RLH: Receiver driven layered hash chaining for multicast data origin authentification

Yacine Challal, Abdelmadjid Bouabdallah, Yoann Hinard

▶ **To cite this version:**

Yacine Challal, Abdelmadjid Bouabdallah, Yoann Hinard. RLH: Receiver driven layered hash chaining for multicast data origin authentification. Computer Communications, 2005, 28 (7), pp.726-740. 10.1016/j.comcom.2004.10.009 . hal-00389982

HAL Id: hal-00389982
https://hal.science/hal-00389982

Submitted on 30 May 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RLH: Receiver driven Layered Hash-chaining for multicast data origin authentication

Yacine Challal , Abdelmadjid Bouabdallah and Yoann Hinard

***Abstract*—Many group-oriented applications, such as broadcasting stock quotes and video-conferencing require data origin authentication of the received traffic. Multicast data origin authentication must take into consideration the scalability and the efficiency of the underlying cryptographic schemes and mechanisms, because multicast groups can be very large and the exchanged data is likely to be heavy in volume (streaming). Besides, multicast data origin authentication must be robust enough against packet loss because most of multicast multimedia applications do not use reliable packet delivery. Therefore, multicast data origin authentication is subject to many concurrent and competitive challenges, when considering these miscellaneous application level requirements and features.**

In this paper, we propose an efficient multicast data origin authentication protocol based on a novel layered hash-chaining scheme. Our protocol tolerates packet loss and guarantees non-repudiation of media-streaming origin. Furthermore, our protocol allows receivers to make the decision regarding the authentication information redundancy degree depending on the quality of reception in term of packet loss ratio. This novel technique allows to save bandwidth since the packet loss distribution over a large scale network is likely to be not uniform. We have simulated our protocol using NS-2, and the simulation results show that the protocol has remarkable features and efficiency compared to other recent data origin authentication protocols.

***Index Terms*—Data origin authentication, Non-repudiation, Layered hash-chaining, Multicast streaming.**

I. Introduction

THE increase of bandwidth in nowadays networks encourages the deployment of multi-party applications, such as videoconferencing, TV over Internet, e-learning and video on demand. Broadcasting information to a group of participants can be achieved using multiple point-to-point transmissions (unicast). This solution is not efficient because of information duplication which induces a high bandwidth consumption. The alternative approach is Multicasting [9] which is an efficient communication mechanism for group-oriented applications. IP multicast saves bandwidth by sending the source traffic on a multicast tree that spans all the members of the group. The lack of security obstructs the large scale deployment of multicast communication applications [19]: *data integrity, secrecy, authentication and access control*. Therefore, securing the multicast communication model is a strategic requirement for effective deployment of large scale business multi-party applications (TV over Internet, Video-on-Demand (VoD), video-conferencing, interactive group games, ...). One of the the main issues in securing multicast communication is the *authentication service*; a keystone of every secure architecture. Even though several authentication mechanisms have existed so far, data origin authentication in multi-party communications remains a challenging problem in terms of scalability, efficiency and performance. Indeed, hashes [21, 41] [11], MACs [23], and digital signatures [42] [37] are the cryptographic answers to integrity, authentication, and non-repudiation in data transmission. However, these mechanisms have been designed typically for point-to-point transmissions, and using them in multicasting yields inefficient and non-adequate solutions. This non-suitability of existing authentication mechanisms is mainly due to the number of group members which may be high in multi-party applications, and to the type of transmitted data which consists generally in continuous streaming of multicast messages with real-time transmission requirement. We distinguish between two types of authentication in group communication [18]:

- **Group authentication**: aims to assure that the received multicast messages by group members originate from a valid group member (no matter its identity).
- **Data origin authentication**: aims to assure that the received multicast messages by group members originate from a source having a specific identity.

In order to assure group authentication, generally group members use a shared key. This key is commonly called: *group key*. Applying a MAC to a message with the *group key* assures that the message originates from a valid group member, since only valid group members are supposed to know the *group key*. Hence, the group authentication problem is reduced to the *group key management* and essentially to its scalability to large groups [8, 18, 19, 39]. In contrast, *multicast data origin authentication* is more complicated because the *group key* which is known by all group members cannot be used to identify a specific sender. We distinguish between two levels of multicast data origin authentication [6]:

A first level guarantees *only* data origin authentication of the multicast source. In this case, a sender needs to use an *asymmetric* mechanism which allows receivers to verify multicast messages authenticity without being able to generate valid authenticators for messages on behalf of the sender. Some solutions [10,45,46] [14] [20,44] [3,5] propose to introduce *asymmetry* in the *key material* used to authenticate messages. In other words, the sender knows the *entire* key material required to authenticate messages, and receivers know only a partial view of the key material, that allows them to verify received messages' authenticity *without being able to generate valid authenticators*. This kind of solutions is subject to *collusions*, where a set

Compiegne University of Technology
Computer Science Department
Heudiasyc lab. France
Phone: +33 (0)3 44 23 44 23
emails: {ychallal,bouabdal,yhinard}@hds.utc.fr

of fraudulent receivers collaborate to reconstruct a part of the whole key material used by the sender, in order to forge authentic messages on its behalf. Other solutions [1, 2] [32, 34, 35] [26, 33, 40] suggest to use *time* as source of *asymmetry*. In other words, receivers are synchronized with the sender's clock and are instructed when to accept a specific key as being used to authenticate received messages. In this case, a fraudulent cannot use a received (or eavesdropped) sender's key to forge messages on behalf of the sender. Indeed, by the time a fraudulent uses a sender's key to forge an authenticator for a message, receivers will reject the fraudulent's message because the used key would have been expired. This approach raises *new security attacks* relating to time synchronization disturbance.

A second level guarantees *non-repudiation in addition to data origin authentication*. In this case, the multicast stream should be signed. Current digital signature mechanisms are *very computationally expensive*. Therefore, it is not practical to sign each packet of the multicast stream. Proposed solutions rely on the concept of *amortizing a single digital signature over multiple packets*. The signature and its amortization induce some *extra-information* called the *authentication information*. Besides, most of multicast media streaming applications do not use reliable transport layer. Hence, some packets may be lost in course of transmission. Therefore, the proposed solutions introduce *redundancy* in the *authentication information*, in a way that even if some packets are lost, the required authentication information can be recovered in order to verify received packets' authenticity. In this case, the *bandwidth overhead*, induced by the redundant authentication information, increases. Proposed solutions deal with how to trade bandwidth for tolerance to packet loss. To tackle these challenges, there exist three main approaches: some protocols [7, 17, 25, 32] amortize a single signature over many packets by chaining these packets using some *hash-chaining* techniques. *Hash-chaining* consists in making each packet carrying hashes that allow the verification of few packets. In turn, these few packets will carry the authentication information of some other packets, and so on .... The overall hash-chaining process culminates into a special packet called *signature packet* which is signed. This signature will then propagate throughout the hash-chain to assure non-repudiation of the chained packets. A second approach [49, 50] [27, 28] [29, 30] consists in signing only a small piece of authentication information (namely hashes of block packets). The resulting authentication information (the signature as well as the original authentication information) is *processed* and *dispersed* among the block packets to be signed. The *processing* is made in a way that even if some packets (that does not exceed a certain threshold) are lost, the received packets can recover the whole authentication information which is required to verify received packets. This approach has the drawback to require *high computation power* to assure the *processing* in both generating and verifying the authentication information. Finally, in another approach [12, 13, 43], instead of signing data itself, the source generates a sequence of (private / public) pairs of keys. Then it signs the public keys which will be, in turn, used to sign data packets using some *fast* signing scheme called: *one-time signing*. One-time signing is known to be very fast with the price that the pair of (private / public) keys can be used to sign only few packets. The essence of this approach is that the slowest phase (signing keys) is made off-line in a way that it does not interfere with the real-time transmission requirement of most of media-streaming applications. Then, each data packet is one-time signed using a beforehand generated and certified (private / public) key (in the off-line phase). The drawback of this approach is that the off-line phase is bounded to produce *certified* private / public keys at a rate which is lower bounded by the rate of data packets arrival at the source. The best solution to this inconvenient is to parallelize the solution and assure the off-line phase using a powerful server.

One problem with existing solutions is that they do not take into consideration the distribution of packet loss throughout a large scale network [51]. Indeed, in existing solutions, the source considers the *worst packet loss ratio* that receivers may encounter in the network and introduces the required authentication information redundancy degree to tolerate this *worst case*. This approach assures a high tolerance to packet loss but introduces extra authentication information overhead since it considers the worst case which is likely to appear only at some parts of the network.

In this paper, we propose an efficient multicast data origin authentication protocol based on a novel layered hash-chaining scheme. We called this protocol: *Receiver driven Layered Hash-chaining for multicast data origin authentication (RLH)*. This protocol tolerates packet loss and guarantees non-repudiation of media-streaming origin. Furthermore, *RLH* allows receivers to make the decision regarding the authentication information redundancy degree depending on the quality of reception in term of packet loss ratio. This novel technique allows to save bandwidth since the packet loss distribution over a large scale network is likely to be not uniform [51]. We have simulated our protocol using NS-2, and the simulation results show that the protocol has remarkable features and efficiency compared to other recent data origin authentication protocols.

In the following section, we recall some useful definitions relating to information security and cryptography, then we present related works that use hash-chaining techniques to amortize signatures over a sequence of packets of the stream. In section IV, we describe our protocol: *RLH*, then we evaluate and compare it with other protocols using NS-2 simulations.

II. Information security and cryptography

In this section, we recall some common information security properties and present an overview of the cryptographic mechanisms used to achieve them.

A. Data integrity

Definition 1: ***Data integrity*** *is the property that data has not been changed, destroyed, or lost in unauthorized or accidental manner [48].*

Cryptographic hash functions are typically used to assure *data integrity* [24].

Definition 2: ***A hash function*** *is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values [24].*

We denote the hash-value of a message m by $h(m)$. Cryptographic *hash functions* have the following properties [22, 24, 47, 48]:

- Given m, it is easy to compute $h(m)$.
- Given $h(m)$, it is hard to compute m such that $h(m) = h$.
- Given m, it is hard to find another message, m', such that $h(m) = h(m')$.

A *hash-value* is also called *message digest*, *hash-result*, or simply: *a hash*.

Example:[1] Suppose that you want to save a large digital document (a program or a database) from alterations that may be caused by viruses or accidental mis-uses. A straighforward solution would be to keep a copy of the digital document on some tamper-proof backing store and periodically compare it to the active version. With a *cryptographic hash function*, you can save storage: you simply save the message digest of the document on the tamper-proof backing store (which because the hash is small could be a piece of paper or a floppy disk) (see figure 1, steps 1 and 2). Then, periodically, you re-calculate the message

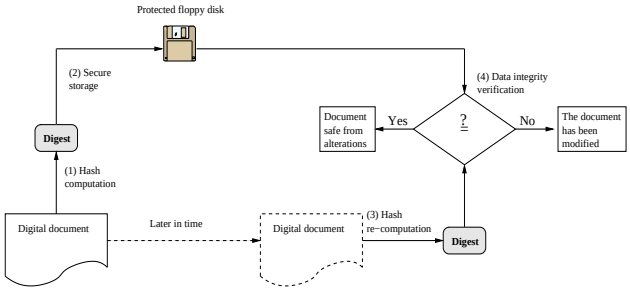

Fig. 1. Assuring data integrity using Message Digests

digest of the document (see figure 1, step 3) and compare it to the original message digest (see figure 1, step 4). If the message digest has not changed, you can be confident none of the data has. Examples of hash functions are: MD2 (Message Digest 2) [21], MD5 [41], SHA-1 (Secure Hash Algorithm 1) [11].

B. Data origin authentication

Definition 3: ***Data origin authentication*** *is the corroboration that the source of data received is as claimed [48].*

[1] This example is cited by many authors such as Kaufman et al. in [22] and Menezes et al. in [24]

Message Authentication Codes (MACs) is a cryptographic mechanism that can be used to assure data origin authentication and data integrity at the same time.

Definition 4: ***A Message Authentication Code (MAC) algorithm*** *is a family of functions h_k parameterized by a secret key k, with the following properties:*

- *Given a key k and an input m, $h_k(m)$ is easy to compute.*
- *h_k maps an input m of an arbitrary finite bitlength to an output $h_k(m)$ of fixed bitlegth.*
  Furthermore, given a description of the function family h, for every fixed allowable value of k (unknown to an adversary), the following property holds:
- *Given zero or more pairs $(m_i, h_k(m_i))$, it is computationally infeasible to compute any pair $(m, h_k(m))$ for any new input m.*

[24]

A MAC can also be seen as *"a cryptographic hash in which the mapping to a hash result is varied by a second input parameter that is a cryptographic key"* [48]. Thus, the point of a MAC is to send something that only someone knowing the secret key can compute and verify. For example, a MAC can be constructed by concatenating a shared secret K_{AB} with the message m, and use $H(m|K_{AB})$ as the MAC (where H is a hash function) [22].

Then, to assure data origin authentication, a sender (A) and a receiver (B) have to share a secret key K_{AB}. Then the sender computes the digest $(MAC(K_{AB}, m))$ corresponding to the message (m), to be sent, using the secret key (K_{AB})(see figure 2 step 1). Upon receiving the message as well as the digest, the receiver verifies the origin of the received message as follows: it recalculates the digest of the received message using the secret key K_{AB} (fig. 1 step 3) and compares it to the received digest (fig. 1 step 4). If the two digests are equal, the message is said to be authentic (has not been altered) and originates from the sender (A) since only (A) and (B) know the secret K_{AB}. Otherwise, the received message has been altered or fabricated by a sender who is not (A). An example of MAC is:

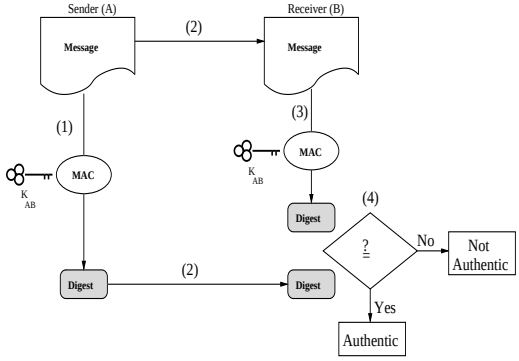

Fig. 2. Assuring data origin authentication using MACs

HMAC [23].

C. Data confidentiality

Definition 5: ***Data confidentiality*** *is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes [48].*

Confidentiality is guaranteed using *encryption.*

Definition 6: ***Encryption*** *is a cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used. If the transformation is reversible, the corresponding reversal process is called* decryption*, which is a transformation that restores encrypted data to its original state [48].*

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic encryption algorithm, which is widely known, but on a piece of information called a *key* that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Depending on whether the same or different keys are used to encrypt and to decrypt the information, we distinguish between two types of encryption systems used to assure confidentiality:

C.1 Symmetric-key Encryption

In a symmetric-key encryption system, a secret key is shared between the sender and the receiver and it is used to encrypt the message by the sender and to decrypt it by the receiver. The encryption of the message produces a non-intelligible piece of information and the decryption reproduces the original message (see figure 3). Examples

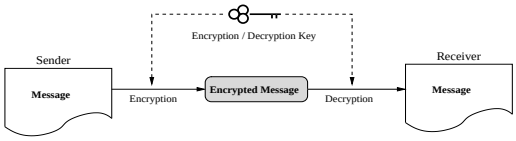

Fig. 3. Assuring confidentiality using symmetric-key encryption

of symmetric-key encryption systems are: DES [36], AES [38], IDEA [22].

C.2 Public-key Encryption

Public-key encryption (also called asymmetric encryption) involves a pair of keys (a *public key* and a *private key*) associated with the sender. Each *public key* is published, and the corresponding *private key* is kept secret by the sender. Data encrypted with the sender's *public key* can be decrypted only with the sender's *private key* (see figure 4).

Fig. 4. Assuring confidentiality using asymmetric-key encryption

In general, to send encrypted data to someone, the sender encrypts the data with that receiver's public key, and the receiver of the encrypted data decrypts it with the corresponding private key. Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, it is possible to use public-key encryption to send a symmetric key, which can then be used to encrypt additional data. An example of asymmetric encryption systems is: RSA [42].

D. Non-repudiation with proof of origin

Definition 7: ***Non-repudiation with proof of origin*** *provides the recipient of data with evidence that proves the origin of the data, and thus protects the recipient against an attempt by the originator to falsely deny sending the data [48].*

Note that a MAC cannot be used as a proof (to a third party) that a message originates from a specific entity. In fact, let us consider that a sender A and a receiver B share a secret key K_{AB}. If A denies having sent a message m, the receiver B cannot use the received MAC of m as a proof of m's origin, because A would then say that B might have created the m's MAC himself!. Thus, asymmetric cryptography is the basic answer for non-repudiation. With asymmetric cryptography, the piece of information sent with the message as a proof of integrity and data origin is computed using a private key held only by the sender and is verified by the receiver using the public key that corresponds to the private key. Hence, since only the sender can compute the piece of information, this latter can be used as a proof of origin to a third party and hence non-repudiation is assured. This cryptographic mechanism is called *Digital Signature.*

To *sign* a message, a sender generates a pair of private/public keys using some asymmetric cryptographic system. The sender keeps the private key secret and publishes the public key. Then the sender calculates the digest of the message to be sent using any hash function (see figure 5 step 1). The digest is then cryptographically transformed using the private key (fig. 5 step 2). The result of this transformation is called: the *digital signature* of the message. Upon receiving the message and the signature, the receiver verifies the signature using the *public key* as follows: first, the receiver recalculates the digest of the received message (fig. 5 step 4). Then, the receiver verifies the received signature using the public key (fig. 5 step 5). If the signature is valid then the message as well as its origin are authentic and non-repudiation is guaranteed. Otherwise the message is rejected. Examples of digital signature schemes are: RSA [42], DSA [37].

D.1 Certification

To verify a signature, a receiver needs to be assured that the public key used in verifying a signature corresponds to the private key of the real sender of the signed message and not generated by an intruder who tries to impersonate the real sender. The electronic document that assures this

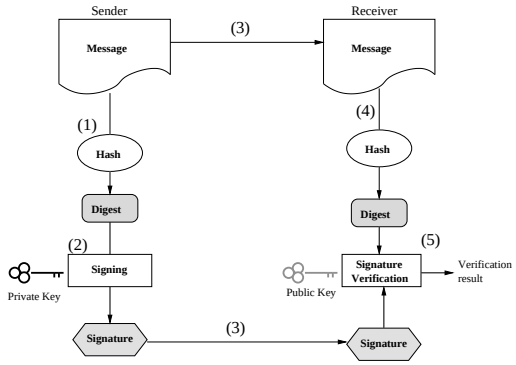

Fig. 5. Assuring non-repudiation using Digital Signatures

matching is called: *public-key certificate*.

Definition 8: ***A public-key certificate*** *is a digital certificate that binds a system entity's identity to a public key value, and possibly to additional data items; a digitally-signed data structure that attests to the ownership of a public key [48].*

Definition 9: ***Certification*** *is the process of vouching for the ownership of a public key by issuing a public-key certificate that binds the key to the name of the entity that possesses the matching private key [48].*

A *certificate* is digitally signed by a *Certification Authority (CA)* which is *trusted* by receivers and whose public key is known by receivers in a secure way. Thus, to publish a public key, a sender should issue a signed certificate of its public key to receivers. The enclosed public key is used, then, by receivers to verify the digital signatures generated by the sender whose identity is also enclosed in the same certificate.

In the rest of the paper, we suppose that the public keys used to verify digital signatures are certified.

III. Related works

In this section, we will present some protocols that use *signature amortization* relying on hash-chaining techniques.

A. Simple off-line hash-chaining

The main idea of the solution proposed by Gennaro and Rohatgi in [15, 16] is to divide the stream into blocks and embed in the current block a hash of the following block (which in turn includes the hash of the following one and so on...)(see figure 6). This way the signer needs to sign only the first block and then the properties of this single signature will *propagate* to the rest of the stream through the hash-chaining. We note that in order to construct this chain, the sender needs to know the entire stream in advance (off-line). With this solution, the authentication information is reduced to one hash per block and the sender signs only the hash of the first block. However, this solution is not fault tolerant: if a block is lost, the authentication chain is broken and hence all subsequent blocks can no longer be authenticated.

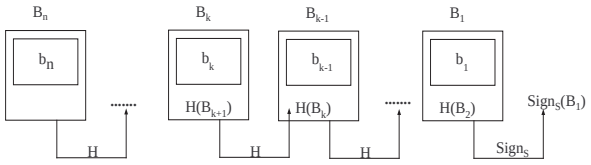

Fig. 6. Simple off-line hash-chaining (Example)

B. Random hash-chaining

Perrig et al. [32] proposed the *Efficient Multi-chained Stream Signature* protocol (EMSS). This protocol introduced the notion of *redundant hash-chaining* which means that each packet's hash of the stream is embedded in several *subsequent* packets. Then a final packet (which is called the signature packet) containing several hashes of previous packets is signed. Therefore, each packet has many hash-chains to the signature packet. Thus, even if some packets are lost, a received packet is verifiable if it remains a hash-chain that relates the packet to the signature packet. For a given packet, EMSS embeds its hash into k subsequent randomly chosen packets (k is called the redundancy degree). Hence, EMSS provides more or less probabilistic guarantees that it remains a hash-chain between the packet and a signature packet, given a certain rate of packet loss in the network. The robustness of the protocol to packet loss is proportional to the *redundancy degree*: k. In order for the sender to continuously assure the authentication of the stream, the sender sends periodic signature packets. To verify authenticity of received packets, a receiver buffers received packets and waits for their corresponding signature packet. The signature packet carries the hashes that allow the verification of few packets. These latter packets carry, in turn, the hashes that allow to verify other packets, and so on until the authenticity of all received packets is verified.

Challal et al. [7] proposed the A^2Cast protocol. A^2Cast uses a technique similar to EMSS, but the authentication information redundancy degree is *source driven* rather than fixed a priori. In other words, the source determines, periodically, the required redundancy degree depending on the *average packet loss ratio* which is calculated using receivers' feedbacks. Simulations showed that this technique allows to save *authentication information bandwidth overhead*.

Minner and Staddon [25] proposed a redundant and random hash-chaining scheme to tolerate packet loss in a network where each packet is lost independently at random with probability q. Authors were interested in applications in which the sender has a priori knowledge of the content. Therefore, the *hash-link topology* is constructed before the first packet of the stream is sent. The *random redundant topology* proposed by the authors is called *p-random graph*. In a basic *p-random graph* scheme, packets of the stream are numbered from 1 to n. P_1 is the *signature packet*, and for all pairs of packets (P_i, P_j) where $j < i$, the hash of

packet P_i is embeded whithin packet P_j whith probability p. Once the p-random graph of the stream is contructed, the packets of the stream are sent respectively. A receiver starts by receiving the *signature packet*. If it is valid, the receiver verifies subsequent packets *on the fly* by checking the existence of a *hash-link path* between the received packt and the *signature packet*.

C. Deterministic hash-chaining

Modadugu and Golle [17] have proposed to use a similar strategy to EMSS, but packets that will carry the hash of a given packet are chosen in a *deterministic* way rather than randomly. The authors proposed *deterministic topologies* of packet hash-chains, called *Augmented Chains*. *Augmented chains* are designed to be optimized to resist a *burst loss*. The goal of the proposed schemes is to maximize the size of the longest single burst of loss that the authentication scheme can withstand (Once few packets have been received after a burst, the scheme recovers and is ready to maintain authentication even if further loss occurs).

Miner and Staddon [25], proposed a similar authentication scheme, based on hash chaining techniques, specifically designed to resist multiple bursts. The proposed scheme, called *Piggybacking*, deals with the case where data carried by different packets has more or less importance from the point of view of the application level. Thus, packets are organized into classes with different priorities. Then hash chaining is made in a way that: the higher is the priority of a class, the more redundant is hash-chaining of packets belonging to that class, in order to resist better against bursty losses.

In what follows, we present our protocol which uses the concept of amortizing a single digital signature over multiple packets using hash-chaining, then we present simulation results that evaluate and compare the performance of RLH to another protocol.

IV. *RLH: Receiver driven Layered Hash-chaining for multicast data origin authentication*

A. Terminology

We define some terminology to simplify the following discussion: if a packet P_j contains the hash of a packet P_i, we say that a **hash-link** connects P_i to P_j, and we call P_j a **target** packet of P_i. A **signature packet** is a sequence of packet hashes which are signed using a conventional digital signature scheme. A hash-link relates a packet P_k to a signature packet S_l, if S_l contains the hash of P_k. We designate by **redundancy degree** the number of times that a packet hash is embedded in subsequent packets to create redundancy in chaining the packet to a signature packet. A packet P_i is **verifiable**, if it remains a **path** (following the hash-links) from P_i to a signature packet S_j (even if some packets are lost). We designate by **verification ratio**: the number of verifiable packets by the number of received packets. The verification ratio is a good indicator of the **verification probability** which means the probability for a packet to be verifiable given that it is received: P(packet is verifiable|packet is received). This probability is equal to the probability that it remains a **hash-link path** (a hash-chain) that relates the packet to a signature packet after removing the lost packets.

B. Overview and motivation

To achieve non-repudiation, we rely on a conventional signature scheme for example RSA [42]. Unfortunately, the computation and communication overhead of current signature schemes is too high to sign every packet individually. To reduce the overhead, one signature needs to be amortized over multiple packets. The amortization is achieved using *hash-chaining*, which consists in signing a single packet and amortizing this single signature over multiple packets by *hash-linking* the current packet to another packet in the stream. In paragraph III.A we discussed a basic chaining scheme. In our protocol, we use a *redundant hash-chaining* scheme to tolerate packet loss. The *redundant hash-chaining* that we propose is organized into different layers of redundancy. A basic layer carries the payload data packets in conjunction with a *minimal hash-chaining redundancy degree*. This layer is *vertically* chained to factual layers with different amounts of redundant hash-chains. Each layer is sent to a different multicast group and assures robustness to a certain amount of packet loss. Periodically, receivers calculate the actual packet loss ratio and use it to decide whether to join a corresponding extra-layer in order to improve the *verification probability*. Figure 7 illustrates a scenario where the source produces three layers of authentication information. $L0$ is the compulsory basic layer that carries the payload data packets. $L1$ and $L2$ are authentication information layers that receivers can join to improve the *verification probability*. In this simple scenario, we consider that $L2$ is more redundant than $L1$, and hence $L2$ is joined only by those receivers that encounter a sever packet loss rate in their subnet.

Since the packet loss distribution over a large scale network is likely to be not uniform [51], this *receiver driven* technique will allow to save bandwidth. Indeed, with this technique, each receiver receives only the required authentication information that allows him to face the actual packet loss ratio in its subnet. In the following paragraphs, we will describe our *layered hash-chaining* scheme. Then we present the *Receiver driven Layered Hash-chaining for multicast data origin authentication* protocol (RLH).

C. Layered Hash-chaining scheme

The basic idea of hash-chaining is that each packet carries the hash code of the previous packet. A final packet (the signature packet) is signed and guarantees data origin authentication and non-repudiation of the chained packets [16]. In order to tolerate packet loss, we make *redundant hash-chaining*: instead of carrying a single hash of the previous packet, each packet carries the hashes of multiple packets, so that even if some packets are lost, there is a *probability* that it remains *hash-link* paths between received packets and the signature packet. If a *hash-link*

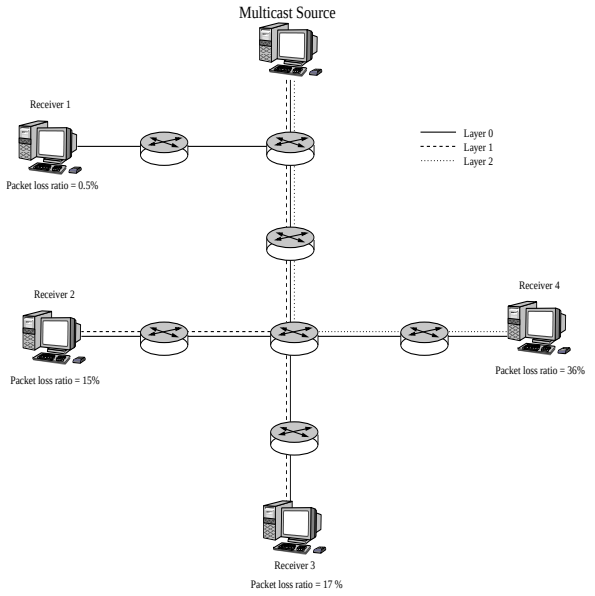

Fig. 7. A simple RLH scenario with three layers

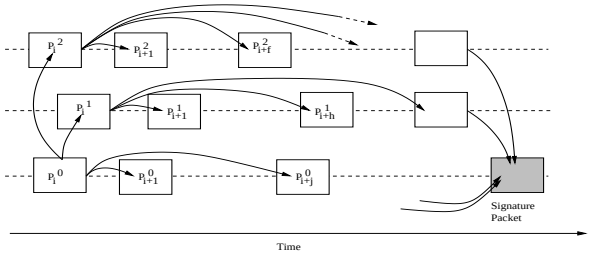

Fig. 8. Layered Hash-chaining

(n)	Number of layers
(h_i, v_i)	The horizontal and vertical redundancy degrees of layer i
(f)	Number of packets after which a signature packet is sent
(d)	The scope within which packets are chosen randomly to embed the hash of the current packet
(t)	The period of time after which receivers analyse packet loss ratio to decide whether to join a new authentication layer

TABLE I
RLH PARAMETERS

path exists between a received packet and the signature packet, then the authenticity of the received packet is *verifiable* [16, 32]. In our case, we have different layers of redundant hash-chains. The first layer is the basic data payload layer. It carries data packets chained using a redundant hash-chaining with a *small* redundancy degree. These packets are also chained to other factual layers. Packets of these layers are only hash-chained using different redundancy degrees and hence carry only hashes of packets from the same layer or from the basic layer. It turns out that each layer i ($i = 0$ for the basic layer and $i = n$ for the last one) is caracterized by two redundancy degrees:

- The *horizontal redundancy degree* h_i: determines the number of times the hash of a packet is embedded into subsequent packets from the same layer i.
- The *vertical redundancy degree* v_i: determines the number of times the hash of a packet from layer 0 is embedded into packets from layer i.

Figure 8 illustrates an example of layered hash-chaining with three layers: the basic layer has horizontal and vertical degrees respectively equal to 2 and 0. Layer 1 has horizontal and vertical degrees respectively equal to 3 and 1, and layer 2 has horizontal and vertical degrees respectively equal to 4 and 1.

When a data packet is presented to be sent at the sender, it is *hash-linked* following two steps:

(a) Horizontal hash-chaining: in this step, the hash of the current data packet is embedded into h_0 subsequent *target packets*: one packet is the next one, and $h_0 - 1$ target packets are chosen randomly. Similarly, authentication packets that are beyond the current packet in the other layers are also horizontally chained to h_i subsequent target packets, where i is the layer number. One target packet is the next one and the other $h_i - 1$ target packets are chosen randomly.

(b) Vertical hash-chaining: in this step, the hash of the current data packet is embedded within v_i target packets (for each layer i): one packet is the packet that has the same sequence number in layer i and $v_i - 1$ target packets are chosen *randomly*.

D. RLH protocol

We consider a multicast source of a stream which consists in a sequence of data packets. The source constructs the different authentication layers according to the layered hash-chaining scheme described above. The source sends each layer i to a different multicast group g_i. In order to assure continuous non-repudiation of the stream, the source sends periodically a *signature packet*. This signature is calculated over the concatenation of the following hashes:

- The hashes of packets from layer 0 for which the signature packet is a *target packet*.
- For each authentication layer i ($i \neq 0$), the hash of the last sent authentication packet.

Receivers of the stream join the group g_0 and start verifying the authenticity of received packets relying on the basic redundant hash-chaining of layer 0. Continuously, receivers report lost packets using time outs and sequence numbers of received packets. Periodically, each receiver uses the packet loss ratio, calculated during the last period of time, to decide whether to join another layer in addition to the basic layer in order to improve the *verification probability*. Indeed, each new layer brings new hash-chains in addition to hash-chains of layer 0, and hence increases the probability that a hash-chain remains between each data packet and a signature packet even if some packets are lost.

Table I summarizes the parameters involved in *RLH* protocol. These parameters influence the computation and communication overhead, the delay until verification, and

the robustness against packet loss. We want to achieve low overhead while retaining high robustness against packet loss.

D.1 The Sender Side Algorithm

In what follows, we denote a packet with a sequence number i and belonging to layer k by P_i^k. A source of a stream applies the layered hash-chaining scheme described above for each packet P_i^0 before it sends it. Packets of layer k are sent to the corresponding multicast group g_k. After each f data packets, the source sends a signature packet. We suppose that signature packets are sent using a certain reliable mean. The algorithm at the source would then be as shown in figure 9.

```
for each packet P_i^0 do
    for each layer k do
        /* make horizontal hash-chaining */
        embed H(P_i^k) = h_i^k in the packet P_{i+1}^k;
        do h_k − 1 times
            generate a random number j so that j ∈ [i + 2, i + d];
            include H(P_i^k) = h_i^k in the packet P_{i+j}^k;
        end;

        /* make vertical hash-chaining */
        embed H(P_i^0) = h_i^0 in the packet P_i^k;
        do v_k − 1 times
            generate a random number j so that j ∈ [i + 1, i + d];
            include H(P_i^0) = h_i^0 in the packet P_{i+j}^k;
        end;
        send packet P_i^k to multicast group g_k;
    end;
end.

after each f packets do
    sign the current packet S_l;
    send the signature packet S_l to multicast group g_0;
end.
```

Fig. 9. The algorithm at the source side

```
do
    receive packet P_i^k.
    if P_i^k is not a signature packet then
        buffer P_i^k;
        buffer hashes h_j^m included in P_i^k;
    else
        /* P_i^k is a signature packet */
        verify(P_i^k);
    end;
while(true).

upon timeout do
    update_membership(packet_loss_ratio);
    schedule timeout after t seconds;
end.
```

Fig. 10. The algorithm at a receiver side

```
verify(P_i^k)
if P_i^k is a signature packet then
    verify the signature of P_i^k;
    if the P_i^k's signature is valid then
        P_i^k is authentic;
        for each hash h_j^m included in P_i^k do
            verify(P_j^m);
        end;
    else
        P_i^k is not authentic;
    end;
else
    /* verify P_i^k against its buffered hash code h_i^k */
    if H(P_i^k) = h_i^k then
        P_i^k is authentic;
        for each hash h_j^m included in P_i^k do
            verify(P_j^m);
        end;
    else
        P_i^k is not authentic;
    end;
end.
```

Fig. 11. The recursive verification procedure

D.2 The Receiver Side Algorithm

When a receiver receives a signature packet S_l, it verifies the signature of S_l and verifies the authenticity of all the packets that have a path to S_l. After each t seconds, the receiver analyses the packet loss ratio and decides whether to join another layer to increase *verification probability* of received packets. This decision is made using a function that we call *update_membership* for which it gives the packet loss ratio as a parameter. The algorithm at the receiver side is shown in figure 10, and the verification procedure is illustrated in figure 11.

V. Simulations and performance evaluation

We carried out simulations using NS-2 to evaluate the performance of *RLH* and compare it with EMSS [32].

A. The bursty packet loss model

We used the two state Markov chain model [52] to extend NS-2 with a new queuing behavior to simulate a bursty packet loss pattern. Indeed, many studies show that packet loss is correlated, which means that the probability of loss is much higher if the previous packet is lost. Paxson shows in [31] that packet loss is correlated and that the length of losses exhibit infinite variance. Borella et al. found that the average length of loss bursts is about 7 packets [4]. Yanik et al. show that a k-state Markov chain can model Internet packet loss patterns [52]. For our simulation purposes, the two-state Markov chain model is sufficient, since it can correctly model simple patterns of bursty packet loss [52]. Figure 12 shows the two-state Markov chain used in our simulations and whose transition probabilities can easily be determined using the average burst length and the packet loss ratio in the network.

B. Simulation parameters

In what follows, we consider a bursty packet loss pattern with bursts having an average length equal to 7. Then, we considered a stream of $10,000$ packets with a signature packet every 500 packets ($f = 500$), and where a packet is *hash-linked* to packets within the scope of 250 packets ($d = 250$). The value of f has been arbitrary chosen. In reality, the value of f should be chosen depending on the application level tolerance to latencies, the computation

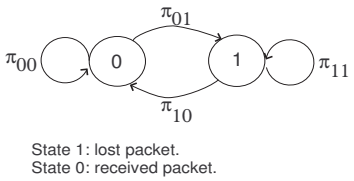

Fig. 12. Two-state Markov chain to simulate bursty packet loss

power of communicating parties and the available bandwidth. The general rule is: if the parameter f is long, then receivers will experience important latencies before verification but will not have too much signatures to verify, and the reduced number of signatures will not consume a lot of bandwidth. Receivers analyse packet loss ratio and eventually update their membership to authentication layers every 30 seconds ($t = 30s$).

C. Updating the membership to authentication layers

Recall that periodically, the receivers analyse the actual packet loss ratio in their subnets. Then use this ratio to join and / or leave authentication layers in order to increase the *verification probability*. This decision is made using the *update_membership* function. To develop this function, we simulated different combinations of different layers with different horizontal and vertical redundancy degrees. At last, we selected the combination of three layers whose verification ratios are illustrated in figure 13.

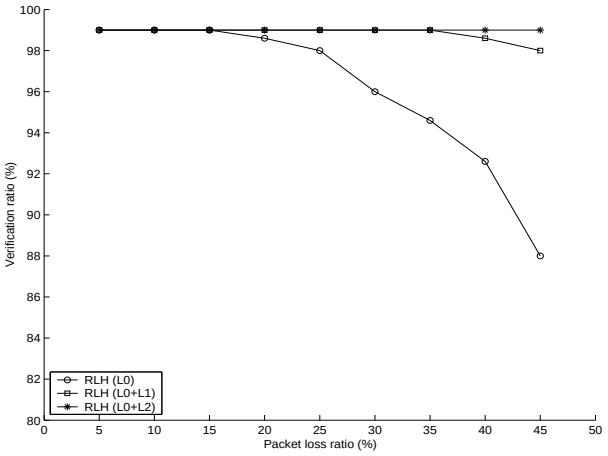

Fig. 13. The verification ratio of different hash-chain layer combinations

Table II illustrates the vertical and horizontal redundancy degrees of the selected combination layers.

We notice that for packet loss ratios that varies from 5% to 15%, the basic layer suffices to reach 99% of *verification ratio*. The basic layer in addition to layer 1 assure 99% of verification ratio while tolerating up to 35% of packet loss.

Layers	Vertical degree	Horizontal degree	Total degree
L0	0	2	2
L0+L1	1	3	4
L0+L2	1	5	6

TABLE II
PARAMETER VALUES OF THE SELECTED COMBINATION OF LAYERS

Finally, the combination of the basic layer with layer 2 assures 99% of verification ratio while tolerating up to 45%. Thus, when a receiver calculates the encountered packet loss ratio in its subnet, it calls the *update_membership* function depicted in figure 14. Without loss in generality, we

```
function update_membership(loss_ratio)
    join basic layer;
    if 15 <loss_ratio≤ 35 then
        join layer 1;
    if 35 <loss_ratio then
        join layer 2;
end.
```

Fig. 14. The *update_membership* function

suppose that the maximum packet loss ratio is 45%

D. Simulation Results

In order to illustrate the behavior of RLH compared to EMSS when considering a large scale network, where the packet loss ratio is likely to be not uniform [51], we considered a network with three different areas. Figure 15 illustrates this simplified scenario. Each area is caracterized by its own packet loss ratio. Namely, the three areas have respectively 5%, 25% and 45% packet loss ratios [2].

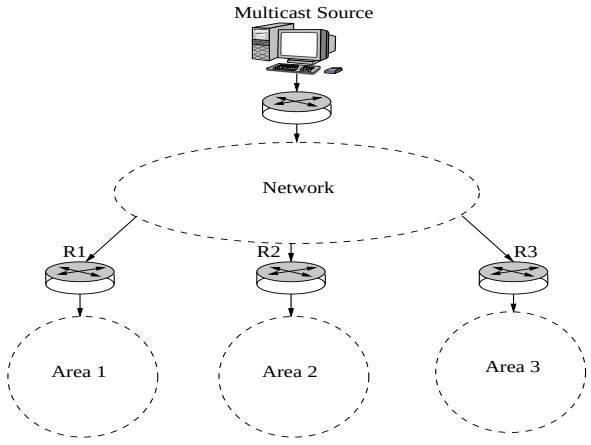

Fig. 15. Simulation scenario

We want to reach a very high verification ratio (99%). With RLH, each receiver in each area joins the required *hash-chain* layers to reach 99% of verification ratio using the *update_membership* function. In contrast, with EMSS,

[2] These values have been chosen to demonstrate the extent of our protocol robustness to packet loss

receivers are not able to choose the best redundancy degree. Figure 16 illustrates the required EMSS redundancy degree to reach 99% of verification ratio when we vary the packet loss ratio from 5% to 60%. Therefore, the multicast

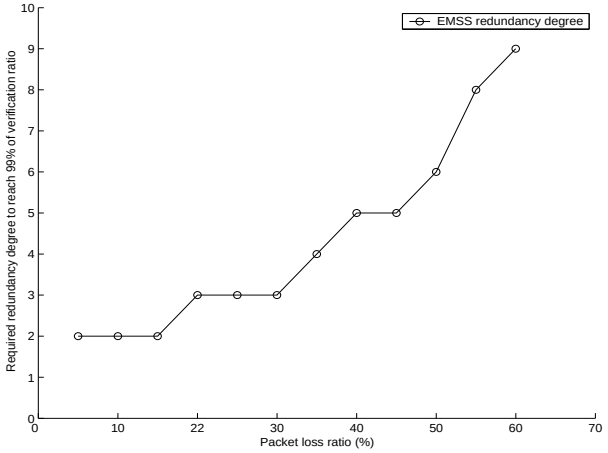

Fig. 16. The required redundancy degree to reach 99% of verification ratio

source has to choose the best redundancy degree so that receivers can verify the authenticity of received packets with a probability equal at least to 99%. Three strategies can be envisioned:

- *Considering the minimal packet loss ratio*: in this technique, the source considers only the area that experiences the minimal packet loss ratio. In this case the source uses the degree 2 which corresponds to the required degree to tolerate 5% of packet loss (see figure 16). This technique allows to save bandwidth but receivers in the other areas will not reach the 99% verification ratio.
- *Considering the maximal packet loss ratio*: in this technique, the source considers only the area that experiences the maximal packet loss ratio. In this case the source uses the degree 5 which corresponds to the required degree to tolerate 45% of packet loss (see figure 16). This technique assures that all receivers in the different areas reach the desired 99% verification ratio, but receivers in areas 1 and 2 will waste bandwidth to receive useless authentication information (extra-redundancy).
- *Considering the average packet loss ratio*: in this technique, the source considers average packet loss ratio. In this case the source uses the degree 3 which corresponds to the required degree to tolerate 25% of packet loss which is the average packet loss of the three areas (see figure 16). With this technique, some receivers may not reach the desired verification ratio.

Figure 17 illustrates the verification ratio reached within each area using these three different strategies. Notice that none of them achieves the best trade-off between *authentication information bandwidth overhead* and *verification ratio*.

However, in the case of RLH, receivers in area 1 join only

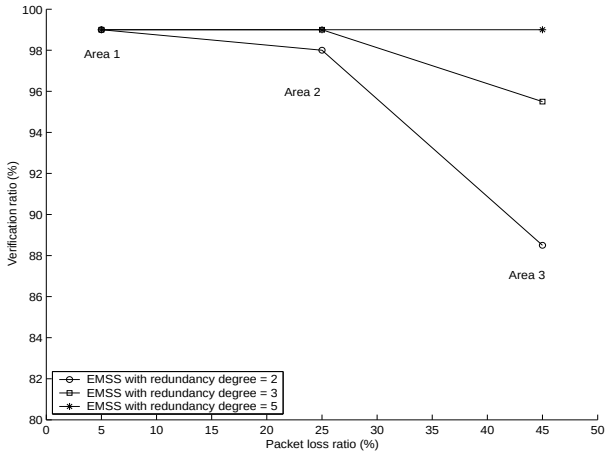

Fig. 17. The verification ratio within the three areas when considering the three different strategies

the basic layer which suffices to reach the target verification ratio. Receivers of area 2 join the basic layer in addition to layer 1, and receivers of area 3 join the basic layer in addition to layer 2. This way, RLH allows receivers of different areas to save useless bandwidth and to *request* the only required redundancy degree to face the packet loss that is encountered in their respective areas. Figure 18 compares RLH to EMSS regarding the authentication information overhead which consists in the embedded hash codes that are used to construct the redundant hash-chains. To make this comparison, we calculated the number of hash codes (the authentication information overhead) that pass through the *on-tree* multicast border routers of each area: R1, R2 and R3 (see figure 15).

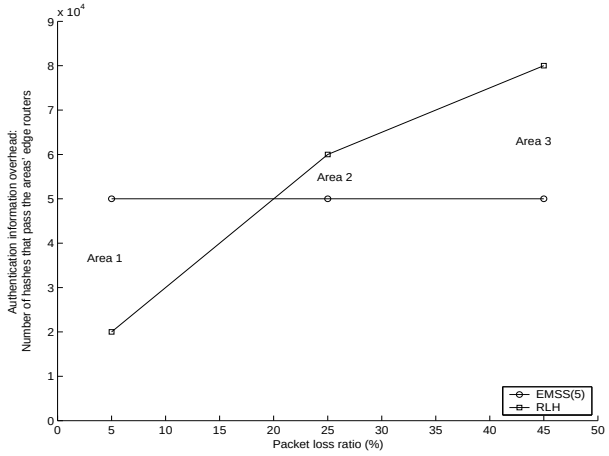

Fig. 18. The authentication information overhead in the different areas

We notice in figure 18, that with EMSS the three areas receive exactly the same amount of authentication information [3], even if each area experiences a different amount

[3] We considered the maximal packet loss ratio strategy so that all receivers reach the target verification ratio

of packet loss ratio. In contrast, with RLH, each area receives a different amount of authentication information (different layers) depending on the encountered packet loss ratio. Figure 19 illustrates the repartition of the *authentication information overhead* per area due to each layer. As expected, receivers of area 1 receive only layer 0 packets. Receivers in area 2 receive layer 0 and layer 1 packets, and receivers in area 3 receive layer 0 and layer 2 packets. This is due to the fact that receivers in each area join only the required layers to reach the target *verification ratio*.

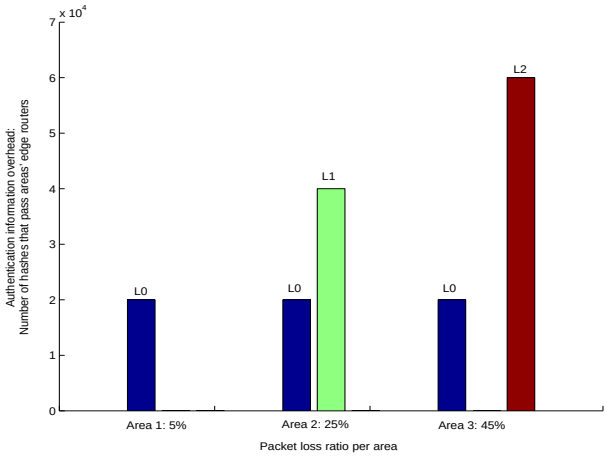

Fig. 19. The authentication information overhead induced by each layer in the different areas

To further illustrate how RLH allows to save bandwidth, let us consider the second scenario depicted in figure 20. The multicast source streams the three RLH layers: the basic data payload layer (layer L0), the medium redundant authentication layer (layer L1), and the highly redundant authentication layer (layer L2). The dashed lines determine the three areas with the different packet loss ratios. The area 3 with 45% packet loss ratio is introduced to demonstrate the extent of RLH robustness to packet loss and its adaptability to packet loss variation. We were interested in measuring the *tree authentication information cost*, which we define as follows:

Definition 10: The tree authentication information cost is the number of *hash codes*, sent over a multicast tree, by the *size of the multicast tree*. We mean by the *size of a multicast tree* the number of network links that constitute the multicast tree. Thus the *tree authentication information cost* measures the *total authentication information bandwidth overhead*.

In our simulation, we used the NS2 implementation of PIM-SM protocol, with RP as a Rendez-vous Point node (see figure 20). In this scenario we considered a $5,000$ packet stream. Figure 21 illustrates the *tree authentication information cost* induced by RLH compared to the one induced by EMSS.

With RLH, to each layer corresponds a *tree authentication information cost*: L0 spans all the receivers in the three areas with a *redundancy degree* equal to 2 hashes per packet. L1 spans only receivers of area 2 with a *redundancy degree* equal to 4 hashes per packet, and finally L2 spans only receivers of area 3 with a *redundancy degree* equal to 6 hashes per packet. The three layers induce *tree authentication information costs*, respectively equal to: $104,500$, $65,000$ and $46,500$ hash codes in average. In contrast, with EMSS there is a single tree that spans all the receivers in the three areas with a *redundancy degree* equal to 5, and hence induces a *tree authentication information cost* equal to $261,250$ hash codes, in average. According to the results depicted in figure 21, we notice that the overall RLH *tree authentication information cost* (sum of the three layer costs) is roughly $50,000$ hash codes less than the cost induced by EMSS. If we consider a 160 bit hash codes (such as SHA-1), RLH would then save up to 1 MBytes of *tree authentication information*. This is due to the fact that with EMSS, the source considers the maximum redundancy de-

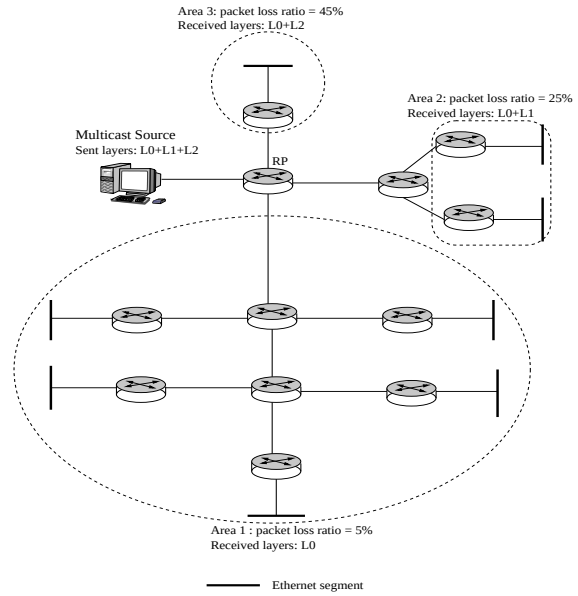

Fig. 20. Simulation scenario with not uniform packet loss distribution

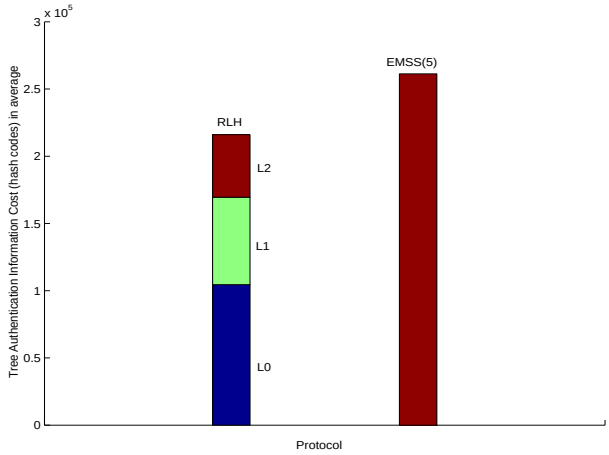

Fig. 21. Tree authentication information cost

gree so that all receivers reach the same target verification ratio. Whereas, with RLH, receivers join only the required authentication layers to reach the target verification ratio. Therefore, RLH allows receivers to adapt the redundancy degree depending on the actual encountered packet loss ratio.

In conclusion, RLH efficiency increases when the multicast tree size is important and the packet loss phenomenon is concentrated in dense areas.

E. RLH security and other performance criteria

RLH guarantees data origin authentication and non-repudiation by relying on the existence of *hash-chains* between data packets and *signature packets*. Hence, the security of our protocol (RLH) relies on the security of this basic technique (hash-chains) which has been proved to be secure by Gennaro and Rohatgi [16]. We have shown in previous sections that RLH reduces the amount of *tree authentication information* while maintaining good performance in term of *robustness against packet loss*. Furthermore, we summarize some other features of RLH in what follows:

- *Storage requirement and delay before verification at receivers*: with RLH, a receiver experiences a delay before verification of received packets, because it has to receive the signature packet which corresponds to received packets in order to launch the verification process. Hence, receivers need to buffer received packets until the reception of the corresponding signature packet. The duration of the delay and the size of the buffer depend on the period (f packets) after which signature packets are sent.
- *Storage requirements and delay before authentication at the source*: with RLH, the source authenticates the packets and signs the stream on the fly. Hence the multicast source does not experience any delay before authenticating the stream packets.
- *Scalability*: since the hash-chaining technique used by RLH is independent from the number of receivers, the protocol scales to large groups.

F. Comparison

The efficiency of a data origin authentication protocol with non-repudiation can be measured according to many criteria. Table III compares some data origin authentication with non-repudiation protocols, described in the related works section, with respect to the following criteria [4]:

1. *The latency at the sender*: corresponds to the fact that the sender needs to buffer packets before sending them.
2. *The latency at a receiver*: corresponds to the fact that a receiver needs to buffer packets before verifying their authenticity.
3. *Tolerance to packet loss*: corresponds to the fact that the authentication process is possible even if some packets are lost.
4. *Authentication information size*: the size of the authentication information embedded to a packet.

[4] With EMSS, we consider results of the special case simulated by authors

VI. Conclusion

Data origin authentication is an important component in the whole multicast security architecture. Besides, many applications need non-repudiation of data-streams. To achieve non-repudiation, we proposed a new efficient protocol called *RLH*. Our protocol uses a layered hash-chaining technique to amortize a single digital signature over many packets. This *RLH*'s hash-chaining technique allows receivers to limit the authentication information bandwidth overhead to only the required overhead that allows to reach a given packet *verification ratio*. Simulation results using NS-2 show that our protocol resists to bursty packet loss and assures with a high probability that a received packet be verifiable. Besides, the simulations and comparisons with another protocol show that our layered hash-chaining technique allows to save bandwidth since the packet loss phenomenon is likely to be not uniform over a large scale network.

References

[1] F. Bergadano, D. Cavagnino, and B. Crispo. Individual Single Source Authentication on the MBone. *IEEE International Conference on Multimedia and Expo*, 2000.

[2] F. Bergadano, D. Cavagnino, and B. Crispo. Individual Authentication in Multiparty Communications. *Computers and Security*, 21(8):719–735, 2002.

[3] Dan Boneh, Glenn Durfee, and Matt Franklin. Lower Bounds for Multicast Message Authentication. *Eurocrypt'01*, LNCS(2045):437–452, 2001.

[4] M. Borella, D. Swider, S. Uludag, and G. Brewster. Internet packet loss: Measurement and implications for end-to-end qos. *International Conference on Parallel Processing*, August 1998.

[5] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast Security: A taxonomy and Efficient Constructions. *INFOCOM*, 1999.

[6] Y. Challal, H. Bettahar, and A. Bouabdallah. A Taxonomy of Multicast Data Origin Authentication: Issues and Solutions. *IEEE Communications Surveys and Tutorials*, 6(3), July 2004.

[7] Y. Challal, H. Bettahar, and A. Bouabdallah. *A^2Cast*: an Adaptive source Authentication protocol for multiCast streams. *IEEE-ISCC'2004*, June 2004.

[8] Y. Challal, H. Bettahar, and A. Bouabdallah. SAKM: A Scalable and Adaptive Key Management Approach for Multicast Communications. *ACM SIGCOMM Computer Communications Review*, 34(2):55–70, April 2004.

[9] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. *ACM SIGCOMM*, August 1988.

[10] Yvo Desmedt, Yair Frankel, and Moti Yung. Multi-receiver / Multi-sender Network Security: Efficient Authenticated Multicast / Feedback. *IEEE INFOCOM'92*, pages 2045–2054, 1992.

[11] D. Eastlake and P. Jones. *US Secure Hash Algorithm 1 (SHA1)*, September 2001. RFC 3174.

[12] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/Off-line Digital Signatures. *Advances in Cryptology - Crypto'89*, LNCS(435):263–275, 1990.

[13] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/Off-line Digital Signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[14] Hiroshi FUJII, Wattanawong KACHEN, and Kaoru KUROSAWA. Combinatorial Bounds and Design of Broadcast Authentication. *IEICE Trans.*, E79-A(4):502–506, 1996.

[15] Rosario Gennaro and Pankaj Rohatgi. How to Sign Digital Streams. *Advances in Cryptology, CRYPTO'97*, 1997.

Protocol	Latency at the source	Latency at receivers	Tolerance to packet loss	Authentication information size
Simple Off-line chaining	Yes	No	No	$\|d\|$
EMSS	No	Yes	The authentication probability of a packet is at least 90%	$6\|d\|$
p-random graphs	Yes	No	$\Pr(P_i$ is verifiable $\mid P_i$ is received$) \geq 1-(1-p)(1-(p(1-q))^2)^{i-2}$	$p(n-1)$ hashes in average
Augmented Chain $C_{a,p}$	Yes	Yes	Yes: Each block of packets tolerates a single burst of length up to $p(a-1)$	$2\|d\|$ in average
Piggybacking	Yes	Yes	Yes: Each prioritized packets' set S_i tolerates x_i bursts of b_i packets	$\|d\| \times (x_i+1)$ at least, for a packet in class S_i
A^2Cast	No	Yes	Yes: 99% average verification ratio	*Source Driven*: depends on average packet loss ratio faced by receivers
RLH	No	Yes	Yes: 99% average verification ratio	*Receiver Driven*: depends on actual packet loss ratio faced by each receiver

$|d|$: size of a digest (hash). n: number of packets in a block. $|S|$: size of a signature. q: loss probability of a packet.

TABLE III

COMPARISON OF SOME *data origin authentication with non repudiation* PROTOCOLS

[16] Rosario Gennaro and Pankaj Rohatgi. How to Sign Digital Streams. *Information and Computation*, 165(1):100–116, February 2001.

[17] Philippe Golle and Nagendra Modadugu. Authenticating Streamed Data in the Presence of Random Packet Loss. *NDSS'01: The Network and Distributed System Security Symposium*, 2001.

[18] Thomas Hardjono and Gene Tsudik. IP Multicast Security : Issues and Directions. *Annales de telecom*, 2000.

[19] Paul Judge and Mostafa Ammar. Security Issues and Solutions in Multicast Content Distribution: A Survey. *IEEE Network*, pages 30–36, January/February 2003.

[20] Kurosawa K. and Obana S. Characterization of (k,n) multi-receiver authentication. *Information Security and Privacy, ACISP'97*, LNCS(1270):204–215, 1997.

[21] B. Kaliski. *The MD2 Message-Digest Algorithm*, April 1992. RFC 1319.

[22] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security : Private Communication in a Public World*. Printice Hall Series in Computer Networking and Distributed Systems, 2002.

[23] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*, February 1997. RFC 2104.

[24] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *HAND BOOK OF APPLIED CRYPTOGRAPHY*. 1996.

[25] Sara Miner and Jessica Staddon. Graph-Based Authentication of Digital Streams. *IEEE Symposium on Security and Privacy*, 2001.

[26] Michael Mitzenmacher and Adrian Perrig. Bounds and Improvements for BiBa Signature Schemes. *Technical Report (TR-02-02), Harvard University*, 2002.

[27] A. Pannetrat and R. Molva. Efficient Multicast Packet Authentication. *10th Annual Network and Distributed System Security Symposium*, February 2003.

[28] Alain Pannetrat and Refik Molva. Authenticating Real Time Packet Streams and Multicasts. 7^{th} *International Symposium on Computers and Communications, ISCC'02*, pages 490–495, July 2002.

[29] J. M. Park, E. K. P. Chong, and H. J Siegel. Efficient Multicast Packet Authentication Using Signature Amortization. *IEEE Symposium on Security and Privacy*, 2002.

[30] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient Multicast Stream Authentication Using Erasure Codes. *ACM Transactions on Information and System Security*, 6(2):258–285, May 2003.

[31] Vern Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, June 1999.

[32] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. *IEEE Symposium on Security and Privacy*, 2000.

[33] Adrian Perrig. The BiBa One-Time Signature and Broadcast Authentication Protocol. *The 8th ACM Conference on Computer and Communications Security*, November 2001.

[34] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and Secure Source Authentication for Multicast. 8^{th} *Annual Internet Society Symposium on Network and Distributed System Security*, 2001.

[35] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*, 5, Summer 2002.

[36] Federal Information Processing Standards Publication. *Data Encryption Standard (DES)*, December 1993. FIPS PUB 46.

[37] Federal Information Processing Standards Publication. *Digital Signature Standard (DSS)*, May 1994. FIPS PUB 186.

[38] Federal Information Processing Standards Publication. *Advanced Encryption Standard (AES)*, November 2001. FIPS PUB 197.

[39] Sandro Rafaeli and David Hutchison. A Survey of Key Management for Secure Group Communication. *ACM Computing Surveys*, 35(3):309–329, September 2003.

[40] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short One-time Signatures with Fast Signing and Verifying. 7^{th} *Australian Conference on Information Security and Privacy*, LNCS(2384):144–153, 2002.

[41] R. Rivest. *The MD5 Message-Digest Algorithm*, April 1992. RFC 1321.

[42] Ronald L. Rivest, Adi Shamir, and Leonard M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[43] Pankaj Rohatgi. A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication. 6^{th} *ACM Conference on Computer and Communications Security CCS'99*, pages 93–100, November 1999.

[44] Obana S. and Kurosawa K. Bounds and Combinatorial Structure of (k,n) Multi-receiver A-codes. *Designs, Codes and Cryptography*, 22(1):47–63, 2001.

[45] R. Safavi-Naini and H. Wang. New Results on Multi-receiver Authentication Codes. *Advances in Cryptology: EUROCRYPT'98*, LNCS(1403):527–541, 1998.

[46] R. Safavi-Naini and H. Wang. Multireceiver Authentication Codes: Models, Bounds, Constructions, and Extensions. *Information and Computation*, 151:148–172, 1999.

[47] Bruce Schneier. *APPLIED CRYPTOGRAPHY: Protocols, Algorithms, and Source Code in C*. Second edition, 1996.

[48] R. Shirey. *Internet Security Glossary*, May 2000. RFC 2828.

[49] Chung Kei Wong and Simon S. Lam. Digital Signatures for Flows and Multicasts. *IEEE ICNP'98*, October 1998.

[50] Chung Kei Wong and Simon S. Lam. Digital Signatures for Flows and Multicasts. *IEEE/ACM Transactions on Networking*, 7(4), August 1999.

[51] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network. *IEEE Global Telecommunications Conference (IEEE/GLOBECOM'96)*, pages 94–99, November 1996.

[52] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and Modeling of the Temporal Dependence in Packet Loss. *INFOCOM'99*, pages 345–352, March 1999.