A graph-theoretical analysis of multicast authentication

Aldar C-F. Chan
Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
10 King’s College Road, Toronto, ON M5S 3G4, Canada
aldar@comm.utoronto.ca

Abstract

Message authentication is considered as a serious bottleneck to multicast security, particular for stream-type of traffic. The techniques of hash chaining and signature amortization have been proposed in many schemes for stream authentication, with or without multicast settings. However, none of them is optimal. They either have a large packet overhead or are not robust to packet loss. Some even have a large receiver delay and/or require a large receiver buffer size. These schemes are constructed based on ad hoc or trial-and-error methods. There lack tools to evaluate and compare their performances. There is no systematic way to construct these schemes either. In this paper, we introduce the notion of dependence-graphs which links these hash-chained schemes to the well-known graph theory, and provides an effective analytical tool to evaluate the performance of these schemes in the presence of packet loss. Many important metrics of a hash-chained authentication scheme can be readily and easily determined from the properties of its dependence-graph. As well, a dependence-graph demonstrates the design tradeoff between pairs of metrics. In fact, the application of dependence-graphs is not limited to analyzing hash-chained schemes, we show that with slight modifications a dependence-graph can be used to evaluate the performance of TESLA, a very efficient MAC-based scheme. Dependence-graphs also provide insights into constructing and optimizing hash-chained schemes.

1 Introduction

Beside ensuring the privacy of data sent in the multicast stream, data authenticity is also a major security concern from a user point of view. (For example, a user does not want to receive stock quotes altered by some malicious parties, or receive some offensive materials). The receiver needs assurance that the data originated from the purported sender and was not modified on the way, even when none of the other receivers of the data is trusted.

The scenario of interest is one on the Internet with a single-source sending a multicast stream of packets to a large number of recipients. The lifetime of this communication could be very long during which recipients join and leave frequently. Examples of this include stock quotes distribution and news/video broadcasting over the Internet. Excluding the works on unconditionally secure schemes, there are two lines of approaches — one uses MAC with asymmetric property and the other uses signature amortization. The latter is the focus of this paper.

Authentication between two parties is usually done by sharing a secret key $k$ between the sender and receiver. When sending a message $M$ the sender computes a keyed hash function MAC (Message Authentication Code), $MAC = H_k(M)$ and appends it with the message. This MAC can be generated and verified quickly but it cannot provide source authentication in the case of multicast. The problem is that any receiver having the shared group key can forge data with a valid MAC and hence impersonate the sender.

In order to extend MAC to multicast settings, Canetti et al. relaxed the security requirements of multicast authentication and introduced the notion of asymmetric MAC or multicast MAC in [2]. This scheme can provide security against a collusion of $w$ receivers. Although asymmetric MAC is optimal for small $w$ (in term of message length) over the class of MAC-based approaches [1] and has the desirable property that each packet in the stream can be authenticated immediately without depending on other packets, the communication overhead of asymmetric MAC is unreasonably large and its initial key distribution is also an overkill. Hence, the MAC-based approach does not seem to be a sufficiently good solution to the problem of multicast authentication. Although Perrig et al. developed a scheme called TESLA [5, 6] which uses MACs but does not seem to have the high overhead problem as in asymmetric MAC\(^1\), we will see later using the notion developed in this paper that TESLA is in fact a derivative of signature amortization.

As an alternative to MAC, digital signatures can be used. Suppose the sender has a secret signing key and each of the receivers has the corresponding public key. Each message sent is signed by the sender using its private key and the receivers can then verify the integrity of messages using the sender’s public key. This method provides very good source and content authentication — no coalition of receivers can forge a message/signature pair that will fool another receiver. It also provides non-repudiation. However it has high overhead, in terms of time to sign and verify, and bandwidth. In streaming data applications like video broadcast, this “sign-each” approach is an overkill solution and not practical. As a result, several chained-hash authentication schemes [3, 4, 6, 7] were proposed to mitigate this overhead by amortizing a single digital signature over several packets. We focus in this paper the analysis of this line of solutions.

\(^1\)TESLA uses an additional time synchronization assumption to achieve a lower bound beating the bounds developed by Boneh et al. [1] who used standard assumptions.
packets is divided into a number of blocks, each made up of contiguous packets. The digital signature algorithm only needs to sign a single packet within this block. The rest of the packets are linked/chain ed to it (through hash concatenation) in such a way that allows recipients to verify that these messages originated from the sender they claim to be. The computation and communication overheads of signature are thus amortized over all packets in the block. Usually hashes can be generated more efficiently and have smaller sizes than digital signatures, the chained-hash signature approach is thus a reasonable solution to stream or multicast authentication. In fact, Boneh et al. [1] showed that it is not possible to build an efficient (in terms of communication overheads) collision resistant multicast authentication scheme without relying on digital signatures. Their work justified the importance of hash-chained signature schemes as a solution to the problem of multicast authentication.

Although the chained-hash signature approach seems to be a reasonable solution to multicast authentication, how to construct the hash chaining topology, that is, in what way the packets should be linked, to give an optimal 2 and bandwidth-efficient scheme remains an open problem. The existing constructions are either not robust to packet loss or have large overheads. Some even have a high receiver delay and/or require a large receiver buffer size. Poor loss-tolerance is not acceptable in Internet multicasts because the underlying network can only provide best-effort delivery. Although there are some schemes [6, 4] which have improved robustness against loss and use reasonable overheads, their performances could vary widely from one set of parameters to another. Besides, there is no effective way of choosing these parameters. Without doing a complete evaluation over the whole parameter space or using a trial-and-error approach, it is not possible to find the optimal set of parameters for a particular scenario. We argue that the main problem is the existing schemes were constructed in an ad hoc or trial-and-error manner and there is no systematic way to analyze and study the topology of hash chaining in those schemes, which is essential to design considerations.

In this paper, we propose the notion of dependence-graph as a framework to analyze the hash-chained signature schemes, which is essential to evaluating the metrics of different hash-chained schemes and avoiding the shortcomings of existing schemes in future designs, and leads to insights into design tradeoff and parameter optimization in these schemes. The notion of dependence-graph bridges the gap between the signature amortization schemes and the resource-rich graph theory. This lays the foundation for the application of algorithms and techniques in graph theory (such as dynamic programming, signal-flow graph, etc.) to the constructions and optimizations of hash-chaining topologies in signature amortization schemes. In this paper, we use dependence-graphs to analyze the performance metrics like authentication probabilities, communication overheads, delays and receiver buffer sizes, of existing hash-chained signature schemes. With slight modifications, we show how dependence-graphs can be used to analyze TESLA [5] too. We also show how dependence-graphs can potentially be used for the constructions and optimizations of future hash-chained schemes.

In the next section, we discuss some of the multicast authentication schemes which will be analyzed in this paper. We give the definition and notion of dependence-graph in Section 3. Then we give the analyses for different schemes together with the network model in Section 4. In Section 5, we present how dependence-graphs can be extended to facilitate designs and parameter selections in different scenarios. Finally, we conclude in Section 6 with a discussion on future works.

2 Multicast Authentication Schemes

2.1 MAC-based Schemes

TESLA. Perrig et al proposed TESLA [5, 6] which is based on symmetric-key MAC and can tolerate essentially any random loss. The basic idea is that different MAC keys are used to create the MACs for messages sent in different time-slots during a communication session. After the messages are sent, the corresponding MAC keys are then broadcast to all the receivers after a sufficiently large delay $T_{\text{disclose}}$, which will then be used by the receivers to verify the previously received messages. The MAC used is the same as those used in the two-party case. To ensure that the disclosed key will not be used to create a forgery, all messages received after the predefined key disclosure time will be dropped. Hence in order to keep this packet dropping low, it is necessary to have a sufficiently large $T_{\text{disclose}}$, to account for the network delay and its variation/jitter. To provide a way for the receivers to verify the MAC keys and to synchronize the time between the sender and receiver, the sender creates and sends to each receiver at the start of communication a “signed” bootstrap packet carrying timing information and commitment to a key chain in which each key is generated by a pseudo random function using its subsequent key as input. This provides robustness against packet loss because a lost MAC key can be generated at the receiver from any one of its subsequent MAC keys. The security is ensured by the difficulty of reversing the pseudo random function. This scheme has a small communication overhead and is robust against packet loss, but it requires that the sender and receivers synchronize their clocks within a certain margin. In some settings this time synchronization assumption may not be met. On the other hand, TESLA is similar to the chained-hash signature scheme. Instead of linking packets together, TESLA links the MAC keys together through hashing and signs the first MAC key disclosed to the receiver using digital signature. We will show, using appropriate network delay model, TESLA can be represented and analyzed by a dependence-graph.

2.2 Signature-based Schemes

In normal digital signatures, messages are usually hashed down to smaller fixed-sized messages before signing. In hash-chaining, a stream of packets is divided into a number of blocks. Within each block, the hash of each packet is appended to some other packets, which in turn generate new hashes appended to other packets. This hash-and-concatenate process continues until it reaches a single packet ($P_{\text{sign}}$), usually the first or last packet in the block, which is signed by the signature algorithm. Thus the computational load of the signature generation and verification is amortized. Some schemes like [7] require each packet to carry its own authentication information (signature and hashes of the neigh-
boring packets) so that each of them can be individually veri-
ifed at the receivers. The communication overhead in these schemes can be very significant. If the condition on individ-
ual packet verifiability is relaxed, then the communication overhead can be reduced substantially. In schemes using this approach [3, 4, 6], veriﬁcation of each packet is not guaran-
teed in the presence of loss, but instead it is assured that this can be done with a certain probability. The communication overhead is amortized in these schemes but the robustness of these schemes against packet loss varies depending on the topology of hash embedding.

Rohatgi’s. Gennaro and Rohatgi proposed the ﬁrst hash-
chained authentication scheme in [3]. Their scheme is very simple but does not tolerate loss. Even missing a single packet can break the chain, which makes it impossible to verify the authenticity of packets after the break point. The basic idea is as follows: For a block size of 3, a hash of the packet \( P_2 \) is appended to \( P_2 \) whose hash (hash of \( P_2 \) is in-
cluded in the hash computation of \( P_3 \)) is in turn appended to \( P_1 \). \( P_3 \) is signed and appended with the digital signature. In this way, non-repudiation is achieved for all the three pack-
ets. A major contribution of the authors to hash-chained authen-
tication scheme is that they proved its security.

Authentication Tree. Wong and Lam give an individu-
The hashes of packets in a block form the leaf nodes of a tree.
For other nodes, the parent nodes are computed as the hashes of
their children. A signature is computed at the root. This scheme has no receiver delay and is robust against packet loss, however it suffers from a high amount of overhead.

EMSS. To overcome the drawback (poor loss tolerance) of Rohatgi’s scheme, EMSS proposed by Perrig et. al. [6] stores the hash of each packet in multiple locations and app-
ends multiple hashes in the signature packet, which is the last packet in a block. The authors used a scheme in which each packet appends its hash to next and the second next packet. The signature packet that contains the hashes of the ﬁnal few packets along with a signature is sent at the end of
the block to authenticate all the packets in the block. EMSS can tolerate packet loss with relatively low overhead at the cost of delayed veriﬁcation - receiver delay. No formal anal-
ysis on EMSS (which was considered as an open problem by
the authors) was given in [6] but we will provide a potential analytical solution using dependence-graph.

Augmented Chain (AC). Since most of the packet loss on the Internet is bursty in nature, Golle et. al. proposed the augmented chain (AC) technique [4] whose goal is to provide stream authentication in the face of a single burst of loss. Like EMSS, this scheme uses a signature on the last packet and the hash of each packet is placed in several other packets in the stream. AC has two integer parameters \( a \) and \( b \), denoted by \( C_{a,b} \) and its hash chain is constructed in two phases as follows:

• In the ﬁrst phase, a chain is formed from a subset of packets in a block in which the hash of each packet \( P_i \) is appended to \( P_{i+1} \) and \( P_{1+a} \).

• In the second phase, the rest of the packets are inserted so that \( b-1 \) additional packets are inserted between each pair of consecutive packets in the original chain. The hash of these newly inserted packets are placed among themselves and in one of the two original packets

now bounding themselves such that each of these newly added packets is always linked to two other packets.

3 Dependence Graphs

In this section, we give the deﬁnition of a dependence-
graph which we will use to analyze the chained-hash authen-
tication schemes. We will also see how some of the met-
rics of the authentication schemes can be easily determined once these schemes are represented as dependence graphs.

We will also demonstrate how the dependence-graph analy-
sis can be extended to schemes which are not chained-hash based, specifically TESLA [5].

Definition 1 Let \( \{P_1, \ldots, P_n\} \) be a block of contiguous
packets in a stream. A dependence-graph is an acyclic la-
beled directed graph \( \mathcal{G} = (V, E, L) \) where the set of vertices
is the set of packets, \( V = \{P_i : 1 \leq i \leq n\}, E \subseteq V \times V \)
is the set of edges, \( L = \mathbb{Z} \) is the set of labels \(^5\) and the follow-
ing properties hold:

• There is a distinguished root vertex \( P_{\text{sign}} \), usually \( P_1 \) or \( P_n \), where the signature applies.

• An edge \( e_{ij} = (P_i, P_j) \in V \times V \) from \( P_i \) to \( P_j \) exists in \( E \) if and only if the following dependence relation (denoted by \( P_i \hookrightarrow P_j \)) holds:

\[ P_j \text{ can be authenticated by a receiver, then } P_i \text{ can also be authenticated by the same receiver using the information carried by } P_i. \]

• Every vertex \( P_i \) can be reached by the root through at
least one path.

• The label \( l_{ij} \in L \) on an edge \( e_{ij} \in E \) is the difference in the sequence number between \( P_i \) and \( P_j \), i.e. \( i - j \).

Depending on the schemes, \( P_{\text{sign}} \) could be the ﬁrst [3]
or the last [4, 6] packet in a block (In [7], every packet in
the block is \( P_{\text{sign}} \)). In our analysis, we assume that \( P_{\text{sign}} \)
can always be received by the receiver, otherwise all other packets in the same block cannot be authenticated. This can be easily achieved by sending it multiple times or applying it to multiple packets. In the hash-chaining techniques we
studied, a dependence relation between \( P_i \) and \( P_j \), denoted by \( P_i \hookrightarrow P_j \), also means that the hash of \( P_j \) is appended to
\( P_i \). This is how the dependence relation between \( P_i \) and \( P_j \)
is implemented. The case in TESLA is a bit different, the
dependence relation \( P_i \hookrightarrow P_j \) exists if the MAC key carried
by \( P_i \) can be used to authenticate the contents of \( P_j \). The
property that every vertex in \( \mathcal{G} \) must be reachable by \( P_{\text{sign}} \)
is essential because if there is no path between any vertex and
the root vertex, then the former cannot be authenticated even
in the absence of packet loss. However, if a probabilis-
tic method is used to place the edges in a dependence-graph, it is possible that some of the vertices in the graph cannot
be reached from the root vertex. In practice, the number of
such vertices is negligibly small. The dependence-graphs of
the schemes discussed in the previous section are depicted in
Figure 1. The efﬁciency of a multicast authentication scheme
can be evaluated using the metrics of authentication proba-
bility, communication overhead, receiver delay and receiver
buffer size. We will see how these metrics of a particular

\(^4\)In [4], the authors call the scheme \( C_{a,b} \). We use \( b \) here to replace \( p \) to avoid confusion with packet loss rate which we use \( p \) to denote.

\(^5\)\( \mathbb{Z} \) is the set of integers.
The authentication scheme can be obtained from its dependence-graph.

**Authentication probability.** Authentication probability for a particular packet $P_i$ is defined as $q_i = P\{P_i \text{ is verifiable} | P_i \text{ is received}\}$. For any particular scheme, the minimum authentication probability $q_{min}$ is a more appropriate performance metric and is given by:

$$q_{min} = \min_{P_i \in \{P_1, ..., P_n\}} \{ P\{P_i \text{ verifiable} | P_i \text{ received}\} \}$$

Whenever a packet $P_i$ is received, it can be authenticated only if the following conditions are satisfied:

1. There exists at least one path in the dependence-graph from $P_{sign}$ to $P_i$ along which all the packets corresponding to the vertices on the path are received or will be received. We denote the probability that this condition is satisfied by $\lambda_i$.

2. The authentication information derived from these received packets is valid. We denote the conditional probability that this condition is satisfied provided that condition (1) is satisfied by $\xi_i | \lambda_i$.

In most of the cases (for examples, [3, 4, 6]), the second condition can always be satisfied once the first condition is fulfilled, that is, $\xi_i | \lambda_i = 1$. However in TESLA [5], the MAC key for $P_i$ could have been received (condition (1) is satisfied), but $P_i$ still cannot be authenticated if it is received after the arrival of the MAC key$^6$ (condition (2) is not satisfied).

For the same authentication scheme, each packet in a block has a different authentication probability and this may vary widely from packet to packet depending on how many copies, how many different locations, and in what way the packet places its hash within the block. Of course, the loss pattern of the channel can affect this too. Some schemes have a smaller variance of authentication probability compared to others. In fact, these can be explained through studying the topologies of the corresponding dependence-graph. In general, with an independent random loss pattern, the greater the number of paths from $P_{sign}$ to $P_i$ or the smaller the number of vertices traversed by each of these paths from $P_{sign}$ to $P_i$, the greater the probability that $P_i$ can be authenticated ($q_i$). The spatial distribution of all the possible paths from $P_{sign}$ to $P_i$ can also affect $q_i$. For example, if a large fraction of these paths go through the same vertex, it is less probable that the authentication of $P_i$ can tolerate more loss due to a lower degree of diversity. On the other hand, if there exists only a single path from $P_{sign}$ to $P_i$, it is obvious that $q_i$ is inversely proportional to the distance that $P_i$ is from $P_{sign}$.

To minimize the variance of the authentication probabilities for different packets, we should introduce more paths for a packet which is farther away from $P_{sign}$, through storing its hash in more locations in a dispersed manner.

In order to quantitatively evaluate the authentication probability for a particular packet $P_i$, we first give a definition of path in the dependence-graph as follows:

**Definition 2** In a dependence-graph $G = \langle V, E, L \rangle$, for some set $\theta(P_i, P_j) \subseteq \forall \{P_i, P_j\}$, we say $\theta(P_i, P_j) \in \Theta(P_i, P_j)$ if and only if there exists a path from $P_i$ to $P_j$ traversing all vertices in the set $\theta(P_i, P_j)$. That is, there always exists a path in $G$ from $P_i$ to $P_j$ traversing some vertices in $\Theta(P_i, P_j)$.

For any set of packets $A$, we denote $S(A)$ as the event that there is at least one packet loss among the packets in $A$. Then $\lambda_i$ can be given as:

$$1 - \lambda_i = P\{\text{Each path between } P_{sign} \text{ and } P_i \text{ has at least one packet loss}\}$$

$$= P\{\bigwedge_{\theta_s(i) \in \Theta(i)} S(\theta_x(i))\}$$

In order to evaluate $P\{\bigwedge_{\theta_s(i) \in \Theta(i)} S(\theta_x(i))\}$, it is necessary to know the exact topology of the dependence-graph. We will see in the next section that many of the existing schemes like EMSS and AC have a regular topology (with periodic structures) and their $\lambda_i$’s can be evaluated recursively. For here, we can derive an upper and a lower bound for $\lambda_i$ based on the best and the worst case topologies. The worst case scenario is that all $\theta_x(i)$’s in $\Theta(i)$ have no overlap. The worst case scenario is when $\theta_x(i)$’s in $\Theta(i)$ intersect each other and the maximum possible coverage of vertices. Let the elements in $\Theta(i)$ be arranged in an order such that $|\theta_1(i)| \leq |\theta_2(i)| \leq \ldots \leq |\theta_x(i)| \leq \ldots \leq |\theta_{|\Theta(i)|}(i)|$. Denote the best and the worst case topologies by $b.c.t.$ and $w.c.t.$, respectively, then

$$P\{\bigwedge_{\theta_s(i) \in \Theta(i)} S(\theta_x(i))\}_{b.c.t.}$$

$$= \prod_{\theta_s(i) \in \Theta(i)} P\{S(\theta_s(i))\}$$

$$\geq \left[ \min_{\theta_s(i) \in \Theta(i)} \{ P\{S(\theta_s(i))\} \} \right]^{\Theta(i)}$$

$^6$For the sake of convenience, we may denote $\theta_s(P_{sign}, P_i)$ by $\theta_s(i)$ and $\Theta(P_{sign}, P_i)$ by $\Theta(i)$. 

.$^6$It is possible to have out-of-order delivery on the Internet.
\[
\Pr\{\bigwedge_{i=0}^{\Theta \{i\}} S\{\Theta \{i\}\}|w.c.t.\} \\
= \Pr\{S\{\Theta \{i\}\} \cap \{\Theta \{i\}\} |w.c.t.\} \\
\times \Pr\{S\{\Theta \{i\}\}|w.c.t.\} \\
= \Pr\{S\{\Theta \{i\}\}|w.c.t.\} \\
\times \Pr\{S\{\Theta \{i\}\}|w.c.t.\}
\]

Therefore,
\[
1 - \Pr\{S\{\Theta \{i\}\}|w.c.t.\} \\
\leq \lambda_i
\]

where \(\min\{\Theta \{i\}\}\) is the number of vertices traversed by the shortest path from \(P_{sign}\) to \(P_i\). The bounds in Equation(1) are not tight. Tighter bounds can be found if more information of the topology is known.

**Communication overhead.** Communication overhead is the size of authentication information (including signatures, hashes and MAC keys) that each packet carries. For any packet \(P_i\), the number of hashes from other packets it carries is equal to its out-degree \(\partial(P_i)\) in the dependence-graph. \(P_{sign}\) carries a signature in addition to these. The average number of hashes \(m\) per packet is then given by:
\[
m = \frac{1}{n} \sum_{i=1}^{n} \partial(P_i) = \frac{1}{n} \sum_{P_i \in \mathcal{G}} \partial(P_i) = \frac{1}{n} |\mathcal{E}|
\]

where \(\mathcal{E}\) is the set of edges in the dependence-graph \(\mathcal{G}\) = \((V, E, \mathcal{E})\) and \(n\) is the total number of packets in the block.

Let \(p_s\) be the loss probability of \(P_{sign}\) and suppose \(P_{sign}\) is transmitted \(\frac{1}{p_s}\) times to ensure that it can be received with a reasonably high probability. If the length of the signature and hash are \(l_{sign}\) and \(l_{hash}\) respectively, then the average communication overhead \(d\) per packet is:
\[
d = \frac{1}{n} (l_{sign} + l_{hash} \times |\mathcal{E}|)
\]

In order to minimize the communication overhead per packet, the construction must keep the total number of edges in the dependence-graph as small as possible while achieving the desired minimum authentication probability.

**Receiver delay.** Receiver delay measures the time delay a packet waits after it has been received but before all the information necessary for its authentication is received. It consists of a deterministic \(t_d\) and a random \(t_r\) component. The deterministic component is positive if the hash of a packet is placed in the subsequent packets or the signature [6, 4] is placed at the end of a block. It is also positive in TESLA[5], which is \(t_{disclose}\). The random component exists in networks which may provide out-of-order deliveries and its value depends on the network delay distribution. In general, for a block size of \(n\) and a packet transmission time of \(T_{transmit}\), the maximum deterministic delay of \(P_i\) is:

\[
t_d(P_i) = \begin{cases} 
T_{disclose}, & \text{if } i = 0 \\
(n-i)T_{transmit}, & \text{otherwise}
\end{cases}
\]

Suppose \(f_t(t)\) is the pdf (probability density function) of the random receiver delay which is i.i.d (identical and independently distributed) for each packet, then the worst case total receiver delay is a random variable given by:
\[
D_{worst} = t_d(worst) + t_r(P_k) - t_r(P_i)
\]

where \(k\) and \(i\) are the sequence numbers of two arbitrary vertices in \(\mathcal{E}\). The pdf of \(D_{worst}\) can then be easily determined from the joint distribution of the random delays, that is,
\[
f_{worst}(t_1, t_2) = f_{t_r}(t_1)f_{t_r}(t_2).
\]

**Receiver buffer size.** There are two types of receiver buffering required, one for messages and the other for hashes. If packets in a scheme rely on subsequent packets to authenticate themselves, then a message buffer is needed. Hash buffer is used to store the hashes necessary for the authentication of the subsequent packets. Neglecting the effects of random network delay, the hash or message receiver buffer size required for a particular scheme is equal to:
\[
max_{i,j|e| \in \mathcal{E}} \{max(l_{hash}, 0)\}
\]

**[Example]** We take the Rohatgi’s scheme [3] with a block size of \(n\) as an example. Assume that each packet is lost independently with a probability \(p\). For any particular \(P_i\), the hash chain is broken and \(P_i\) cannot be authenticated if any of the \((i-2)\) packets in between \(P_i\) and \(P_{sign}\) (the signature packet) is missing. Furthermore, \(P_n\) has the smallest authentication probability. Therefore, the authentication probability \(q_i = \lambda_i = (1-p)^{i-1}\) and \(q_{min} = (1-p)^{n-2}\). There are \((n-1)\) edges in the dependence-graph, therefore the communication overhead is \(2(n-1)\) hashes per packet. Each packet can be authenticated as soon as it is received without having to wait for the arrivals of subsequent packets, therefore the deterministic receiver delay \(t_d(worst) = 0\). All the edges exist between consecutive nodes, thus 1 hash buffer and no message buffer is needed at the receiver.

### 3.1 Design Tradeoff

The dependence-graph exhibits design tradeoff among the authentication probability, overhead, receiver delay and buffer size (Buffer size is closely proportional to receiver delay). In order to increase the authentication probability of a construction, we need to add path from \(P_{sign}\) to \(P_i\) or shorten the existing paths. It is obvious that adding paths adds edges and hence overhead. Besides, the new path added may have a larger receiver delay. On the other hand, a reduction of receiver delay can reduce the number of possible paths from \(P_{sign}\) to \(P_i\) and hence lowering the authentication probability. Thus a practical way to design an optimal multicast authentication scheme in any particular network with packet loss rate \(p\) is to minimize the number of edges of the dependence graph subject to the constraint that there is at least a certain number of paths from \(P_{sign}\) to each \(P_i\) (This number of paths depends on the loss pattern). If zero-receiver-delay is required, \(P_{sign}\) must be the first packet and
additional constraints on edge placements are required (e.g. an edge can only be placed from a vertex closer to $P_{\text{sign}}$ to a vertex farther away from $P_{\text{sign}}$ and not in the reversed way).

### 3.2 Dependence-graph of TESLA

The notion of dependence-graphs can be extended to analyze TESLA by making slight modifications in its construction. Let the disclosure delay be $T_{\text{disclose}}$. To construct a dependence-graph for TESLA, each packet is considered as two vertices in the dependence-graph — one for the message part and the other for the MAC key part. An example of TESLA dependence-graph is shown in Figure 2. We denote the message and key nodes by $P_i$ and $K_{i,a}$ (the MAC key for $P_i$ which is carried by $P_{i+a}$) respectively. $P_{\text{sign}}$ is assigned to the bootstrap packet (signed by the sender). Once the bootstrap packet is received, all the MAC keys are authenticated. Hence, an edge is added from each $K_{i,a}$ to a particular packet $P_i$. For each $i$, $i \geq j$. Unlike the normal dependence-graph, this one has no label.

Let the number of packets sent during the whole lifetime of the hash chain be $n$. Denote the end-to-end delay of $P_i$ by $t_i$ which follows some random distribution. MAC key for any packet $P_i$ can be obtained as long as any one of $\{K(j,a) : j \geq i\}$ can be received, therefore

$$\lambda_i = 1 - Pr\{\text{all packets in } \{K(j,a) : i \leq j \leq n\} \text{ are lost}\}$$

Assuming independent packet loss with probability $p$, then

$$\lambda_i = 1 - p^{n+1-i} \quad \text{and} \quad \xi_{i|\lambda_i} = Pr\{t_i \leq T_{\text{disclose}}\}$$

The authentication probabilities are then given by:

$$q_i = \lambda_i \xi_{i|\lambda_i} = [1 - p^{n+1-i}] \times Pr\{t_i \leq T_{\text{disclose}}\}$$

$$q_{\text{min}} = (1-p) \times Pr\{t_i \leq T_{\text{disclose}}\}$$

Receiver delay of TESLA is $T_{\text{disclose}}$ and the message buffer size required is $\rho T_{\text{disclose}}$ where $\rho$ is the message arrival rate at the sender.

### 4 Analyses

#### 4.1 Network Models

In this paper, we use a simple independent random loss model for analysis, that is, each packet is lost independently with a certain probability $p$. When a packet traverses through a large network like the Internet, it passes through a large number ($N$) of routers, each incurring a random queuing and servicing delay with an overall delay $D_i$. If we assume $N$ is sufficiently large, each of these $D_i$ can be considered as i.i.d. (identical and independently distributed) and we can then use Gaussian approximation to model the end-to-end delay ($D_{d2e}$) distribution.

The central limit theorem tells us that the probability distribution of $D_{d2e} = \lim_{N \to \infty} \sum_{i=1}^{N} D_i$ approaches to a Gaussian distribution $N(\mu, \sigma^2)$ where $\mu$ is the mean end-to-end delay and $\sigma^2$ is the end-to-end delay variance/jitter, and:

$$Pr\{D_{d2e} \leq d\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{d-\mu}{\sigma}} e^{-x^2/2} dx$$

#### 4.2 Analyses

We evaluate and compare the efficiency of 5 schemes — Rohatgi’s[3], Authentication Tree[7], TESLA[5], EMSS[6] and AC[4]. We have given the analysis of Rohatgi’s scheme as an example in Section 3 and the analysis of Authentication Tree is trivial. We will see that the analyses of EMSS and AC are very similar in which the authentication probabilities can be evaluated using recurrence relations or difference equations. In fact, any dependence-graph with periodic structures like EMSS and AC can use recurrence relations to evaluate its authentication probability over an independent random loss channel.

An analysis on TESLA has been given in Section 3.2. Incorporating the Gaussian delay model (Equation(5)) gives:

$$q_i = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{(1-p)T_{\text{disclose}}} e^{-x^2/2} dx$$

$$q_{\text{min}} = \frac{1-p}{\sqrt{2\pi}} \int_{-\infty}^{(1-p)T_{\text{disclose}}} e^{-x^2/2} dx$$

where $\mu, \sigma \approx \alpha T_{\text{disclose}}$ for some $0 \leq \alpha \leq 1$ and $\sigma$ are respectively the mean end-to-end delay and delay variance of the network.

For the sake of simplicity, we introduce a notation $E_{m,d}$ for the EMSS scheme with each packet depending on $m$ previous packets each of which is separated by $(d-1)$ packets. We use the same notation as in [4] for the AC scheme, that is $C_{a,b}$. Since both EMSS and AC sign the last packet in a block, it is inconvenient in the analysis. We make a reassignment of packet indices in a reversed order so that the signature packet is $P_1$ and the packet sent earlier has a higher index. The purpose of such rearrangement is just for convenience and it does not affect the analytical results.

In the EMSS shown in Figure 1, each packet relies on the previous two packets to authenticate itself, that is, an $E_{2,1}$...
scheme. We let \( n \) be the block size, the authentication probability \( q_i \) is the same as \( \text{Prob} \{ 3 \text{ at least one path between } P_i \text{ and } P_{\text{sign}} | P_i \text{ is received} \} \). Based on this, \( q_i \) of EMSS can be found easily from the following recurrence relation:

\[
q_i = 1 - [1 - (1 - p)q_{i-1}] [1 - (1 - p)q_{i-2}]
\]

i.e. : \( q_1 = q_2 = q_3 = 1 \) (8)

where \( p \) is the packet loss rate. This recurrence relation is not in the linear form so no closed form solution exists. But if we can do some substitutions of terms, we can still derive a lower bound in closed-form for \( q_{\text{min}} \). For example, EMSS \( E_{2,1} \) has \( q_{\text{min}} \geq 1 - p(1-p)^{-1} \).

This recurrence relation can be generalized for any periodic structure as follows:

\[
q_i = 1 - \prod_{a \in \mathcal{A}} [1 - (1 - p)q_{i-a}]
\]

i.e. : \( q_i = 1, \forall i \leq \max_{a \in \mathcal{A}} \{ a \} \) (9)

where \( \mathcal{A} \subset (-n, n) \) is the set of integers which are differences between \( P_i \)'s index and the indices of packets it relies on. For example, in a scheme each packet \( P_i \) puts its hash into the packets \( P_{i-a}, P_{i-b}, P_{i-c} \), then \( \mathcal{A} = \{ a, b, c \} \). The elements of \( \mathcal{A} \) could be negative since a packet may put its hash into packet which is farther away from \( P_{\text{sign}} \) than itself.

There are two levels of periodic structures in an augmented chain \( [4] \). To facilitate analysis, we label packet \( P_i \) in a \( C(a, b) \) AC scheme as \( P(x,y) \) where \( x = (i-1)/(b+1) \) \( \in [0, (n-1)/(b+1)] \) and \( y = i \text{ mod } (b+1) \) \( \in [0, b] \). Let \( q(x,y) \) denote the corresponding \( q_i \), then the two levels of recurrence relations are as follows:

\[
q(x,y) = 1 - [1 - (1 - p)q(x-1,y)] [1 - (1 - p)q(x-a,y)],
\]

i.e. : \( q(x,y) = 1, x \leq a, \quad x \equiv 0 \text{ mod } (b+1) \)

\[
q(x,y) = 1 - [1 - (1 - p)q(x,y+1)] [1 - (1 - p)q(x,0)],
\]

i.e. : \( q(x,b) = 1 - [1 - (1 - p)q(x+1,0)] [1 - (1 - p)q(x,0)] \),

otherwise (10)

In AC, \( q_i \)'s in the first level are solved first, which are then used as initial conditions for solving the \( q_i \)'s in the second level.

### 4.3 Numerical Results

Depicted in Figure 3 is a plot of the minimum authentication probability \( q_{\text{min}} \) of a TESLA scheme with block size of 1000 packets using \( T_{\text{disclose}} = 1s \) against the network end-to-end delay \( \mu = \alpha T_{\text{disclose}} \) and its variance \( \sigma^2 \). As expected, \( q_{\text{min}} \) drops as either \( \mu \) or \( \sigma \) increases. Figure 4 shows the plot of \( q_{\text{min}} \) against normalized key disclosure delay \( T_{\text{disclose}}/\sigma \) and packet loss rate \( p \) for the same TESLA scheme on networks with different end-to-end delay \( \mu \). As can be seen from these graphs, the performance of TESLA is quite robust to packet loss if \( T_{\text{disclose}} \) is chosen sufficiently large compared to \( \mu \) and \( \sigma \). Of course, the time at the sender and receiver need to be synchronized to provide this appealing performance.

We show in Figure 5 and 6 how the augmented chain schemes \( C(a,b) \) are affected by the parameters \( a \) and \( b \). In Figure 5, a fixed block size of 1000 is used in the calculation and the results are for packet loss rates at 0.1, 0.3 and 0.5. We can see that the minimum authentication probability \( q_{\text{min}} \) drops when either \( a \) or \( b \) decreases. For a fixed block size \( n \), when \( a \) increases, the number of packets in the first level directly linked to \( P_{\text{sign}} \) increases. When \( b \) also increases in this case, the depth of the first level of packets is smaller for a fixed \( n \), hence the minimum authentication probability of the first level packets increases. In fact, if the number of packets in the first level is kept constant (i.e. \( n \) varies with \( b \)), increasing \( b \) has little effect on \( q_{\text{min}} \), which is shown in Figure 6. We can see \( q_{\text{min}} \) is relatively insensitive to the variation of \( b \) if \( b \) is larger than a certain value. Because of this, AC provides an efficient way to insert new packets without degrading the performance of the scheme.

Figure 7 shows how \( q_{\text{min}} \) of an EMSS scheme \( E_{m,d} \) is affected by varying \( m \) and \( d \) using a block size (\( n \)) of 1000 packets. Compared to \( m \), the performance of EMSS is much less sensitive to changing \( d \), the separation between the hash links. We can see from the plots that the change in \( q_{\text{min}} \) is significant only when the change in \( d \) is more than 20% of \( n \). Receiver delay and buffer size are proportional to \( d \), the design with delay constraints is thus relatively simple for EMSS. It can also be seen that the performance of EMSS levels off when \( m \) is larger than a relatively small value, say 2-4. This is interesting because \( m \) is directly proportional to the packet overhead. The results demonstrate that EMSS can perform well with a relatively small overhead.

Then we can compare the performance of 4 specific schemes — Rohatgi’s, TESLA, EMSS \( E_{2,1} \) and AC \( C_{3,3} \). Figure 8 shows how \( q_{\text{min}} \) of different schemes vary with the packet loss rate \( p \) and the block size \( n \). In general, \( q_{\text{min}} \) drops as \( n \) increases because the depth of the corresponding dependence-graph increases as the total number of packets in a block increases. The figures show that the robustness of Rohatgi’s scheme against loss is incredibly low whereas the
other three have similar performance. Figure 9 shows a closer look on these three schemes as n varies. It can be seen that the performances of EMSS and AC are very close to each other. This could be partly explained by the observations in Figure 7. In both AC $C_{3,3}$ and EMSS $E_{2,1}$, each packet is linked to two other packets. The difference between them is just the way these two links are constructed, which can be roughly quantified by the parameter $d$ in EMSS. Figure 7 shows that the performance is relatively insensitive to $d$, hence this could explain why EMSS $E_{2,1}$ and AC $C_{a,b}$ have so similar performances. If the disclosure delay of TESLA is chosen to be sufficiently large to absorb all the effects of $\mu$ and $\sigma$, TESLA can out perform EMSS and AC. TESLA is also less sensitive to packet loss compared to the other two — EMSS and AC can out perform TESLA at small $p$, but at larger $p$ TESLA is significantly better. On the Internet such a high loss rate does not occur very often but delay jitter can be very large. Hence whether TESLA is optimal for use over the Internet is still in doubt since it is so strongly affected by the network delay and jitter. All the 3 schemes have performance which is not so sensitive to the change in $n$, this simplifies the designs. For the Authentication Tree scheme [7], each packet carries its authentication information and the authentication probability is thus not affected by the packet loss and hence is always 1.

A comparison on overhead and delay for different schemes is show in Figure 10. The hash-chained schemes have similar overhead and delay. Like TESLA, EMSS and AC require buffering at the receiver which is undesirable and is subject to Denial of Service attacks.

5 Design Considerations

After showing how dependence-graphs can be used to analyze the performance of a number of hash-chained schemes, in this section, we give insights into using dependence-graphs as a design tool for the hash-chained schemes. As mentioned earlier, the design objective of the hash-chained schemes is to construct a dependence-graph which has the minimum total number of edges and each vertex in it is reachable by $F_{\text{sign}}$ through at least a certain number of paths each having a pre-defined maximum length. Making the design more difficult is that the number of packets in a block over a fixed period of time is normally not fixed and online constructions are necessary.

In fact, nearly all of the existing graph algorithms can be used here. A relatively straight forward but heuristic way to construct dependence-graphs is by starting with a tree and then adding edges in each subsequent levels until the given constraints on authentication probabilities are all satisfied. But this method requires a lot of evaluation of $q_{\text{min}}$. On the other hand, the algorithms commonly used for spammer constructions in fault tolerant networks can be used.

We can also formulate the dependence-graph construction as a dynamic programming problem — Given a certain number of vertices, find the optimal policy which minimizes the total number of edges required while satisfying the constraints that $q_i$ is greater than certain design minimum for all vertices. The advantage of dynamic programming is that it can usually give a simple policy suitable for online constructions. Probabilistic methods may also be used. A simple method is that for each of the vertices, we construct an edge to each of the earlier vertices with a probability $p_x$.

6 Conclusions

In this paper, we introduced the notion of dependence-graphs to study multicast authentication schemes. Through the existing and well-known graph-theoretical framework, we can analyze and construct multicast authentication schemes in a systematic and efficient way. Although we originally targeted at chained-hash type of authentication schemes, we found that the notion of dependence-graphs can be applied to study MAC-based schemes like TESLA very effectively. In fact, a dependence-graph has many nice properties which allow us to optimize the design of an authentication scheme. As future works, we will use the dependence-graph and some of the existing graph algorithms to construct optimal schemes and investigate the design tradeoff. It is also interesting to extend the derivations to other loss models like the m-state Markov model.
Figure 7. Minimum authentication probability of EMSS, $E_{\text{min}}$, for different $n$ (the number of previous packets) and $d$ (the separation between these packets) at packet loss rates $p = 0.1$, $0.3$, and $0.5$.

Figure 8. Minimum authentication probability for different schemes at different (a) packet loss rate $p$ and (b) block size $n$.

Figure 9. Minimum authentication probability vs. block size $n$ for different schemes at $p = 0.1$, and $0.5$.

Figure 10. Overhead and delay for different schemes.

References


