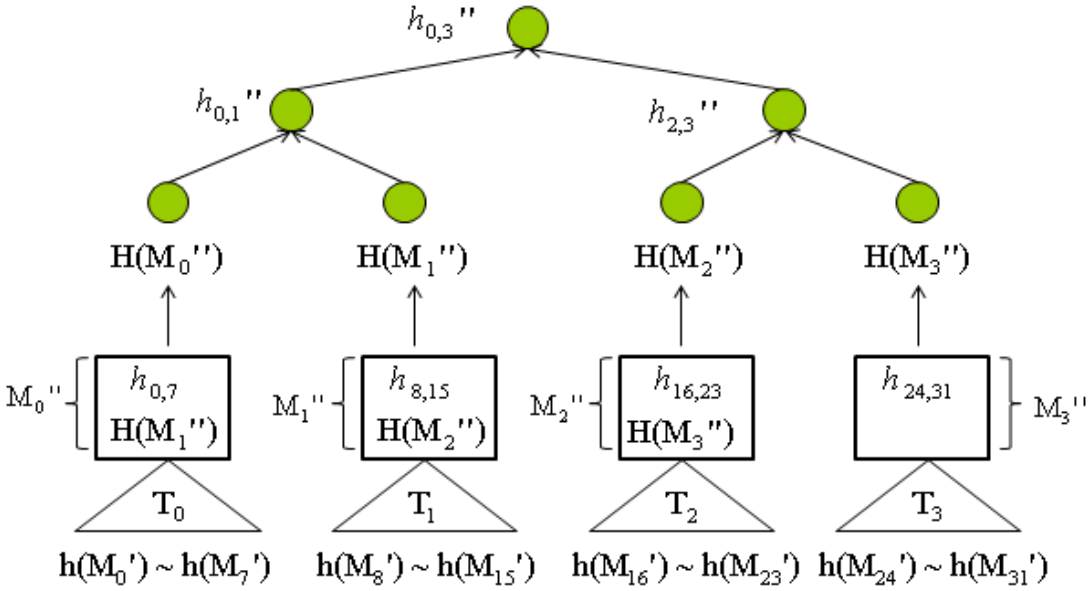

Figure 4-5: Complete tree structure of scheme-2.

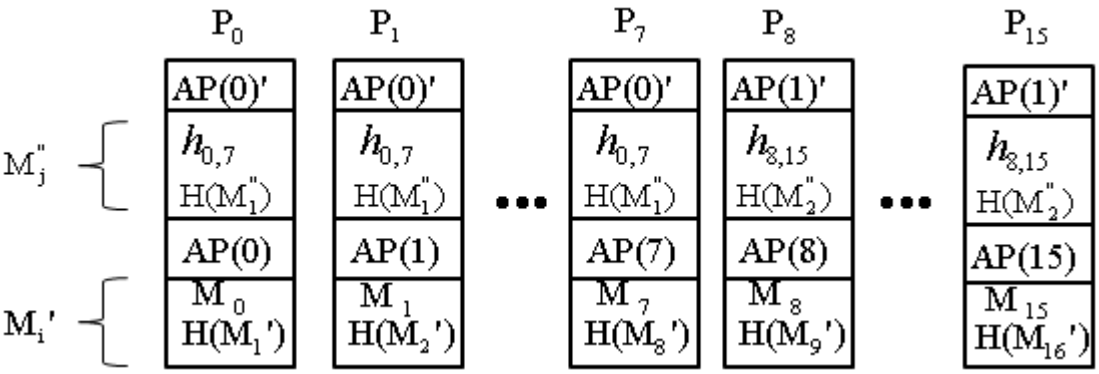

Figure 4-6: Packet format of scheme-2.

In Figure 4-6, the upper part of a packet is composed of the signature of root, the authentication path for upper tree and M_j''. Note that packets in the same sub-tree have the same upper part. M_i' contains the hash of M_{i+1}', which can be used to verify the message of the next packet. M_j'' contains the hash of M_{j+1}'', which can be used to verify the root of the next lower sub-tree. From scheme-1, we know that every M_j'' in Figure 4-4 can be authenticated. So, the root $h_{8\times j,8\times j+7}$ of lower sub-tree T_j can be authenticated. Using the root of the lower sub-tree, we can authenticate every message M_i as the manner of Merkle hash tree. Thus, each packet in the scheme-2 is individually verifiable. After authenticating the i-th packet, the receiver can use the lower hash chain to verify the message of the (i+1)-th packet just one hash operation. So, when the probability of packet loss is low, the scheme-2 is as efficient as the scheme-1. When the loss rate is not low, for example, in Figure 4-6 assume the receiver authenticates the fifth packet P_4 in the first sub-tree T_0 but doesn't receive P_5, P_6 and P_7. When receiving a packet P_9 in the second lower sub-tree, the receiver can use the upper hash chain to verify M_1'' in P_9 and authenticates $h_{8,15}$. After four hash operation, the receiver can check the validity of M_9 by using AP(9). The receiver doesn't have to authenticate M_1'' by using $AP(1)'$. In general case, the number of packets of a block ranges from 128 to 1024, in these cases, scheme-2 require lesser hash operation than scheme-1 at receiver side.

There is one additional advantage of the scheme-2. In scheme-0 and scheme-1 for the first packet of the block, the attacker may send many invalid first packet and claims that these packets are originated from the sender, we call this the first packet denial of service attack. When the receiver wants to verify the

validity of the first packet of a block, he has to conducts about $\log_2 n$ hash operations for each. If the attacker forges k first packets, the receiver has to do $k \times \log_2 n$ hash operations in scheme-0 and scheme-1. In scheme-2, the receiver executes the packet authentication from the middle of the tree. That is, when the receiver gets P_0 in Figure 4-5, he uses M_0'' and the authentication path ($H(M_1'')$ and $h_{2,3}'$) for upper tree to compute $h_{0,3}''$. If $h_{0,3}''$ is invalid, scheme-2 finds this error with lesser hash operations than scheme-0 and scheme-1.

Chapter 5

Comparison and experiment results

In this section, we compare Wong and Lam's scheme [3] to our two schemes. In section 5.1, we compare the verification times of different schemes under the uniform packet loss model. In section 5.2, we compare the verification times of different schemes under the 2-MC packet loss model in order to simulate the packet loss in real world. In section 5.3, we compare the verification times of different schemes under the injection attack. From section 5.1 to section 5.3, we assume the receiver caches the authentication path. We also assume the receiver just caches $\log_2 n$ hash values in scheme-0 where n is the number of packets per block. That is, the receiver doesn't cache all internal nodes of a tree. For example, in Figure 2-1, after authenticating P_5, the receiver stores $h_{0,7}$, $h_{4,7}$ and $h_{4,5}$. In scheme-1 and acheme-2, the way of caching is similar. The experiments were performed on a Pentium IV 1.66 GHz. We implement these schemes using JAVA. We use SHA-1 as the hash function and 512-bit DSA as the digital signature scheme. The number of packets in a block is 1024 and the original message of each packet is 512 bytes. In scheme-2,$\lambda = 8$.

5.1 Results under the uniform packet loss

Assume all types of attacks don't exist in the network. Figure 5-1 shows the average verification times under the uniform packet loss model. The verification

time of the Merkle hash tree scheme (or scheme-0) decreases when the loss probability increases because the number of packets needed to be checked decreases. When the loss probability increases, the verification time of scheme-1 approaches that of scheme-0 but doesn't exceed it. The verification time of scheme-2 is the lowest one because of the upper hash chain.

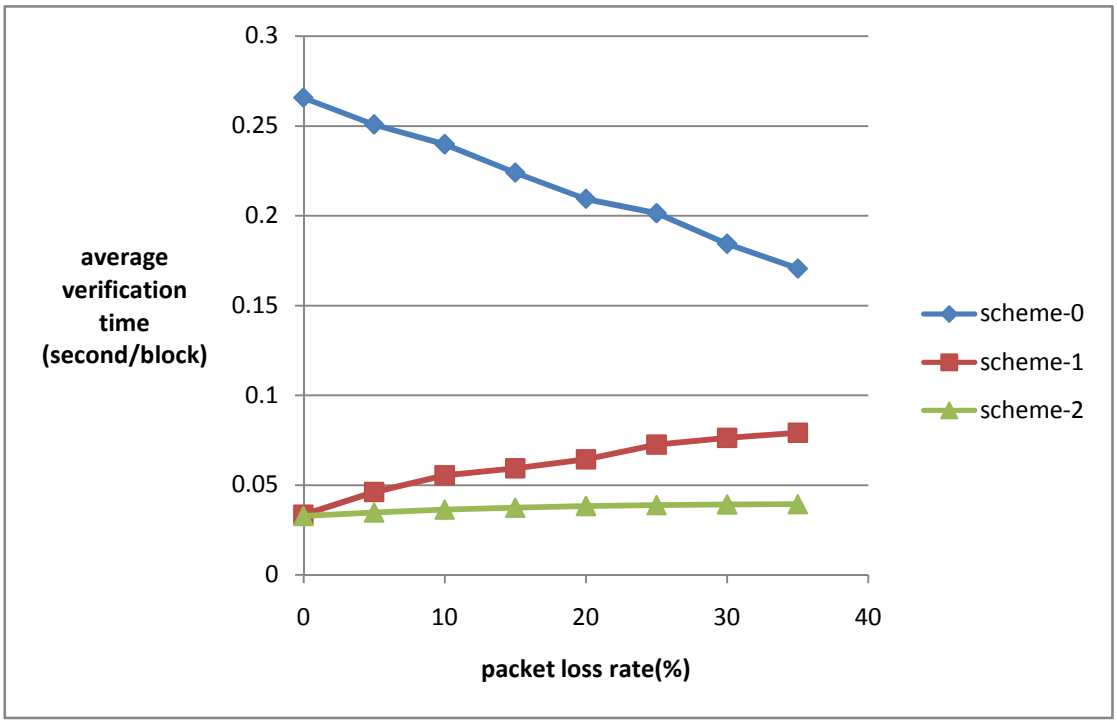

Figure 5-1: Results under the uniform packet loss.

5.2 Results under the 2-MC loss model

Assume all types of attacks don't exist in the network. Figure 5-2 shows the average verification times when the average length of burst loss is 8. We can see that the verification times of scheme-1 and scheme-2 are almost the same since the loss is bursty. However, scheme-2 is still the lowest one.

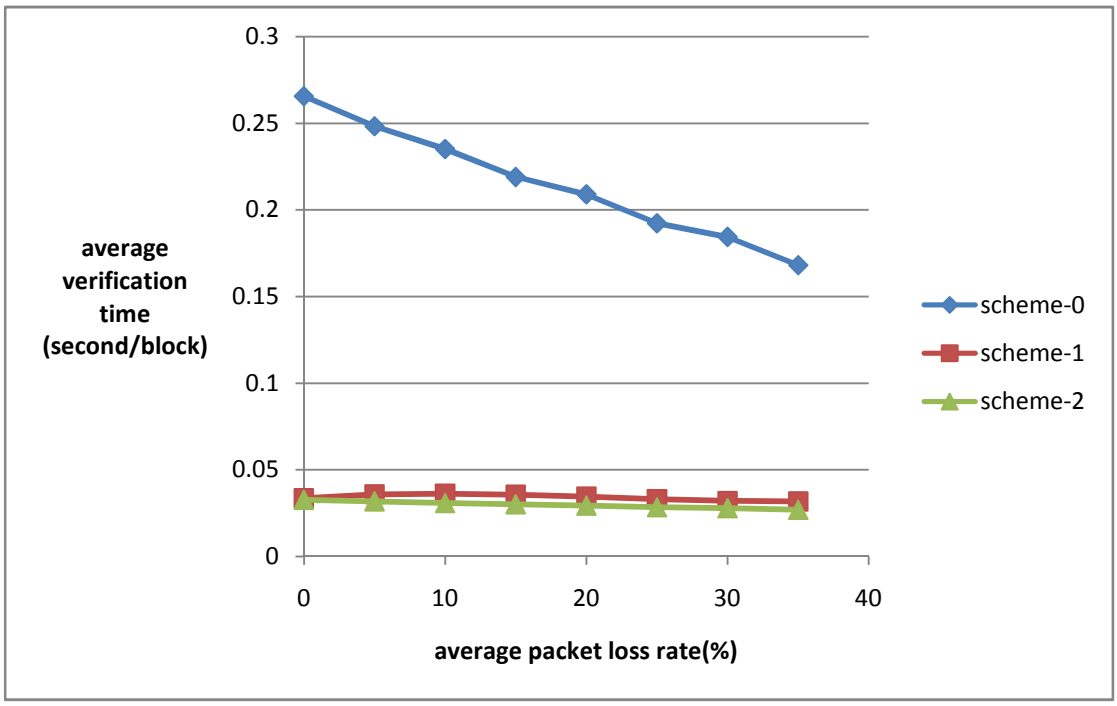

Figure 5-2: Results under the 2-MC loss model.

5.3 Results under the injection attack

We simulate the injection attack in this section. The uniform packet loss probability is 10%. In Figure 5-3, when the injection factor is 2, for every packet the receiver gets one packet produced by the trust sender and two invalid packets produced by the attacker. For every packet the attacker just generates two packets randomly and sends them to receiver in order to lunch denial of service attack. In Figure 5-3, the verification time of each scheme is proportional to the injection factor. The slope of scheme-1 is lower than the scheme-0, and the slope of scheme-2 is the lowest one. It means that scheme-2 is the robustest scheme among these schemes against the injection attack.

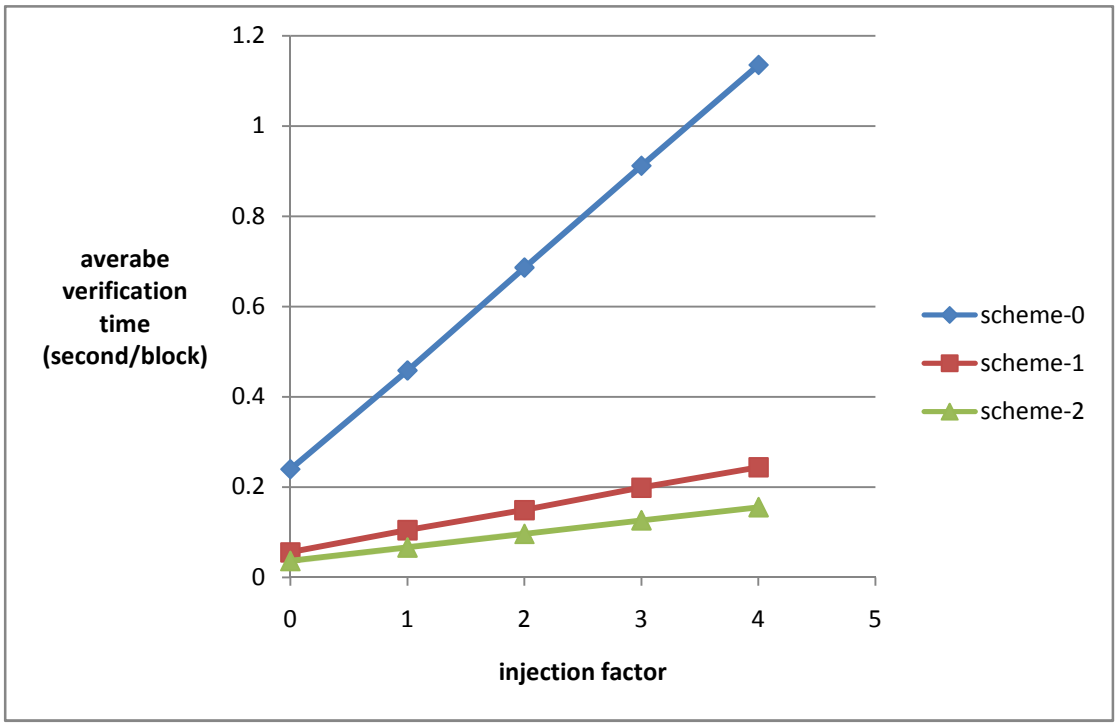

Figure 5-3: Results under the injection attack.

5.4 Comparison

Table 5.1 shows the comparison between Hash chain, Wong-Lam, SAIDA, TESLA and the proposed schemes. We make the following assumptions:

- A block is composed of n messages and one signature is done for each block. For Hash chain and TESLA, n messages consist of a block.
- The computation overhead: We only consider the computational cost at sender side.
- Injection attack resistance: When a scheme is injection attack resistance, this scheme reduces the computational cost of receiver.
- The communication overhead: We consider the average communication overhead for a single packet. For scheme-0, every packet has $\log_2 n$ hash overhead and one signature.

- The storage overhead: We only consider the number of hash values stored at receiver.
- In SAIDA, the encoder is a (t,n)-encoder.

	Hash chain	Scheme-0	Scheme-1	Scheme-2	SAIDA	TESLA
Computation	n-1	2n-1	3n-2	$3n-3+\left(\frac{2n}{\lambda}\right)$	$n+O(n^2)$ field OP	n
Communication	1	$\log_2 n$,1	$1+\log_2 n$,1	$3+\log_2 n$,1	1	3
Individual authentication	No	Yes	Yes	Yes	No	No
Non-repudiation	Yes	Yes	Yes	Yes	Yes	No
Loss resistance	No	Yes	Yes	Yes	Yes	No
Injection attack resistance	Yes	Middle	Yes	Yes	No	No
Storage	1	$\log_2 n$	$1+\log_2 n$	$3+\log_2 n$	t	2

Table 5.1: Comparison of selected schemes.

Chapter 6

Conclusions and future work

We propose two schemes for multicast stream authentication problem. Wong and Lam's scheme has many outstanding features: each packet is individually verifiable and reasonable storage overhead at receiver side. If a multicast stream authentication protocol adopts a digital signature scheme, it is hard for attacker to forge a packet. Besides, the sender only broadcasts packet and doesn't receive any packet. Thus it is highly possible that the attacker may inject many invalid packets to receivers in order to waste the resources of receivers such as computational power and storage. We combine the ideas of Merkle hash tree and hash chain to obtain scheme-1. We extend scheme-1 to scheme-2. Scheme-1 achieves individual packet authentication, lower computational overhead at receiver side and reasonable storage overhead at receiver side with the costs of additional computational cost at sender side for the hash chain. In most cases, the computational power of sender is strong, thus our schemes are reasonable in real world. From the experiment result, scheme-2 is the robustest scheme among three schemes against the injection attack. Scheme-2 is suitable when the packet loss is not low and the data is important for receivers.

We propose an open problem as below. Our schemes achieves individual packet authentication, robust to packet loss, low computational overhead at receiver side and reasonable storage overhead at receiver side. Is it possible to design a scheme that also achieves low communication overhead?

References

[1] Chris Karlof, Naveen Sastry, Yaping Li, Adrian Perrig, and J. D. Tygar. Distillation codes and applications to DoS resistant multicast authentication. In *11th Network and Distributed Systems Security Symposium (NDSS)*, San Diego, USA, February 2004.

[2] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Advances in Cryptology -Crypto'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 180–197, Santa Barbara, USA, August 1997. Springer-Verlag.

[3] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502 – 513, August 1999.

[4] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient multicast packet authentication using signature amortization. In *IEEE Symposium on Security and Privacy*, pages 227–240, Oakland, USA, May 2002. IEEE Press.

[5] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56 – 73, Oakland, USA, May 2000. IEEE Press.

[6] E. N. Gilbert. Capacity of Burst-Noise Channel. In *The Bell System Technical Journal*, pages 1253-1265, September 1960.

[7] E. O. Elliott. Estimates of Error Rates for Codes on Burst-Noise Channels. In *The Bell System Technical Journal*, pages 1977-1997, September 1963.

[8] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*,7(3):277 – 292, June 1999.

[9] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE INFOCOM 1999*, volume 1, pages 345 – 352, New York, USA, March 1999. IEEE Press.

[10] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[11] J. M. Park, E. K. P. Chong, and H. J. Siegel. Efficient multicast packet authentication using erasure codes. In *ACM Transactions on Information and System Security*, pages 6(2):258–285, May 2003.

[12] M. Luby. LT codes. In *43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '02)*, 2002.

[13] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, pages 47(2):569–584, February 2001.

[14] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. In *Journal of ACM*, pages 36(2):335–348, 1989.

[15] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *In Journal of the Society for Industrial and Applied Mathematics*, pages 8(2):300–304, 1960.