Multicast Routing Authentication System using Advanced Tesla

V. Ganesh Babu
Research Scholar
Vinayaka Missions University
Salem, Tamil Nadu

Dr. T. Senthil Kumar
Anna University-Tiruchirappali
Tiruchirappali, Tamil Nadu

ABSTRACT
The present strategies that reduce the delay associated with multicast authentication, make more efficient usage of receiver-side buffers, make delayed key disclosure authentication more resilient to buffer overflow denial of service attacks, and allow for multiple levels of trust in authentication. Throughout this base paper, the main focus of discussion will be on the popular multicast authentication scheme Timed Efficient Stream Loss-tolerant Authentication (TESLA) based upon the delayed key disclosure principle. Similar to other schemes based upon delayed key disclosure, TESLA is susceptible to Denial-of-Service (DoS) attacks and is not well suited for delay-sensitive applications.

Keywords:
Denial-of-Service (DoS) attacks, forge – capable area, Message Authentication Code (MAC), multigrade source authentication, Timed Efficient Stream Loss-tolerant Authentication (TESLA), Packet Interleaving, trust.

1. INTRODUCTION

A multicast protocol enables a sender to efficiently disseminate digital media data to many receivers. Due to the time-sensitive requirement of some applications, reliable transmission protocol like TCP (Transmission Control Protocol) is impractical for multicast. Therefore, unreliable transmission protocol such as UDP (User Datagram Protocol) is generally adopted for multicast applications. Multicast protocol is suitable for many applications, e.g. video transmissions, live broadcasts, stock quotes, or news feeds. These applications may have many receivers or distribute time-sensitive data. To ensure secure communications between a sender and its receivers, it is important to implement security measures in a multicast environment.

An attacker may impersonate a sender to transmit malicious packets to receivers, causing disruptions in the communications. Multicast authentication is used to defend against forged packets injected by the attackers by enabling a receiver to authenticate the packet source and discard malicious packets. There have been many multicast authentication approaches, which can be roughly divided into two categories: symmetric cryptographic primitives and asymmetric cryptographic primitives. Symmetric cryptographic primitives, such as MAC (Message Authentication Code), generally use a symmetric key to authenticate a data source. In MAC, an identical secret key is maintained by the sender and receiver. The sender uses the secret key to generate a MAC for a packet, and the receiver is able to authenticate the packet source by verifying the MAC of the packet with the secret key. Asymmetric cryptographic primitives, such as digital signatures, use an asymmetric key pair to authenticate a data source. In general, an asymmetric key pair consists of two keys; one key is used to generate the signature, while the other key is used to verify the signature. Using digital signatures like RSA, for authentication is popular and believed to be secure; nevertheless, digital signature generation and verification incur significant computation overhead.

Signature amortization addresses this concern by generating a single digital signature for a block of packets. After verifying the signature, a receiver can consider this block of packets authentic. Signature amortization makes a tradeoff between security and computation overhead. Fault-tolerant coding algorithms, like erasure codes, (Luby. M. G *et al*, 2001), (Rabin. M. O, 1989), (Reed *et al*,1960) or diversity codes (Ayanoglu *et al*, 1993), partition information into many segments and can correctly reconstruct the original information even though a number of segments, up to a threshold, may be lost. Although signature amortization with a fault-tolerant coding algorithm reduces computation overhead and tolerates packet loss, it suffers from pollution attacks (Chris Karlof *et al*, 2004). Pollution attacks occur when an adversary injects a large quantity of forged packets into a block of valid packets. The receiver fails to decode a correct signature using the fault-tolerant coding algorithm, forcing the receiver to drop the entire block of packets, which may include valid packets.

Distillation codes (Chris Karlof *et al*, 2004) have been proposed for signature amortization to defend against pollution attack. In distillation codes, the sender augments each packet with a witness. Distillation codes guarantee that all valid packets are partitioned into groups that do not contain forged packets, allowing the receiver to decode the correct signature from the packets in this group. Unfortunately, the receiver cannot realize in advance which group contains valid packets; therefore, it must attempt to decode a valid signature from each group. Furthermore, the receiver cannot immediately distinguish between valid and invalid packets, making it necessary to first buffer all received packets. Distillation codes incur high computation overhead, storage space, and verification delay.

2. EFFICIENT MULTICAST PACKET AUTHENTICATION USING SIGNATURE AMORTIZATION

2.1 Rationale for this approach
The sender transmits the hash of a packet appended to *k* other packets for increased resistance to loss. The authors assume that a block consists of *n* packets. If independent packet loss is

assumed, then the probability that at least one out of the k packets will reach the destination is 1- q^k , where q is the packet loss probability. The communication overhead would be kh, where h is the size of the hash. Using the same overhead, one can encode the hash using IDA and append the encoded n segments to the n packets of the block (i.e., each packet in the block would contain one of the encoded segments). The minimum number of encoded segments needed for reconstruction of the hash is only$m = \lceil n/k \rceil$, where $\lceil x \rceil$ denotes the smallest integer not less than x. The probability that the hash can be reconstructed successfully at the receiver is given by

$$1 - \sum_{i=0}^{m-1} \binom{n}{i} (1-q)^i q^{n-i} \quad (1)$$

Again, independent packet loss was assumed.

2.2 Signature amortization using IDA

At the receiving end, verification of the packet stream is straightforward. Assuming that at least m authenticated packets are received, the receiver can successfully reconstruct F^1 and $\sigma(K_r, H_G(G_1))$ from any combination of m packets as follows:

1. Assume that segments $F_1^1, \ldots, F_m^1$ are received. Using the m pieces, it is readily seen that
$$A.\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_1.S_1 \\ \vdots \\ a_m.S_1 \end{bmatrix}$$
where $\mathbf{A} = (a_{ij})1 \leq i,j \leq m$ is an m x m matrix whose i-th row is $\boldsymbol{a_i}$.
2. Because $\mathbf{A}$ is invertible (because of the independence condition on $\boldsymbol{a_i}$,$1 \leq i \leq n$), $\boldsymbol{S_1}$ can be obtained from
$$S = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = A^{-1}.\begin{bmatrix} a_1.S_1 \\ \vdots \\ a_m.S_1 \end{bmatrix}$$
3. Using the same procedure, $\boldsymbol{S_2}, \ldots, \boldsymbol{S_{N/m}}$ can be obtained, and F^1 is reconstructed by concatenating these values.
4. The same technique is applied to reconstruct $\sigma(K_r, H_G(G_1))$.
5. Once the reconstruction is complete, all the packets in G_1 can be verified using F^1 and $\sigma(K_r, H_G(G_1))$.

As already mentioned, SAIDA might not be appropriate in situations where the data to be sent is generated in real time is given by (3), and immediate broadcast of it is crucial. This scheme is most useful in cases where the sender has a priori knowledge of at least a portion of the data to be broadcast (e.g., broadcast of prerecorded material).

3. ENSURING AUTHENTICTION OF DIGITAL INFORMATION USING CRYPTOGRAPHIC ACCUMULATORS

3.1 Stream Authentication over Erasure Channels

In the remaining of this base paper, the authors work with a unforgeable S-bit long digital signature (Sign_{SK},Verify_{PK}) (Stinson, 2006) the key pair of which (SK,PK) is created by a generator KeyGen and a cryptographic hash function h' outputing H′-bit long digests with $H' = k \log_2(r)$.

3.2 Authentication Protocol

The stream is a continuous flow of information. First, the sender generates the signature σ on the digest $h'(P_1)$ of the first stream packet. He then encodes the concatenation $\sigma||h'(P_1)$ using a MDS code of length n and dimension $n - t$. The corresponding codeword is denoted $(C_1, \ldots, C_n)$ where each C_i is $\left\lceil \frac{S+\mathcal{H}'}{n-t} \right\rceil$ - bit long.

Second, the sender buffers the first n packets $P_1, \ldots, P_n$ as list L_1. He computes the accumulated value A_1 of L_1 and builds the augmented packet: $AP_1 := 1||P_1||A_1||C_1$. Third, when a new stream packet P_{n+j-1} ($j \geq 2$) is available, the sender builds the list $L_j := \{P_j, \ldots, P_{n+j-1}\} \cup \{h'(P_1)\}$. He computes the corresponding accumulated value A_j and builds the augmented packet: $AP_j := j||P_j||A_j||C_{[j]}$ where $[j]$ denotes the unique integer in $\{1, \ldots, n\}$ congruent to j modulo n. In particular: $[n] = [2n] = [3n] = \cdots = n$. We notice that the delay at the sender is n packets as it sends into the network AP_j after P_{n+j-1} be available.

3.3 Analysis of the Protocol

Theorem 1 Our authentication scheme is a non-degrading authentication protocol. The sender processes data with a delay of n packets throughout streaming while the receiver can authenticate packets on-the-fly from the $(n - t + 1)^{th}$ received element.

Remark 1 A single signature is needed to ensure non-repudiation of the **whole** stream.

Remark 2 One can notice that, when n is fixed, the lower t is, the larger the delay gets. This might seem to be surprising at first but having low t's implies having small redundancy for the codeword coordinates as n−t is large. That is why one requires more codeword information to reconstruct $(C_1, \ldots, C_n)$. The trade-off delay/overhead is an efficiency trade-off.

4 STREAM AUTHENTICATION OVER ADVERSARIAL CHANNELS

In this channel model, O can inject bogus data packets into the network. In this situation, we will process the whole data stream per block of n packets $P_1, \ldots, P_n$. Each of these blocks is located within the whole stream using an identification value BID. This approach is used in the different schemes designed for adversarial networks including TWMDS and PRABS. This is to be opposed to the on-the-fly authentication process from the $(n - t + 1)^{th}$ packet at the receiver for the protocol.

4.1 Scheme Overview

Due to erasure of information, we want to generate n augmented packets $AP_1, \ldots, AP_n$ such that we can reconstruct all packets $P_1, \ldots, P_n$ from any $\lceil \alpha n \rceil$ -subset of $\{AP_1, \ldots, AP_n\}$. Therefore, our first step consists of encoding $P_1, \ldots, P_n$ using a $[n, \lceil \alpha n \rceil, n - \lceil \alpha n \rceil + 1]$ code since it can

correct up to $n - \lceil \alpha n \rceil$ erasures. Note that this approach implies that the elements of the code alphabet are larger than the size of a data packet as the message $(M_1, \ldots, M_{\lfloor \alpha n \rfloor})$ to be encoded into the codeword $(C_1, \ldots, C_n)$ should represent the concatenation $P_1 \| \ldots \| P_n$.

4.2 Authentication Protocol

We assume that the values α and β are rational numbers so that we can represent them over a finite number of bits. In order to run Poly-Reconstruct as a subroutine of MPR, we have to choose a parameter $\rho \in \left(0, \frac{\alpha^2}{\beta}\right)$. Notice that ρ has to be rational since ρ n is an integer. Without loss of generality, one can consider that the value ρ is uniquely determined when n, α and β are known. The scheme parameters which are assumed to be publicly known. The bit size S of the signature and its public key PK are also publicly known. Note that, once r is known, then k is uniquely determined since the digests of h' are $(k \log_2(r))$-bit long.

4.3 Analysis of the Protocol

Security: As the channel model allows an adversary to inject bogus elements into the network, we adopt the same security definition as in (Christophe Tartary *et al*, 2006).

Here two protocols are presented for the broadcast authentication problem using a modified version of Nyberg's accumulator due to Yum *et al*. The first scheme was related to erasure channels. It was shown that its packet overhead was less than the length of a digest and, in particular, far less than (Park *et al*, 2004, Park *et al*, 2003, Park *et al*, 2002, Pannetrat *et al*, 2002). Even if the sender processes the stream in delay of n packets, the receivers can authenticate packets on-the-fly from the $(n - t + 1)^{th}$ received element to the end of the stream (if any). In addition, a single signature is needed to provide the non-repudiation of the whole stream. The second scheme was designed for adversarial networks. It is obvious that the number of signature queries at the receiver is the same as for TWMDS due to the use of Poly-Reconstruct in both constructions. This number turns to be $O(1)$ as a function of the block length n (Christophe Tartary *et al*, 2006). Furthermore, the packet overhead of our new scheme is smaller than PRABS' and TWMDS'. Another interesting result from this comparative study was that we obtained more a extensive comparison between PRABS and TWMDS showing that the overhead of TWMDS was smaller.

5 MULTICAST SECURITY: A TAXONOMY AND SOME EFFICIENT CONSTRUCTIONS

Multicast communication is becoming the basis for a growing number of applications. It is therefore critical to provide sound security mechanisms for multicast communication. Yet, existing security protocols for multicast offer only partial solutions. We first present taxonomy of multicast scenarios on the Internet and point out relevant security concerns. Next we address two major security problems of multicast communication: source authentication, and key revocation. Maintaining authenticity in multicast protocols is a much more complex problem than for unicast; in particular, known solutions are prohibitively inefficient in many cases.

5.1 Multicast Security issues

We overview salient characteristics of multicast scenarios, and discuss the relevant security concerns. The various scenarios and concerns are quite diverse in character (sometimes they are even contradictory). Thus it seems unlikely that a single solution will be satisfactory for all multicast scenarios. This situation leads us to suggest two benchmark scenarios for developing secure multicast solutions

5.2 Efficient Authentication Schemes Per-message unforgability of MAC schemes:

We distinguish between two types of attacks against a MAC scheme. One is a complete break, where the attacker can authenticate any message of its choice (e.g., a key recovery attack). The other attack allows the attacker to randomly authenticate false messages; here the attacker can authenticate a given message with some fixed and small probability (but does not know a-priori whether it will be able to authenticate the message). Our schemes do not allow complete break with higher probability than the underlying MAC scheme. Yet, we do allow for random authentication errors with non-negligible probability (say, 2^{-20} up to $2^{-10)}$.

5.3 The Basic Authentication Scheme for a Single Source

Let w be the maximum number of corrupted users. The basic scheme proceeds as follows:

- The source of the transmissions (S) knows a set of $l = e\,(w + 1)\,ln\,(1/q)$ keys, $R = \langle K_1, \ldots, K_l \rangle$
- Each recipient u knows a subset of keys $R_u \subset R$. Every key K_i is included in R_u with probability $1/(w+1)$, independently for every i and u.
- Message M is authenticated by S with each key K_i using a MAC and $\langle MAC(K_1, M), MAC(K_2, M), \ldots, MAC(K_l, M) \rangle$ is transmitted together with the message.

Each recipient u verifies all the MACs which were created using the keys in its subset R_u. If any of these MACs is incorrect then u rejects the message

5.4 Smaller Communication Overhead

We now describe a scheme with a lower communication overhead. The idea behind it is that using just four times as many keys as in the basic scheme, one can ensure that the coalition does not know $log\,(1/q)$ of the user's keys. Each key can therefore be used to produce a MAC with a *single bit output* and the communication overhead is improved. The coalition would have to guess $log\,(1/q)$ bits to create a false authentication and
its probability of success is as before.
Recall the basic scheme: it limits the success probability of a corrupt coalition to be $q+q'$, where q' is the per-message unforgeability. The MAC output must be at least $log_2\,(1/q')$ bits long. Therefore, assuming $q=q'$, the communication overhead is $C > e(w+1)ln^2\left(\frac{1}{q}\right)$ *bits*. The improved scheme achieves a communication overhead smaller than $4e(w+1)\ln\left(\frac{1}{q}\right)$ *bits*.

5.5 Multiple Dynamic Sources

It supports a *dynamic* set of sources and has the following properties:

- The total number of keys is as in schemes for a single source, but every party can send authenticated messages.
- The scheme does not require the set of sources to be defined in advance or to contain all parties. Rather, it allows to *dynamically* add sources.
- The scheme distinguishes between the set of sources and the set of receivers. Only coalitions of more than *w receivers* can send false authenticated messages. The keys of sources do not help such coalitions. This property is especially useful if receivers are more trusted than senders, as might be the case for example if the receivers are network routers.
- The scheme provides a computational (rather than information theoretic) security against revealing to a coalition all the keys in the intersection of a source and a receiver's subsets.

The scheme uses a family of pseudo-random functions *{ f_k }* (see (Luby.M, 1996) for a discussion of pseudo-random functions). It is based on a single source scheme and can be built upon the basic scheme or the communication efficient scheme.

5.5.1 Initialization: The scheme uses *l primary keys* $\langle k_1, \ldots, k_l \rangle$, where *l* is as in the single source schemes (*l=O (w log (1/q))*). Each key k_i defines a pseudo-random function f_{k_i}.

5.5.2 Receiver Initialization: Each party *v* which intends to receive messages obtains a subset R_v of primary keys. Every primary key k_i is included in R_v with probability *1/(w+1)*.

5.5.3 Source Initialization: Every party *u* which wishes to send messages receives a set of *secondary keys* $S_u = \langle f_{k_1}(u), f_{k_2}(u), \ldots, f_{k_l}(u) \rangle$. This set can be sent any time after the system has been set-up, and the identity or the number of sources does not have to be defined in advance.

5.5.4 Message Authentication: When a party *u* sends a message *M* it authenticates it with all the secondary keys in S_u. That is $\forall k \in S_u$, it computes and attaches a MAC of *M* with *k*.

5.6 Signatures vs. MACs: a rough performance comparison

Compared to the performance of public key signatures, our authentication schemes dramatically reduce the running time of the authenticator. The running time of the verifier and the communication overhead are of the same order as public key signatures (the exact comparison depends on the size of the corrupt coalitions against which the schemes operate).

It is seen that the signing time is much shorter in our scheme than with public key given (8)signatures. The verification time is comparable to (highly optimized) RSA and much faster than DSS.

Table 1. A Performance Comparison of Authentication Schemes

	Auth.	**Ver.**	**Comm.**	**Source Key**	**Receiver Key**
Units	(ops/ sec)	(ops/sec)	(bits)		
RSA 1024 bits	50	30000	1024	2048 bits	1024 bits
DSS, 768 bits	70	40	1536	1536 bits	1536 bits
Basic scheme, *w=10*, $q=10^{-3}$	2650	26500	1900	190 MAC keys	19 MAC keys
Low Comm. *w=10*, $q=10^{-3}$	660	6600	760	760 MAC keys	76 MAC keys
Perfect Sec. $n=10^4$, $q=10^{-3}$	200	2000	25000	2500 MAC keys	250 MAC keys

5.7 Dynamic Secrecy – User Revocation

We survey some solutions for the member deletion problem, describe a particularly appealing construction from (Wallner *et al*), (Wong *et al*, 1998) based on binary trees, and present an improved construction with reduced communication overhead. They also show how our construction is more resistant to a certain kind of attack

5.8 A Tree Based Scheme

Tree based group rekeying schemes were suggested by Wallner et al. (Wallner *et al*) (who used binary trees), and independently by Wong et al. (Wong *et al*, 1998) (who consider the degree of the nodes of the tree as a parameter). We concentrate on the scheme of (Wallner *et al*) since it requires a smaller communication overhead per user revocation. This scheme applied to a group of *n* users requires each user to store *log n+1* keys. It uses a message with *2log n -1* key encryptions in order to delete a user and generate a new group key. This process should be repeated for every deleted user. The scheme has better performance than the Fiat-Naor scheme when the number of deletions is not too big. It is also secure against any number of corrupt users (they can all be deleted from the group, no matter how many they are). A drawback of this scheme is that if a user misses some control packet relative to a user deletion operation (e.g., if it temporarily gets disconnected from the network), it needs to either ask for all the missed control packets, or incur in a communication overhead comparable to a user addition operation.

5.9 The Improved Scheme

The improved scheme reduces the communication overhead of (Wallner *et al*) by a factor of two, from *2 log n* to only *log n*. The initialization of the scheme is the same as in (Wallner *et al*). The authors now describe the user revocation

procedure. Let *G* be pseudo-random generator which doubles the size of its input (Blum *et al*, 1984). Denote by *L(x), R(x)* the left and right halves of the output of *G(x)*, i.e., *G(x) = L(x) R(x)* where *|L(x)| = |R(x)| = |x|*. To remove a user *u*, the group controller associates a value r_v to every node *v* along the path from *u* to the root as follows: It chooses $r_{p(u)} = r$ at random and sets $r_{p(v)} = R(r_v) = R^{|u|-|v|}(r)$ for all other *v* (where *p(v)* denotes the parent of *v*). The new keys are defined by $k_v' = L(r_v) = L(R^{|u|-|v|-1}(r))$. Notice that from r_v, one can easily compute all keys $k_v', k_{p(v)}', k_{p(p(v))}'$ up to the root key k'. Finally each value $r_{p(v)}$ is encrypted with key $k_{s(v)}$

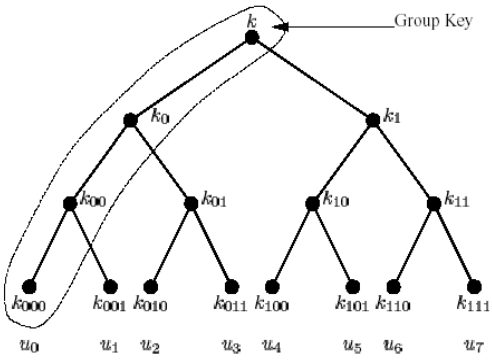

Fig1.1 The tree key data structure (the keys of u_o are encircled).

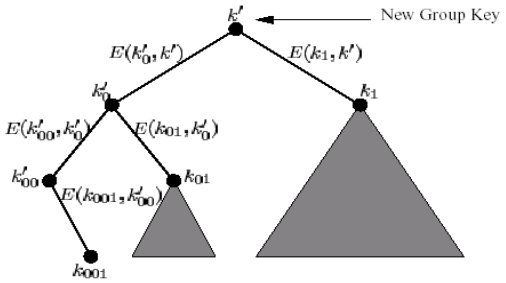

Fig1.2 Key revocation in the basic scheme.

5.10 Advantages of the new scheme

This construction halves the communication overhead of the basic scheme to only *log n* and its security can be rigorously proven. It has an additional advantage: In the scheme of Wallner et al the group controller chooses the group key (the root key), whereas is our construction this key is the output of a pseudo-random generator. Suppose that there is an adversary which can break encryptions performed with a subset of the key space (for example keys in which certain bits have a linear dependency), and furthermore that this adversary has gained temporary control over the group controller (e.g. when the controller was manufactured). Then if the scheme of (Wallner *et al*) is used, the adversary might corrupt the method by which the group controller generates keys in such a way that the root key would always be chosen from the "weak" subspace. However, if our scheme is used, and the pseudo-random generator *G(x) = L(x) R(x)* is cryptographically strong, then it will be hard to find values *r* such that the root key *k = L (R (R(…(r) …)))* is weak.

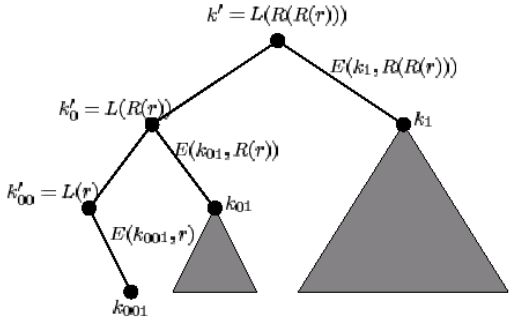

Fig: 1.3Key revocation in the improved scheme.

6 EFFICIENT AND SECURE SOURCE AUTHENTICATION FOR MULTI-CAST

One of the main challenges of securing multicast communication is source authentication, or enabling receivers of multicast data to verify that the received data originated with the claimed source and was not modified enroute. The problem becomes more complex in common settings where other receivers of the data are not trusted, and where lost packets are not retransmitted. Several source authentication schemes for multicast have been suggested in the past, but none of these schemes is satisfactorily efficient in all prominent parameters. We recently proposed a very efficient scheme, TESLA that is based on initial loose time synchronization between the sender and the receivers, followed by delayed release of keys by the sender. This paper proposes several substantial modifications and improvements to TESLA. One modification allows receivers to authenticate most packets as soon as they arrive (whereas TESLA requires buffering packets at the receiver side, and provides delayed authentication only). Other modifications improve the scalability of the scheme, reduce the space overhead for multiple instances, increase its resistance to denial-of-service attacks, and more.

Another approach to providing source authentication uses only symmetric cryptography, more specifically on message authentication codes (MACs), and is based on delayed disclosure of keys by the sender. This technique was first used by Cheung (Cheung, 1997) in the context of authenticating communication among routers. It was then used in the Guy Fawkes protocol (Anderson *et al*, 1998) for interactive unicast communication. In the context of multicast streamed data it was proposed by several authors (Briscoe, 2000), (Bergadano *et al*, 2000a), (Bergadano *et al*, 2000b), **(**Perrig *et al*, 2000a). In particular, the TESLA scheme described in (Perrig *et al*, 2000a) was presented to the reliable multicast transport (RMT) working group of the IETF and the secure multicast (SMuG) working group of the IRTF and was favorably received. TESLA is particularly well suited to provide the source authentication functionality for the MESP header (Canetti *et al*, 2000), or for the ALC protocol proposed by the RMT (Luby *et al*, 2000) is given by (10). Consequently, an Internet-Draft describing the scheme was recently written (Perrig *et al*, 2000b).

6.1 An Overview of TESLA

The security property TESLA guarantees is that the receiver never accepts M_i as an authentic message unless M_i was actually sent by the sender. Note that TESLA does not

provide non-repudiation, that is, the receiver cannot convince a third party that the stream arrived from the claimed source.

TESLA is efficient and has a low space overhead mainly because it is based on symmetric-key cryptography. Since source authentication is an inherently asymmetric property (all the receivers can verify the authenticity but they cannot produce an authentic data packet), we use a delayed disclosure of keys to achieve this property. Similarly, the data authentication is delayed as well. In practice, the authentication delay is on the order of one roundtrip- time (RTT).

TESLA has the following properties. First, it has a low computation overhead, which is typically only one MAC function computation per packet, for both sender and receiver.

TESLA also has a low per-packet communication overhead, which is about 20 bytes per packet. In addition, TESLA tolerates arbitrary packet loss. Each packet that is received in time can be authenticated. Except for initial time synchronization, it has only unidirectional data flow from the sender to the receiver. No acknowledgments or other messages are necessary. This implies that the sender's stream authentication overhead is independent of the number of receivers, hence TESLA is very scalable. TESLA can be used both in the network layer and in the application layer. The delayed authentication, however, requires buffering of packets until authentication is completed.

6.2 Bootstrapping a new Receiver

TESLA requires an initially authenticated data packet to bootstrap a new receiver. This authentication is achieved with a digital signature scheme, such as RSA (Rivest *et al*, 1978), or DSA (NIST, 1991). We consider two options for synchronizing the time, direct and indirect synchronization. Whichever time synchronization mechanism is used, the receiver only needs to know an upper bound on the sender time.

The initial authenticated packet contains the following information about the time intervals and key chain:

- The beginning time of a specific interval T_j , along with its id I_j
- The interval duration T_{int}
- Key disclosure delay d (unit is interval)
- A commitment to the key chain K_i ($i < j$ -d where j is the current interval index)

6.3 Sending Authenticated Packets

Each key of the key chain is used in one time interval. However many messages are sent in each interval, the key which corresponds to that interval is used to compute the MAC of all those messages. This allows the sender to send packets at any rate and to adapt the sending rate dynamically. The key remains secret for d-1 future intervals. Packets sent in interval I_j can hence disclose key K_{j-d}. As soon as the receivers receive that key, they can verify the authenticity of the packets sent in interval I_{j-d}.

6.4 Receiver Tasks

Since the security of TESLA depends on keys that remain secret until a pre-determined time period, the receiver must verify for each packet that the key, which is used to compute the MAC of that packet, is not yet disclosed by the sender. Otherwise, an attacker could have changed the message data and re-computed the MAC. This motivates the security condition, which the receiver must verify for each packet it receives

6.5 Authors Extensions

The authors extend TESLA in a number of ways to make it more efficient and practical. First, we present a new method to support *immediate authentication*, meaning that the receiver can authenticate packets as soon as they arrive.

Second, they propose optimizations concerning key chains. In particular, for applications that use multiple authentication chains with different disclosure delays, we present a new algorithm that reduces the communication overhead.

6.6 Immediate Authentication

A drawback of the original TESLA protocol is that the receiver needs to buffer packets during one disclosure delay before it can authenticate them. This might not be practical for certain applications if the receivers cannot afford much buffer space and bursty traffic might cause the receivers to drop packets due to insufficient buffer space. Moreover, as we show later in section 2.5.4.1, the requirement of receiver buffering introduces a vulnerability to a denial-of-service attack. To solve these problems caused by receiver-buffering, we propose a new method to support *immediate authentication*, which allows the receiver to authenticate packets as soon as they arrive.

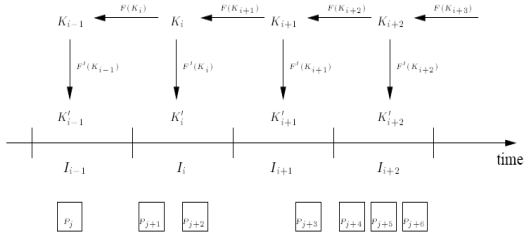

Fig: 1.4:TESLA key chain and the derivedMAC keys

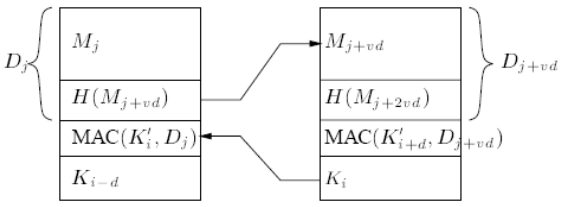

Fig: 1.5Immediate authentication packet example.D_j = $H(M_{j+vd}) \mid M_j$ and D_{j+vd} = $H(M_{j+2vd}) \mid M_{j+vd}$.

If each packet can only carry the hash of one other packet, it is clear that the sending rate needs to remain constant. Also it is clear that if a packet is lost, the corresponding packet cannot be immediately authenticated. To achieve flexibility for dynamic sending rate and robustness to packet loss, the sender can add the hash values of multiple future packets to a packet, similar to the EMSS scheme is given by (11) (Perrig *et al*, 2000a).

6.7 Concurrent TESLA instances

In this section, the authors present a space optimization technique in the case the sender uses multiple TESLA instances for one stream.

Choosing the disclosure delay involves a tradeoff. Receivers with a low network delay welcome short key disclosure delays because that translates into a short authentication delay. Unfortunately, receivers with a long network delay could not operate with a short disclosure delay because most of the packets will violate the security condition and hence cannot be authenticated. Conversely, a long disclosure delay would suit the long delay receivers, but causes unnecessarily long authentication delay for the receivers with short network delay. The solution is to use multiple instances of TESLA with different disclosure delays simultaneously, and each receiver can decide which disclosure delay, and hence, which instance to use. A simple approach to use concurrent TESLA instances is to treat each TESLA instance independently, with one key chain per instance. The problem for this approach is that each extra TESLA instance also causes extra space overhead in each packet. If each instance requires 20 bytes per packet (80 bit for key disclosure and 80 bit for the MAC value), using three instances results in 60 bytes space overhead per packet. We present a new optimization which reduces the space overhead of concurrent instances.

6.8 Time Synchronization Issues

Loose time synchronization is an important component in TESLA. Although sophisticated time synchronization protocols exist, they usually require considerable management overhead. Furthermore, they generally have a high complexity and achieve properties that TESLA does not require. An example is the network time protocol (NTP) by Mills (Mills, 1992). Bishop performs a detailed security analysis of NTP (Bishop, 1990). For these reasons, we outline a simple and secure time synchronization protocol that suffices the humble requirements of TESLA.

Direct Time Synchronization

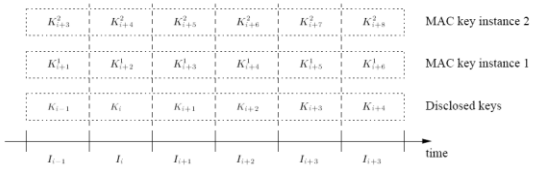

Fig: 1.6**Multiple TESLA instances key chain optimization.**

Figure :1.7 is shows sample time synchronization between the receiver and the sender. Upon receiving the signed response, the receiver checks the validity of the signature and the matching of the nonce and computes $\Delta = t_S - t_R$. It is easy to see that the Δ computed this way satisfies the requirement that $\Delta \geq \delta$. Because $\Delta = t_S - t_R = (t_S - t_3) + (t_3 - t_R)$, $t_S - t_3 = \delta$, and $t_3 - t_R$ is the network delay for sending the request from the receiver to the sender which is greater or equal to 0, hence $\Delta \geq \delta$. An interesting point is that the network delay of the response packet and the delay caused by the computation of the digital signature do not influence Δ at all. Since only the initial timestamp matters, it is important that the sender immediately stores the arrival time t_S of the time synchronization request packet. The subsequent processing and propagation delay does not matter.

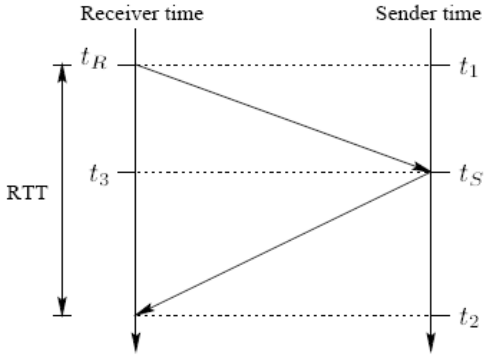

Fig:1.7The receiver synchronizes its time with the sender.

6.9 Indirect Time Synchronization

In indirect time synchronization, both the sender and the receivers synchronize their time with a time reference and hence the sender and the receiver can reach implicit time synchronization. This approach is favorable especially in cases where the application needs time synchronization with a time reference anyhow. Let $\Delta_{SC+|\epsilon_{SC}|}$ denote the measured upper bound of the difference of the sender's time and the time reference's time with $|\epsilon_{SC}|$ as the maximum error, and let $\Delta_{CR+|\epsilon_{CR}|}$ denote the measured upper bound of the difference of the time reference's time and the receiver's time with $|\epsilon_{CR}|$ as the maximum error. Thus the receiver could reach implicit time synchronization with the sender as $\Delta = \Delta_{SC} + \Delta_{CR} + |\epsilon_{SC}| + |\epsilon_{CR}|$ with $\epsilon = |\epsilon_{SC}| + |\epsilon_{CR}|$ as the maximum error.

6.10 Delayed Time Synchronization

Another interesting relaxation of the time synchronization requirement is that, if we assume that the receiver's clock drift is negligible during a period of time, then the receiver can receive the data stream from the sender before doing a time synchronization and authenticate the data later after a time synchronization. The receiver only needs to store the arrival time of each packet, so that it can evaluate the security condition after it performed the time synchronization. This is highly useful for many applications, for example a router can use TESLA to authenticate itrace messages (Bellovin, 2000), and the victim can authenticate the routers' IP markings afterwards when it wants to trace an attacker by performing approximate time synchronization with the router (Song *et al*, 2000).

6.11 Determining the Key Disclosure Delay

An important parameter to determine for TESLA is the key disclosure delay d. A short disclosure delay will cause packets to violate the security condition and cause packet drop, while a long disclosure delay causes a long authentication delay. Note that although the choice of the disclosure delay does not affect the security of the system, it is an important performance factor. We describe a new method

on how to choose a good disclosure delay d. In particular, we show as follows that if RTT is a reasonable upper bound on the round trip time between the receiver and the sender, then in case of using direct time synchronization, we can choose $d = \lceil RTT/T_{int} \rceil + 1$, where T_{int} is the interval duration. In case of indirect time synchronization, we can choose $d = \lceil (D_{SR}+\epsilon)/T_{int} \rceil + 1$, where ϵ is the sum of both the sender and receiver time synchronization error, and D_{SR} is a reasonable upper bound on the network delay of a packet traveling from the sender to the receiver.

6.12 Security Discussion and Robustness to DoS

In an IP multicast environment, however, DoS is a considerable threat and requires careful consideration. We discuss potential security problems in this section and show how to strengthen TESLA to thwart them. In particular, we show that there is no DoS attack on the sender if the receivers perform indirect time synchronization. In case of direct time synchronization, we show how to mitigate DoS attacks on the sender. Although there are some potential DoS attacks on the receiver side, we show that TESLA does not add any additional vulnerability to DoS attacks if the receiver has a reasonable amount of buffer space, otherwise we describe schemes that alleviate the exposure to DoS.

6.13 DoS Attack on the Sender

A DoS attack on the sender is not possible if TESLA is used with indirect time synchronization, because the sender does not keep per-receiver state or perform perreceiver operations. In the case of direct time synchronization, a DoS attack is possible, since the sender is required to digitally sign each nonce included in a time synchronization request. An attacker can perform a DoS by flooding the sender with requests.

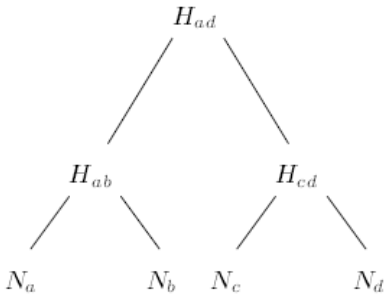

Fig:1.8Hash tree over receiver nonces refer(13). Node H_{ab} =H(N_a, N_b). H_{ad} = H(H_{ab},H_{cd}).

6.14 DoS Attack on the Receiver

In this section, we discuss two DoS attacks on the client. Since we assume the attacker could have full control of the network, some DoS attacks such as delay or drop packets are always possible. Delay packets could cause packets to violate the security condition and hence not to be authenticated. On the other hand, speeding up packets does not do anything at all. The receiver even benefits from this since she might be able to use a chain with a short disclosure delay that she could not use otherwise. We can show that replay packets cannot do much harm either. First, a duplicated packet is only accepted by the receiver within a short time period, since the security condition drops packets if they are replayed with a long delay. Second we can prevent the replay attack by adding a sequence number to each packet and by including the sequence number in the MAC. The TESLA protocol in the network layer or in the application layer will filter out duplicate packets.

6.15 DoS on the Packet Buffer

An powerful attack is to flood the multicast group with bogus traffic. This attack is serious because current multicast protocols do not enforce sending access control. The solution we propose involves a weak but efficient and immediate authentication method that offers some protection against a flooding attack. First if the receiver has a certain size buffer, we show that flooding cannot do much harm. Because the scheme only requires the receiver to buffer packets for the duration of one disclosure delay until the authenticity of the packets can be verified, hence the buffer size only needs to be the multiplication of the network bandwidth and the disclosure delay time. Assuming that the receiver has a 10Mbps network connection and a 500ms disclosure delay, the required buffer size is around 640kB, which should in general not be a major concern with today's workstations. Assuming 512byte network packets, the computation overhead to authenticate the packets is on the order of 1280 HMAC computations per second. Since the openssl HMAC-MD5 implementation processes on the order of 120; 000 512-byte blocks per second on a 500MHz Pentium III Linux workstation, the estimated processor overhead for TESLA authentication is on the order of 1% of the CPU time.

6.16 DoS on the Key Chain

Another DoS attack is specific to how the TESLA receiver reconstructs the key chain. If an attacker could fool a receiver to believe that a packet was sent out far in the future, and the receiver would try to verify the key disclosed in the packet by applying the pseudo-random function until the last committed key chain value. This attack can be easily prevented by checking that the packet interval is less or equal the latest interval that the sender can possibly be in. For an incoming packet sent in interval I_j, the receiver can verify if the interval I_j is not in the future, i.e. if the sender can already be in that interval. The verification condition is that $I_j < \lfloor (t_i - T_0)/T_{int} \rfloor$, where t_i is an upper bound on the sender's time that the receiver computes at the arrival of the packet.

6.17 Related Work

Researchers have proposed signing data packets to achieve source authentication. Since a digital signature achieves non-repudiation, a signature is much stronger than just authentication. As we mentioned in the introduction, the communication and computation overhead of current signature schemes is more expensive than schemes that are based on symmetric cryptography. We will review only the schemes that provide source authentication and not the schemes providing non-repudiation, i.e. (Gennaro *et al*, 1997), (Rohatgi, 1999), (Wong *et al*, 1998), (Perrig *et al*, 2000a).

7. CONCLUSION

To summarize, authentication in multicast applications is an important security measure that cannot be neglected. However, signing every multicast packet with a digital signature incurs high overhead, which may be impractical for many resource-limited devices. Signature amortization can reduce the computation and communication overhead, and a fault-tolerance coding algorithm can help tolerate packet loss. Despite these countermeasures, a signature amortization scheme still suffers pollution attack. To solve this problem, a lightweight and pollution attack resistant multicast authentication protocol (PARM) have been designed which is fast and lightweight and ideal for multicast applications with time-sensitive requirements or devices with limited computational power. In contrast to distillation codes, this scheme requires less computation overhead and storage space.

8. REFERENCES

[1] Alon.N (1991), "Probabilistic Methods in Extremal Finite Set Theory", in Extremal Problems for Finite Sets, 39–57.

[2] Anderson. R. J, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. M. Needham (1998). A new family of authentication protocols. Operating Systems Review, 32(4):9–20.

[3] Anna Lysyanskaya, Roberto Tamassia, and Nikos Triandopoulos (2003). Multicast authentication in fully adversarial networks. In IEEE Symposium on Security and Privacy, pages 241 – 253, Oakland, USA, IEEE Press.

[4] Ayanoglu. E, I. Chih-Lin, R.D. Gitlin, J.E. Mazo (1993). Diversity Coding for Transparent Self-Healing and Fault-Tolerant Communication Networks. IEEE Transactions on Communications, 41(11).

[5] Ballardie. A. J (1995), "A New Approach to Multicast Communication in a Datagram Network", Ph.D. Thesis, University College London.

[6] Bellare. M, R. Canetti and H. Krawczyk (1996), "Keying Hash Functions for Message Authentication", Advances in Cryptology – Crypto '96, LNCS vol. 1109, Springer-Verlag.

[7] Bellare. M, R. Canetti, and H. Krawczyk (1997). HMAC: Keyedhashing for message authentication. Internet Request for Comment RFC 2104, Internet Engineering Task Force.

[8] Bellovin. S (2000). The icmp traceback message. http:// www.research.att.com / ˜smb.

[9] Bergadano. F, D. Cavagnino, and B. Crispo (2000a), "Individual single source authentication on the mbone," in Proc. IEEE Int. Conf. Mutlimedia Expo. pp. 541-544.

[10] Bergadano. F, D. Cavagnino, and B. Crispo (2000b). Chained stream authentication. In Selected Areas in Cryptography 2000, Waterloo, Canada.

[11] Bhaskar. N and I. Kouvelas (2000). Source-specific protocol independent multicast. Internet Draft, Internet Engineering Task Force.

[12] Bishop. M (1990). A Security Analysis of the NTP Protocol Version 2. In Sixth Annual Computer Security Applications Conference.

[13] Brown, C. Perkins, and J. Crowcroft, "Watercasting: Distributed Watermarking of Multicast Media," Networked Group Commun. '99, Pisa, Italy, Nov. 1999, pp. 286–300.

[14] C. K. Wong, M. Goudo, and S. S. Lam, "Secure Group Communication Using Key Graphs." Prac. ACM SIGCOMM 1998.

[15] D. Bolenron, D. McGrew, and A. Sheimon, "Keyhkgement for Large Dynamic Groups: One-way Function Trees and Amortized Initialization," IETF Internet droft (work in progrerr], Feb. 1999

[16] D. M. Wollner, E. J. Harder, and R. C. Agee, "Key Management forMulti-cast: Issues and Architectures," IETF Info. RFC, Sept. 1998.

[17] H. Chu, L. Qiao, and K. Nahrstedt, "A Secure Multicast Protocol with Copyright Protection," Proc. IS&T/SPIE's Symp. Elect. Imaging: Sci. and Tech., Jan. 1999.

[18] I.Chang et al, "A toolkit for Secure Multicast Services over the Internet", IBM tech rep., l998.

[19] P. Q. Judge and M. H. Ammar, "WHIM: Watermarking Multicast Videowith a Hierarchy of Intermediaries," Proc. NOSSDAV, Chapel Hill, NC, June 2000.

[20] Perrig, R. Canetti, J. Tygar, and D. X. Song. Efficient authenticationand signing of multicast streams over lossy channels. In IEEE Symposium on Security and Privacy, May 2000.

[21] T. Hardjono and B. Cain, "Key establishment for IGMP authentication in IP multicast," in IEEE European Conference on Universal Multiservice Networks(ECUMN), CREF, Colmar, France, 2000.