Simulation-Based Performance Evaluation of Predictive-Hashing Based Multicast Authentication Protocol[1]

Seonho Choi
Department of Computer Science, Bowie State University, Bowie, MD 20715, U.S.A.
and
Hyeonsang Eom
School of Computer Science and Engineering, Seoul National University, Seoul, South Korea
and
Edward Jung
School of Computing and SE, Southern Polytechnic University, Marietta, GA 30067

Abstract

A predictive-hashing based Denial-of-Service (DoS) resistant multicast authentication protocol was proposed based upon predictive-hashing, one-way key chain, erasure codes, and distillation codes techniques [4, 5]. It was claimed that this new scheme should be more resistant to various types of DoS attacks, and its worst-case resource requirements were derived in terms of coarse-level system parameters including CPU times for signature verification and erasure/distillation decoding operations, attack levels, etc. To show the effectiveness of our approach and to analyze exact resource requirements in various attack scenarios with different parameter settings, we designed and implemented an attack simulator which is platform-independent. Various attack scenarios may be created with different attack types and parameters against a receiver equipped with the predictive-hashing based protocol. The design of the simulator is explained, and the simulation results are presented with detailed resource usage statistics. In addition, resistance level to various types of DoS attacks is formulated with a newly defined resistance metric. By comparing these results to those from another approach, PRABS [8], we show that the resistance level of our protocol is greatly enhanced even in the presence of many attack streams.

Key Words*: denial of service, network protocol, authentication, multicast, resource requirement, cryptographic hashing, simulation*

1. Introduction

For real-time streaming applications, a new multicast authentication protocol was proposed which shows a higher level of resistance to Denial-of-Service (DoS) attacks [4, 5]. It was claimed that the resource (CPU, buffer, and network bandwidth) usage level can be greatly reduced by utilizing predictive hashing (PH) technique.

Our scheme is based on a block-based approach where a ~~real-time data stream is divided into bl~~ocks of packets. In the predictive-hashing approach, packets in one block contain messages along with predictive authentication information required for authenticating the next block messages. Preliminary analysis on worst-case resource requirements conducted in our previous work [5] indicates that this new scheme consumes much less CPU and buffer space than one of the recently proposed denial-of-service (DoS) resistant multicast authentication schemes, pollution resistant authenticated block streams (PRABS) [8], and that its resistance to DoS attacks is greatly enhanced.

However, in the previous analysis, we derived various formulae, for estimating upper bounds on the resource requirements, in terms of coarse-level system parameters such as CPU times needed for performing erasure decoding, distillation decoding, signature verifications, etc. In addition, worst-case scenarios were assumed in analyses, which may yield too pessimistic estimation results that may not happen in real attack situations. Also, the analysis didn't provide a way to estimate the resource requirements for different attack types and/or different parameter settings. For example, by examining resource requirement changes for different block sizes (with the same bandwidth maintained for the data stream), we may have more insight on which block size we need to choose. The use of a smaller block size may lead to reduced resource usage compared to the use of a bigger block size. Also, using a smaller block size may loosen security condition under which it may be guaranteed that the worst DoS attack type cannot be launched, which the system designer may prefer. However, using a smaller block size means that the receivers will be more susceptible to messages losses due to bursty packet losses (from network congestion and/or from attack streams). Hence, the system designer needs to take these factors into consideration when determining values of the system parameters. Finally, to evaluate resistance level of our protocol against other approaches, a formal metric should be devised. By using a simulator we developed, it becomes possible to quantify resistance levels on different platforms.

We designed and implemented a simulator for the predictive-hashing based multicast authentication protocol. Multiple attack streams along with an authentic data stream may be generated and launched against a virtual receiver which is equipped with the predictive-hashing based protocol. Different attack types may be used in generating such attack streams with various system parameters such as block size, redundancy level for erasure encoding, message size, loss rate, packet (or block) period, and simulation duration, etc. The packets generated by the stream generators will be written to multiple files, and the system

[1] This work was supported by ARO grant 48575-RT-ISP.

parameters used by the stream generators are written to a separate file. These files are read by the virtual receiver process later and the packets will be fed as inputs to the packet processing object (called decoder) which performs authentication operations. One feature of this simulator is that it is platform independent, and the timing parameters such as block period or packet period may be specified in absolute time and will be enforced in any platform where the simulator is running. This simulator is written in Java.

By using this simulator, we may be able to obtain accurate information on resource usages by the protocol in a variety of attack scenarios with different parameter settings. The simulator also outputs detailed information on how much resources are used by each component in the protocol implementation. The type of DoS attack that may be launched varies depending upon the level of attack complexity. The attacker may simply generate packets randomly and launch an attack. Or, he/she may intercept authentic packets somewhere in the network and modify some fields in the packet and launch them against receivers. Alternatively, the attacker may intercept packets at one location in the network, reuse/modify some fields in the packets, and relay them to another location in the network, launching an attack with those relayed packets. This type of attack is named in this paper as a strong relay attack (SRA) and it will be addressed later on the effects of this type of attack and how to avoid such attack scenarios in PH-based approach. These various attack types may be specified as one of the input parameter to our simulator, and attack streams matching these specifications will be generated. Also, by using the simulator, we may formally define the concept of DoS resistance level (DRL) for different attack types as the number of attack streams the receiver may tolerate in terms of authentication throughput and memory requirement.

Preliminary information is given in Section 2. Section 3 provides a brief explanation on predictive-hashing based authentication protocol. Section 4 describes approaches we have taken in the design and implementation of the simulator. In Section 5, detailed simulation results are explained. Conclusions are presented in Section 6.

2. Preliminaries

Block: The original data (or message) stream at the sender side is divided into blocks, and each block data is packetized into the same number of packets. Each packet contains message portion (from original data stream) and additional information related to authentication may be attached. Block period, *p*, corresponds to a duration of time during which block packets are generated by the sender.

Erasure codes: This is one of the forward error correction (FEC) techniques to recover lost packets during transmission [6]. The encoder redundantly encodes information into a set of symbols. If the decoder (receiver) receives sufficiently many symbols, it can reconstruct the original information. An *(n, t)* erasure encoder generates a set of *n* symbols from the input. The decoder can recover all the original data as long as *n-t* symbols are available. *t* is named as a redundancy level.

Strong Threat Model: In this model, there is no limitation on attacker's capability:

- Packets may be eavesdropped, deleted, and spoofed (and sent).
- More than one of these attack operations may be combined to launch more powerful attacks against receivers. For example, packets may be first eavesdropped and deleted (blocked) so that receivers may not receive them. Based on the eavesdropped contents, newly spoofed packets may be sent immediately to the receivers. This attack scenario assumes that attackers can combine all of the above three operations at the same time.

Denial of service attacks

As in [6], an *attack level*, *f,* is introduced and used in this paper which defines the ratio of the bandwidth of injected invalid traffic to the bandwidth of valid traffic. For example, if an adversary injects 10,000 bytes of invalid data in one unit time while the sender is sending 1,000 bytes in a unit time, then the attack level is 10.

One-way accumulators

We can build a secure set membership operation by using one-way accumulators [1, 2]. There are several one-way accumulator schemes based on different cryptographic techniques. In distillation codes, Merkel hash trees [3] are used as one-way accumulators. When Merkel hash trees serve as one-way accumulators, the size of witnesses grows logarithmically with the size of the accumulated set.

3. Predictive Hashing Based Approach

We developed a new mechanism, which is based on *Predictive Hashing* (PH) and *One-way Key Chain* (OKC), to significantly reduce resource requirements at a receiver even in the presence of DoS attack packets flowing in. The basic idea of predictive hashing is that each block of packets conveys authentication information that will be used to authenticate the next block packets instead of sending authentication information within the same block as in previous approaches [6, 9, 10]. The PH technique allows receivers to save significant amount of buffer space since only authentication-related portions from each packet needs to be saved for future packet authentication, while the message portions of arrived packets are processed (or authenticated) immediately upon receipt. However, in our scheme, the sender needs to keep the message portions from two consecutive blocks in its buffer to calculate PH.

One-way key chain technique is already used in other contexts such as in one-time password [8], TESLA [11], etc. In our approach, the sender obtains a hash chain by applying hash operations recursively to some seed value, and obtained key values are assigned to blocks in backward order of their generation times. The sender uses the assigned key to calculate Message Authentication Codes (MAC) images of the prediction hashes/signature information for the next block, and attach them (along with other authentication related information) to the current

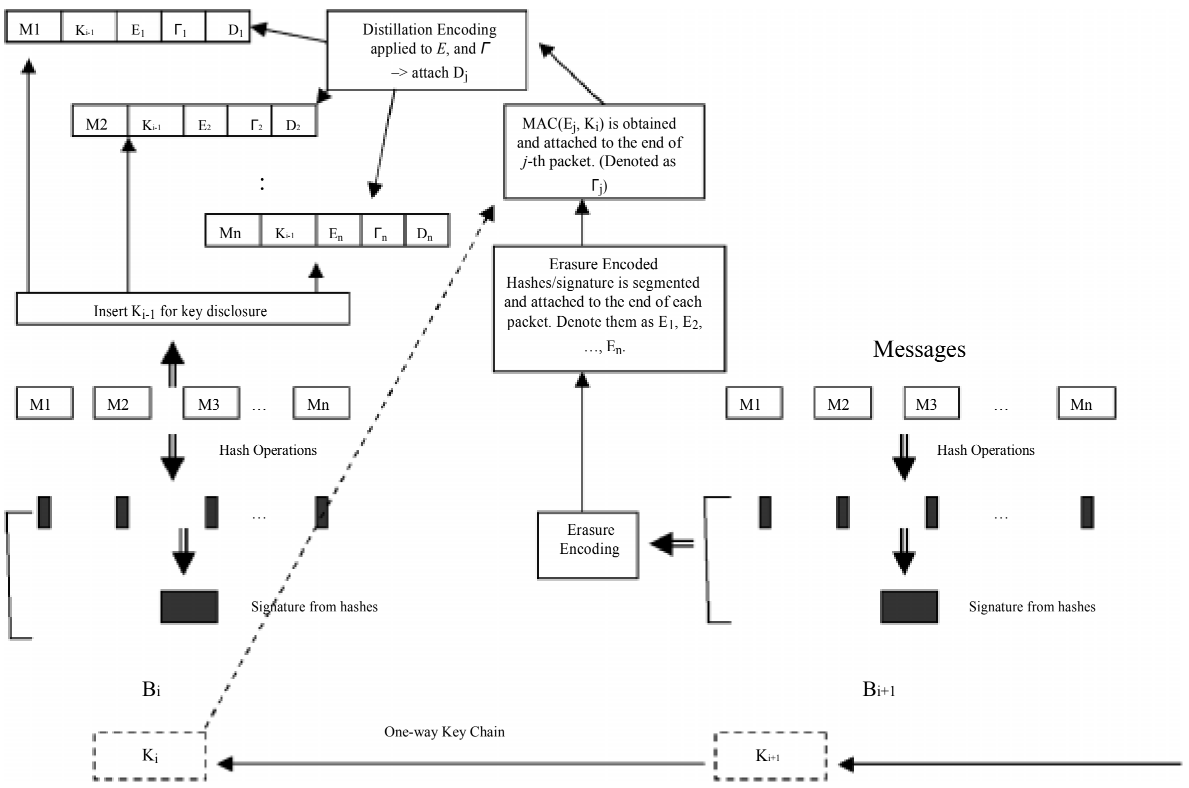

M1
K_{i-1}
E1
Γ1
D1
Distillation Encoding applied to E, and Γ –> attach Dj
M2
E2
Γ2
D2
MAC(Ej, Ki) is obtained and attached to the end of j-th packet. (Denoted as Γj)
Mn
En
Γn
Dn
Insert Ki-1 for key disclosure
Erasure Encoded Hashes/signature is segmented and attached to the end of each packet. Denote them as E1, E2, …, En.
Messages
M1
M2
M3
Mn
Hash Operations
Erasure Encoding
Signature from hashes
Bi
Bi+1
One-way Key Chain
Ki
Ki+1

Figure 1. Overview of our PH-based scheme at a sender

block packets. Also, each block packet reveals the key used in the previous block to let the receivers use it in authenticating the previous block packets (or partitions) without applying erasure decoding and signature verifications in most of the cases. These mechanisms are combined with erasure codes and distillation codes to develop a multicast authentication protocol which is very resistant to Denial-of-Service attacks and resource-efficient. Figure 1 shows an overview of our approach at sender side. The receiver side operation is the reverse of the process shown in Figure 1.

PH Decoding Algorithm at Receiver

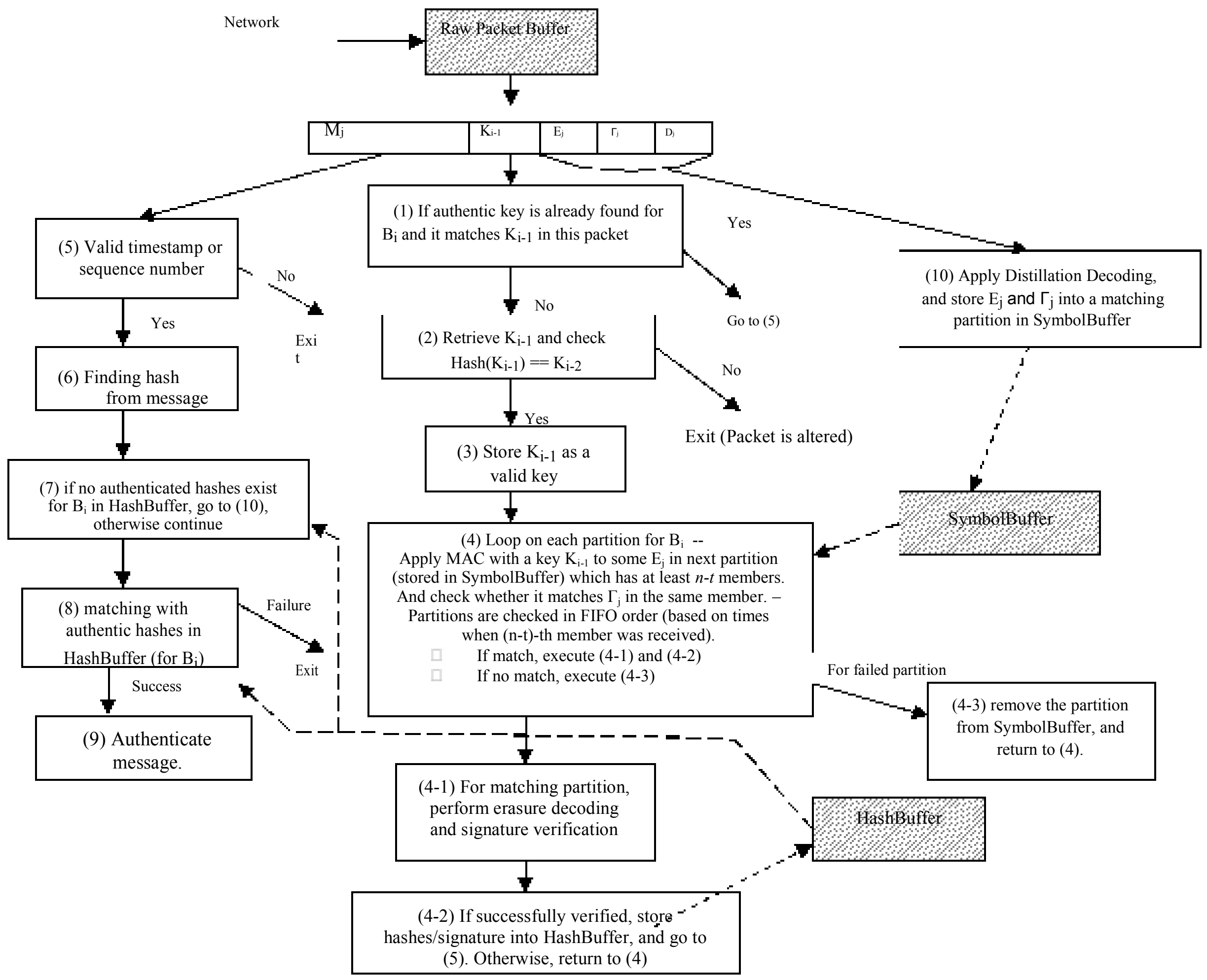
The receiver side algorithm is presented in Figure 2 in detail [4, 5].

4. Simulator Design and Implementation

The following are the goals of our simulator design:

- accurate measurement of resource usages at a receiver where the simulator is running: whatever computing platform the simulator is running, it should provide accurate measurements on resource usages (CPU time, memory, and bandwidth) as if it were acting as a receiver in a real network. To achieve this goal, the simulator is implemented in Java with a capability to adjust its execution scenario based upon timing parameters such as block period (or packet inter-arrival times).
- platform independence: this is achieved by implementing the simulator in Java. Also, specified timing parameter such as packet inter-arrival time will be enforced regardless of the computing platform.
- support for a variety of DoS attack types and other system parameters to be specified as input to the simulator: attack packet streams may be generated with various attack types including simple relay-attack and strong relay-attack types by attack stream generators. Also, other system parameter values, such as block size(n), redundancy level (t), message size in each packet, loss rate, and the number of blocks to be simulated, may be specified to the simulator.
- formulation and estimation of DoS resistance: the resistance level to various attack types is formulated as a number of attack streams that may be tolerated without affecting packet authentication throughput. That is, a threshold on the number of attack streams will be found beyond which packet authentication delays will increase indefinitely.

Taking these goals into consideration, we designed and implemented a simulator in Java, and carried out extensive simulations. Figure 3 shows the overall architecture of the simulator. The receiver side decoding is performed following the algorithm given in Figure 2.



Network
Raw Packet Buffer
Mj
Ki-1
Ej
Γj
Dj
(1) If authentic key is already found for Bi and it matches Ki-1 in this packet
Yes
Go to (5)
No
(2) Retrieve Ki-1 and check Hash(Ki-1) == Ki-2
No
Exit (Packet is altered)
Yes
(3) Store Ki-1 as a valid key
(4) Loop on each partition for Bi -- Apply MAC with a key Ki-1 to some Ej in next partition (stored in SymbolBuffer) which has at least n-t members. And check whether it matches Γj in the same member. – Partitions are checked in FIFO order (based on times when (n-t)-th member was received).
If match, execute (4-1) and (4-2)
If no match, execute (4-3)
For failed partition
(4-3) remove the partition from SymbolBuffer, and return to (4).
(4-1) For matching partition, perform erasure decoding and signature verification
(4-2) If successfully verified, store hashes/signature into HashBuffer, and go to (5). Otherwise, return to (4)
HashBuffer
(10) Apply Distillation Decoding, and store Ej and Γj into a matching partition in SymbolBuffer
SymbolBuffer
(5) Valid timestamp or sequence number
No
Exit
Yes
(6) Finding hash from message
(7) if no authenticated hashes exist for Bi in HashBuffer, go to (10), otherwise continue
(8) matching with authentic hashes in HashBuffer (for Bi)
Failure
Exit
Success
(9) Authenticate message.

Figure 2. Detailed algorithm at the receiver

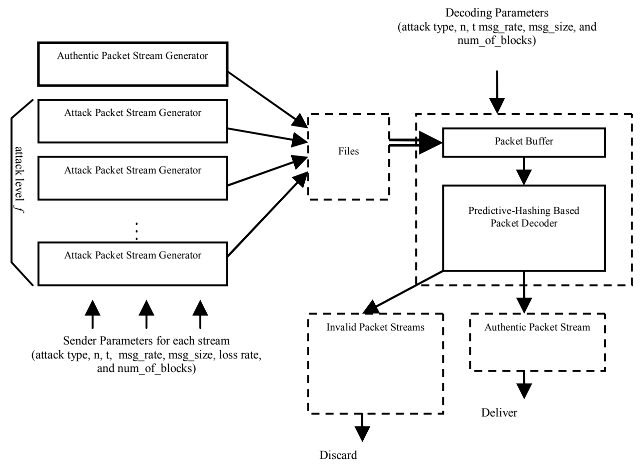
The reason why authentic and attack packets are stored into multiple files is to allow all the resources to be devoted later for running the receiver side decoding routine. Another alternative approach might have been to let packet generators run concurrently along with the receiver side decoding process. But, this approach would not permit us to measure exact resistance levels and resource usages as if the computer were wholly used for processing received packets as in real situation. We also assume that the overheads resulting from file access is insignificant to the simulator performance compared to the cases where the packets are received from the network.

5. Simulation Results

We conducted extensive simulations by running the simulator on a PC with Pentium 4 CPU (1.7 GHz) and 1GB of memory.

CPU Time Requirements

In Table 1 we show resource usage statistics in terms of CPU times with an attack level f=0 (i.e., no attack stream was introduced). The simulator was run for 100 block periods where a block period was 1.5 second. We used system parameter values n=32, t=16, loss rate =0.01, and message size = 1024 bytes.



Decoding Parameters (attack type, n, t msg_rate, msg_size, and num_of_blocks)
Authentic Packet Stream Generator
Attack Packet Stream Generator
Attack Packet Stream Generator
Attack Packet Stream Generator
attack level f
Files
Packet Buffer
Predictive-Hashing Based Packet Decoder
Sender Parameters for each stream (attack type, n, t, msg_rate, msg_size, loss rate, and num_of_blocks)
Invalid Packet Streams
Authentic Packet Stream
Deliver
Discard

Figure 3. Simulator Architecture

algorithm steps (Figure 2)	Cumulative execution time (ms)	percentage
(1)	0	0.0
(2)	0	0.0
(3)	0	0.0
(4-1)	0	0.0
(4-2)	31	3.03
(4-3)	0	0.0
(4-Erasure Decoding)	0	0.0
(4-Signature Verification)	838	81.8
(5)	0	0.0
(6)	30	2.93
(7,8,9)	16	1.56
(10)	109	10.64
Total	1024	100%

Table 1. Cumulative CPU times and percentages of CPU times spent for each algorithm step when there was no attack stream.

Most of the CPU time is spent for the signature verification step (81.8%), followed by the distillation decoding step (10.64%). The average signature verification operation with data size of around 1200 bytes and signature size of 46 bytes took about 23-26 ms.

Figure 4 shows changes on CPU usage percentages among different algorithm steps with varying values of attack level. As is shown, more CPU time will be used for step (10), distillation decoding operation, while the percentage for step (4), signature verification operation, decreases as the attack level increases. This is due to the fact that only one signature verification operation is needed regardless of attack level in the PH-based approach, while the percentages of CPU times spent for the other steps will increase for higher attack levels. Note that this would not be true in PRABS [8] and signature verification cost will still dominate for higher attack levels.

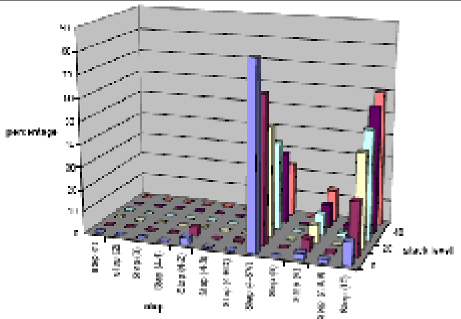


Figure 4. CPU time percentage changes with varying attack levels (from 0 through 50) in the presence of simple relay-attack streams

Resistance Level

For two different attack types, we ran the simulator to find threshold points (in terms of attack level) beyond which inter-packet authorization delay becomes steadily bigger than inter-packet arrival time. The ratios of inter-packet authentication delays to inter-packet arrival times are

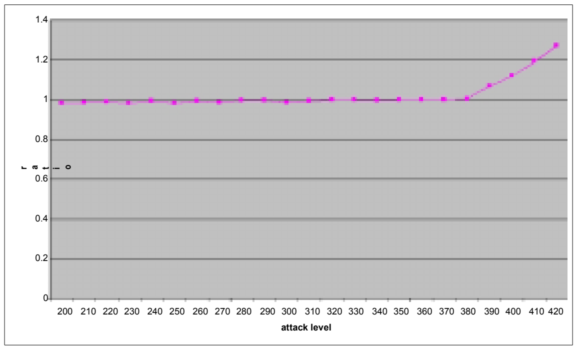


Figure 5. Ratios of inter-packet authentication delays to inter-packet arrival times with simple relay-attack streams. Resistance level is 390. With the PRABS approach, the resistance level becomes 26 as is shown in Figure 6.

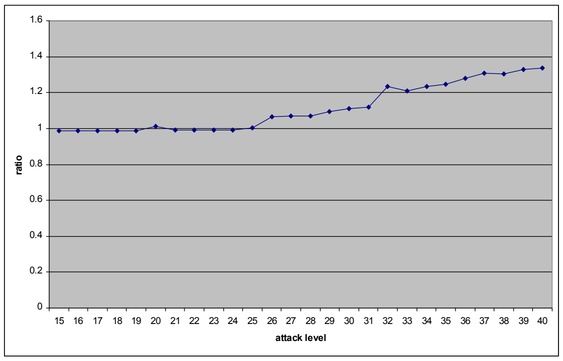


Figure 6. Ratios of inter-packet authentication delays to inter-packet arrival times with strong relay-attack streams. Resistance level is 26

shown in Figure 5 for simple relay attacks with different attack levels, where erasure-encoded symbol values in authentic packets are arbitrarily modified to generate attack packets. If this ratio is greater than 1.0, it means that the resistance level is reached. The simulations for other simple-relay attacks (such as modifying the key values or distillation code values) showed similar results due to the fact that, for any kind of simple relay-attacks, only one signature verification operation is needed in the PH-based approach.

Figure 6 shows the ratios of inter-packet authentication delays to inter-packet arrival times for strong relay-attacks when the same system parameter settings are used as in Figure 5. This figure may also be considered as the one showing performance gains we may achieve by using the PH-based approach compared to the PRABS approach when simple relay-attacks are launched. This is because the resource consumption in PRABS will increase in proportion to the number of attack streams regardless of attack types. In PRABS, the same number of signature verification operations is needed as the number of streams that a receiver receives.

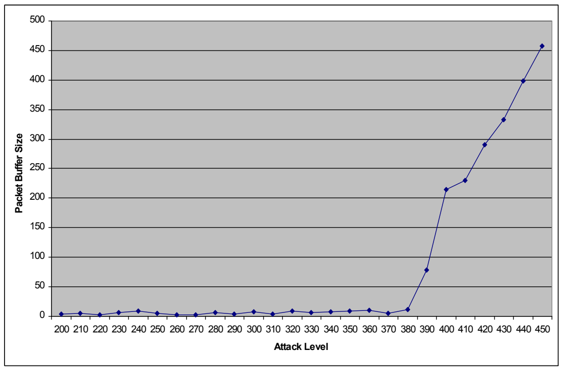


Figure 7. Memory requirement in terms of packet buffer size in the PH-based approach. The buffer size begins increasing around resistance level 390. With PRABS, the buffer size begins increasing around attack level = 26 as is shown in Figure 8.

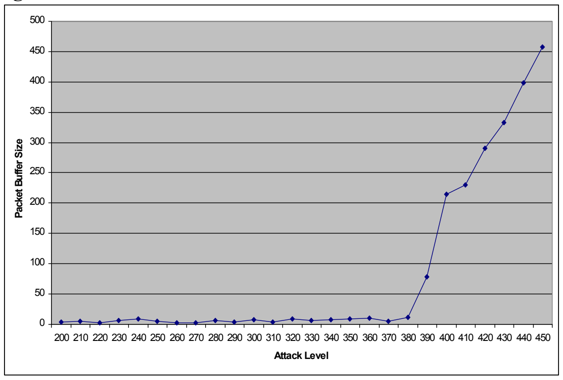


Figure 8. Memory requirement with strong relay-attack streams. It begins increasing near resistance level=26. Packet Buffer Size is measured in terms of the maximum number of packets stored in the packet buffer at any time

Memory Requirements

The memory requirements in the presence of simple relay-attacks streams and strong relay-attack streams are shown in Figure 7 and 8. Note that the major memory requirement comes from the packet buffer where the incoming packets are stored while the receiver is processing packets. The memory requirements for other buffers, such as Symbol Buffer and Hash Buffer, are negligible compared to that for the packet buffer due to the size difference, and are not shown here due to space limitation.

6. Conclusion

We designed and implemented a simulator based upon a new PH-based multicast authentication protocol. This simulator may be used on any computing platform to measure exact resistance level to various types of DoS attacks in different parameter settings. This tool may also be used to determine optimal system parameter values such as block period (p), block size (n), redundancy level (t), etc. This simulator was used to derive detailed resource usage information, and to measure the resistance levels against different types of DoS attacks on a selected computing platform. The result shows that our PH-based protocol outperforms other protocols (including PRABS which is outperformed by 15 times) in terms of resistance to DoS attacks.

References

[1] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Taming IP packet flooding attacks. In *Proceedings of Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.

[2] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology --EUROCRYPT ’97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, 1997.

[3] J. Benaloh and M. de Mare. One way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – EUROCRYPT ’93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, 1993.

[4] Seonho Choi, "Denial-of-Service Resistant Multicast Authentication Protocol with Prediction Hashing and One-way Key Chain," ism, pp. 701- 706, In Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM'05), 2005.

[5] Seonho Choi and Yanggon Kim, “Resource Requirement Analysis for a Predictive-Hashing Based Multicast Authentication Protocol,” In Proceedings for EUC Workshops, pages 302-311, IFIP, August 2006.

[6] M. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proceedings of Information Security Conference (ISC 2002)*, volume 2433 of *Lecture Notes in Computer Science*, pages 372–388, 2002.

[7] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, Distillation codes and applications to DoS resistant multicast authentication, in Proc. 11th Network and Distributed Systems Security Symposium (NDSS), San Diego, CA, Feb. 2004.

[8] Leslie Lamport, "*Password Authentication with Insecure Communication*", Communications of the ACM 24.11 (November 1981), 770-772

[9] A. Pannetrat and R. Molva. Efficient multicast packet authentication. In *Proceedings of the Symposium on Network and Distributed System Security Symposium (NDSS 2003)*. Internet Society, Feb. 2003.

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

[11] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pages 35–46. Internet Society, Feb. 2001.