

---

# **2005 FIRST WORKSHOP ON SECURE NETWORK PROTOCOLS (NPSEC)**

**BOSTON, MASSACHUSETTS, USA**

**NOVEMBER 6, 2005**

**HELD IN CONJUNCTION WITH ICNP 2005:  
THE 13<sup>TH</sup> IEEE INTERNATIONAL CONFERENCE  
ON NETWORK PROTOCOLS**

---

---



*Copyright and Reprint Permission:* Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limit of U.S. copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. For other copying, reprint, or republication permission, write to IEEE Copyrights Manager, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. All rights reserved. Copyright ©2005 by the Institute of Electrical and Electronics Engineers, Inc.

IEEE Catalog Number: 05EX1182  
ISBN: 0-7803-9427-5  
Library of Congress: 2005930834

*Additional copies may be ordered from:*

IEEE Operations Center  
445 Hoes Lane  
Piscataway, NJ 08854-4150  
Tel: +1 800 678 IEEE (+1 800 678 4333)  
+1 732 981 1393  
Fax: +1 732 981 9667  
<http://shop.ieee.org/store/>  
E-mail: [customer-service@ieee.org](mailto:customer-service@ieee.org)

## TABLE OF CONTENTS AND SCHEDULE

Time		Page No.
8:30	Welcome and Introductions	
8:45	Keynote, <i>Hari Balakrishnan (Massachusetts Institute of Technology)</i>	
10:30	L3A: A Protocol for Layer Three Accounting ..... <i>Alwyn Goodloe, Matthew Jacobs, Gaurav Shah (University of Pennsylvania), Carl Gunter (University of Illinois, Urbana-Champaign)</i>	1
10:50	Mitigating DoS Attack Through Selective Bin Verification ..... <i>Micah Sherr, Michael Greenwald, Carl Gunter, Sanjeev Khanna, Santosh Venkatesh (University of Illinois, Urbana-Champaign)</i>	7
11:10	On Design Tradeoffs between Security and Performance in Wireless Group Communicating Systems ..... <i>Jin-Hee Cho and Ing-Ray Chen (Virginia Tech)</i>	13
11:30	Detecting and Filtering Instant Messaging Spam – A Global and also Personalized Approach ..... <i>Zhijun Liu, Weili Lin, Na Li, David Lee (The Ohio State University)</i>	19
11:50	Analysis of IPsec Overheads for VPN Servers ..... <i>Craig Shue, Youngsang Shin, Minaxi Gupta, Jong Youl Choi (Indiana University, Bloomington)</i>	25
1:30	Practical Security for Disconnected Nodes ..... <i>Aaditeshwar Seth and Srinivasan Keshav (University of Waterloo)</i>	31
1:50	Achieving K-Anonymity in Mobile Ad Hoc Networks ..... <i>Xiaoxin Wu and Elisa Bertino (Purdue University)</i>	37
2:10	Phyllo: A Peer-to-Peer Overlay Security Framework ..... <i>William Heinbockel (MITRE) and Minseok Kwon (Rochester Institute of Technology)</i>	43
2:30	Identity Theft Protection in Structured Overlays ..... <i>Lakshmi Ganesh and Ben Y. Zhao (UC Santa Barbara)</i>	49
3:30	An Effective Intrusion Detection Approach for OLSR MANET Protocol ..... <i>M. Wang, L. Lamont (Communications Research Centre), P. Mason, M. Gorlatova (DRDC)</i>	55
3:50	The Attackers' Influence on the Tactical Assessments Produced by Standard Alert Correlation Systems ..... <i>Stephen Neville (University of Victoria)</i>	61
4:10	Policy Segmentation for Intelligent Firewall Testing ..... <i>Adel El-Atawy, Khaled Ibrahim, Hazem Hamed, Ehab Al-Shaer (DePaul University)</i>	67
4:30	A Distributed Reputation Approach to Cooperative Internet Routing Protection ..... <i>Harlan Yu, Jennifer Rexford, Edward Felten (Princeton University)</i>	73
4:50	Closing Remarks and Discussion	
	<b>Author Index</b> .....	79

## MESSAGE FROM THE GENERAL CHAIR AND PROGRAM CHAIRS

---

Welcome to the 2005 First Workshop on Secure Network Protocols (NPSec), and welcome to Boston! NPSec is a one-day event held in conjunction with IEEE ICNP 2005: The 13th IEEE International Conference on Network Protocols.

NPSec focuses on two general areas. The first focus is on the development and analysis of secure or hardened protocols for the operation (establishment and maintenance) of network infrastructure, including such targets as secure multidomain, ad-hoc, sensor or overlay networks, or other related target areas. This can include new protocols, enhancements to existing protocols, protocol analysis, and new attacks on existing protocols. The second focus is on employing such secure network protocols to create or enhance network applications. Examples include collaborative firewalls, incentive strategies for multiparty networks, and deployment strategies to enable secure applications.

We would like to thank those who have made special contributions to this workshop. The ICNP 2005 general chairs, Professor Azer Bestavros (Boston University) and Professor James Kurose (The University of Massachusetts at Amherst), and the ICNP 2005 technical program committee chairs, Professor Ibrahim Matta (Boston University) and Professor Mohamed Gouda (The University of Texas at Austin), have guided us through the process of starting the first workshop to be held in conjunction with ICNP, and provided significant support in handling local and logistical arrangements and financial matters. We sincerely thank them for their tremendous efforts, without which the creation of this workshop would not have been possible. We would also like to extend special thanks to Professor James Minseok Kwon and Professor Cristina Nita-Rotaru who did an excellent job in publicizing the workshop and creating the web pages. Finally, we would like to thank the IEEE acquisitions staff, especially Pat Thompson, for guiding us through the production of IEEE Xplore-compliant proceedings.

The key to the success of this workshop is its technical program. We would like to thank the NPSec technical program committee for their tremendous efforts in the paper review and selection process. Thirty-one papers were submitted to the workshop. We have selected thirteen excellent papers for publication in the workshop proceedings and presentation at the workshop. The selected papers span several exciting and timely topics, ranging from access control and routing, to wireless networks and peer-to-peer systems.

Last but not least, we would like to thank all speakers, and all authors who submitted their work to NPSec, and we look forward to several stimulating discussions. We hope that you will enjoy the workshop!

Sonia Fahmy, George Kesidis, Nicholas Weaver  
NPsec 2005 General Chair and Program Chairs

## COMMITTEES

---

### ORGANIZING COMMITTEE:

*General Chair:*

Sonia Fahmy, Purdue University

*Technical Program Committee Chairs:*

George Kesidis, Pennsylvania State University

Nicholas Weaver, International Computer Science Institute

*Publicity Chair:*

James Minseok Kwon, Rochester Institute of Technology

*Web Chair:*

Cristina Nita-Rotaru, Purdue University

### TECHNICAL PROGRAM COMMITTEE:

Marco Ajmone-Marsan, Politecnico di Torino, Italy

Ehab Al-Shaer, DePaul University

David Brumley, Carnegie Mellon University

Guohong Cao, Pennsylvania State University

Joseph Evans, U.S. National Science Foundation

Lixin Gao, University of Massachusetts, Amherst

Carl A. Gunter, University of Illinois at Urbana-Champaign

George Kesidis, Pennsylvania State University

Edward Knightly, Rice University

Iordanis Koutsopoulos, University of Thessaly, Greece

Carl Landwehr, University of Maryland

Douglas Maughan, U.S. Department of Homeland Security

Patrick McDaniel, Pennsylvania State University

Jelena Mirkovic, University of Delaware

Peng Ning, North Carolina State University

Cristina Nita-Rotaru, Purdue University

Phil Porras, SRI

Saswati Sarkar, University of Pennsylvania

Lakshminarayanan Subramanian, University of California at Berkeley

Nina Taft, Intel Research

Nicholas Weaver, International Computer Science Institute

Felix Wu, University of California at Davis

Jun Xu, Georgia Institute of Technology

Bulent Yener, Rensselaer Polytechnic Institute

# L3A: A Protocol for Layer Three Accounting

Alwyn Goodloe, Matthew Jacobs and Gaurav Shah  
University of Pennsylvania

Carl A. Gunter  
University of Illinois Urbana-Champaign

## Abstract

*Accounting protocols are used to quantify traffic to support billing, QoS assurances, and other objectives. Current protocols do not provide complete security for this purpose because of the threat of ‘cramming’ attacks in which unauthenticated parties can introduce traffic that the accounting system attributes incorrectly. In this paper we explain this vulnerability and introduce a protocol, Layer Three Accounting (L3A), that addresses it through the coordinated establishment of a family of IPsec tunnels. Our goal is to give a practical specification and implementation of the protocol and show its efficiency. We demonstrate that the latency for setting up and tearing down L3A connections is about one third slower than one gets for end-to-end connections alone, but the bulk rate of transfer is improved by 100% over the typical alternative configuration for accounting.*

## 1. Introduction

Accounting protocols are used to quantify traffic to support billing, QoS assurances, and other objectives. A common way to do this, as seen in cellular data systems, is to associate traffic with specific clients on an access network and use a Network Access Server (NAS) with an associated accounting system such as a RADIUS server to maintain accounting records such as the number of bytes or packets the client exchanges with a server on the Internet. This is done by authenticating the client to the NAS by the use of a cryptographic tunnel authenticated with a credential from the client, or by cruder means, such as associating the client to a specific MAC or IP address. This approach provides some level of secure attribution of the traffic that passes from the client across the accounting node into the Internet. Response traffic that is directed to the client is also attributed as part of the accounting system. However, this response traffic is less securely identified than the client’s traffic to the NAS since it is not explicitly authenticated as valid response traffic at the accounting node. This raises a threat that an adversary in the Internet will ‘cram’ traffic into the authenticated channel between the NAS and the client and such traffic will be attributed to the user as valid response traffic. If the user is properly authenticating his connection to the server in the Internet, then such traffic will be discarded, but not before it has been attributed to his account by the accounting node, which did not perform a similar authentication.

The aim of this paper is to explore an approach to prevent-

ing such attacks by authenticating traffic across the accounting node in both directions, including traffic sent *to* the client as well as traffic sent *from* the client. Specifically, we consider how to do this with a protocol that coordinates a collection of network layer tunnels. It is common to provide accounting through the use of link layer mechanisms, especially for wireless links, but a network layer solution offers advantages for portability. The primary contributions are a protocol, L3A, for layer three accounting that addresses cramming attacks and a demonstration that the protocol can be efficiently implemented using a family of IPsec tunnels. Excellent progress has been made in recent years on the Internet Key Exchange (IKE) protocol for dynamic tunnel establishment between pairs of nodes. The next step to leverage this progress is to show how IKE can be used as building block in more complex multi-node protocols to achieve high-level security objectives such as robust accounting. The design and analysis of such protocols can be subtle. In this paper we focus on practical aspects such as the specification of a protocol that is simple to implement with satisfactory efficiency using existing standards and software. In particular, we have specified L3A and implemented it on FreeBSD based on our own implementation of (a fragment) of IKEv2. Our experiments show that L3A accounting costs about 160ms for both set up and tear down. This is about 2.4 times more than the same operations for an IPsec tunnel alone. However, tunnel reuse in L3A reduces this to a factor of 1.5 in the common case where the client-to-NAS tunnel already exists. On the other hand, L3A improves bulk traffic performance by 100% over a naive (but typical) approach to accounting where accounting uses an encrypted tunnel to the NAS.

In the second section we provide background on cramming attacks, our network layer approach, and related work. The third section describes the L3A protocols. The fourth section describes our implementation and experiments. The fifth section concludes.

## 2. Background

Some background is required to understand cramming attacks and our approach to solving them. We begin with a description of the problem, then we sketch our network layer approach to solving it, and then survey related work.

### 2.1. Cramming Attacks

Figure 1 illustrates a typical example of a packet-based communication link with an accounting element. The NAS is

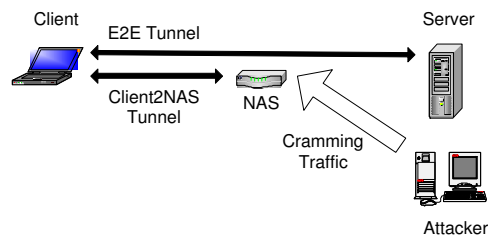


Figure 1: Cramming Attack

placed at a network bottleneck, such as a wireless access point or router, where it monitors traffic to and from the clients who will be charged for network access. The NAS is typically supported by an authentication and accounting system such as a RADIUS server and collects information about such parameters as throughput of the client, the number of sessions it runs, the duration of its access, or anything else it is able to record. To ensure proper attribution, a tunnel can be placed between the client and the NAS so that each packet from the client is authenticated. Such tunnels are often placed at link layer, but could be placed at virtually any network layer. The client uses its connection through the NAS to visit various sites in the Internet where it finds servers. Clients often secure the link to the server with an additional tunnel, which stretches end-to-end between the client and the server through the NAS. This tunnel provides privacy from, among other things, the NAS itself. As the client makes requests to the server and the server sends its responses through the NAS, the NAS does its accounting.

The architecture of Figure 1 suffers from a gap in its protection of the NAS accounting system. The NAS is able to authenticate all traffic coming from the client and will (typically) drop traffic it receives from any other source on the client-side interface that purports to come from the client but is not authenticated. By contrast, response traffic from the server is unauthenticated by the accounting system. This raises a threat that a node on the Internet could direct false response traffic into the NAS. Since the NAS does not authenticate traffic on the server-side interface, it will typically dispatch this traffic on to the client. The client will probably discard the traffic since it will not match its tunnel to the server, but by the time this traffic reaches the client, it has been attributed to it by the NAS thus compromising the integrity of the accounting database. We refer to this as a *cramming attack*.

The actual details of the crammering attack depend on the network architecture and details of the accounting mechanism. The seriousness of the threat depends on how response traffic is forwarded to the client by the NAS. For networks that use globally routable network IP addresses and allow arbitrary services to be run on the clients (*i.e.* outside hosts can initiate connections to these services), the crammering attack is easy to perform in the absence of additional firewalling mechanism at the NAS. Firewalling on the client side may still be in use, but as the NAS will not be aware of it, it will forward any packet (and account for it) onto the client host.

If Network Address Translation (NAT) is being used, a crammering attack is more complicated. For purposes of this

discussion we only consider the TCP protocol: details for UDP are similar. As NAT is used to share one globally routable address with hosts having private addresses, an incoming packet will only be forwarded if the NAT router determines it to be a part of an existing connection initiated earlier by a client. Connections are identified by a 4-tuple which response packets must use. As the destination IP in the response packets will be that of the NAS address, three remaining values need to be determined by the attacker: the IP address of the server, the server port number, and the client port number. Guessing these values for a particular client's connection is challenging for an off-path attacker (that is, an attacker that is not on the routing path between the NAS and the server).

However, some tricks can be used to make the network vulnerable to crammering attacks against random clients. For networks that support a large number of clients, many users are likely to be connected to relatively popular services on the web such as popular search engines and portals (*e.g.* `google.com`), instant messaging services (*e.g.* AIM), IMAP and POP mail access (*e.g.* `gmail.com`, `yahoo.com`), and so on. Thus, the attacker has a large number of fixed server IP/port pairs to choose from as possible endpoints for different connections a NAT NAS might be tracking. Only the client port information needs to be guessed. Client ports are often chosen from a fixed set of ranges (ephemeral port ranges) whose exact values are dependent on the particular OS and configuration. By using different client port ranges and sending out packets with different client port values picked from probable ranges, it is possible to get response packets past the NAS and hence successfully perform the crammering attack. There are some NAT implementations that make this a very effective approach. For instance, if port numbers are allocated sequentially and there is an insider behind the NAS, then active port number can be guessed easily. Even if there is no such edge, a brute force attack can achieve some success. On a Pentium 4 running Linux 2.6.10 at 2.4 Ghz with 1GB of RAM we were able to send packets with a 1.4Kb payload and varying port values at a rate of around 10,000 packets/second with code that had no optimizations and no changes to the drivers or the kernel. Thus, a brute force attack on the client port numbers can be performed in a small amount of time. Although we do not know a specific way to do it, the existence of any technique for telling if a crammering packet 'hit' a client would make such an attack quite effective.

The time window in which the attack is successful depends on the length of the time period for which the NAT router maintains state information for each connection. As this state information is the one that is used to ascertain whether to forward a certain response packet to a client, it is maintained for at least the period of the connection. For connections that only last momentarily (*e.g.* HTTP), it is important that the attack take place in the period when the NAT router still has this state information stored. RFC 2663[18] recommends that the NAT router maintain state for at least another 4 minutes ( $2 * \text{Maximum Segment Lifetime}$ ) after it thinks that the connection has terminated. As a NAT router can never be sure whether the connection tear-down packets it saw on the wire actually reached the destination host, it continues to forward packets for that connection for a little while after the observed teardown (to en-

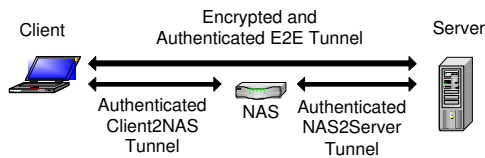


Figure 2: Layer Three Accounting Tunnels

able retransmissions). Our experiments with a Linksys wireless router doing NAT showed that connection state was maintained for 7 more minutes after the actual connection was terminated. Thus it was possible to cram packets into this connection after its actual termination for another 7 minutes. These factors contribute to increasing the length of the vulnerability time window and hence give plenty of time to the attacker for a brute force attack.

## 2.2. Layer Three Accounting

These points above show that the most assured approach to preventing a cramming attack is to establish some kind of tunnel between the NAS and the server with parameters that are not vulnerable to guessing, even by an on-path attacker. A simple approach would be for the NAS to set up the tunnel when it sees a communication request from one of its clients. This can be done at a variety of network layers, possibly using higher-layer protocols like SSL or SSH.

Our approach with L3A is to coordinate the establishment of all three of the tunnels involved in this protection system and base them on IPsec. IPsec [12] is a suite of network-layer security protocols that has been standardized in the IETF. Cryptographic tunnels called *security associations* (SA) define the transformations that get applied to each packet traveling in the association. Although unidirectional, they are usually created in pairs with one association flowing in each direction. Each node maintains a security association database (SADB) containing information on the associations active at that node. *Security policies* determine which packets travel in which security association. Each node maintains a security policy database (SPDB) that contains the policies that apply to incoming and outgoing traffic. Though they use classical cryptographic protocols as building blocks, L3A and its sibling tear-down protocol can be viewed as signaling protocols. That is, they install and manipulate state in the network comprising the association and policy databases. To avoid manual key configuration, the symmetric keys used by IPsec security associations are usually established by invoking a key-exchange protocol. The IETF designed the Internet Key Exchange (IKE) [11] protocol for just this purpose. IKE not only establishes the shared key, it sets up a pair of associations between the two parties.

Figure 2 illustrates the basic configuration, where each of the black double-headed arrows indicates an IPsec connection established using IKE and with corresponding entries in the SADB and SPDB at each end. Having a unified network-layer multi-tunnel protocol has at least three advantages. First, the tunnels can be coordinated for efficiency. The two NAS tunnels can do only authentication since they were intended only for ac-

counting while the end-to-end tunnel provides confidentiality. Second, it is possible to develop a suitable credential mechanism for the overall protocol rather than just for each individual tunnel. For instance, the NAS is able to use an L3A credential from the client to establish its connection to the server. Third, an implementation at network layer can exploit the elegance and advances of IPsec. For instance, IPsec provides robust DoS protection, has hardware support, and is portable across many link layers.

## 2.3. Related Work

There are two main areas of related work for L3A: protocols for accounting and protocols for dynamic establishment of IPsec tunnels.

There are many proposed schemes proposed for accounting for wireless services [4, 9, 13]. In order to create reliable bills, the accounting system must be secure and reliable. There are several accounting protocols in use today. Among them are Simple Network Management Protocol (SNMP) [1] and the RADIUS accounting protocol [16]. Although they may require a client to authenticate themselves, they do not protect against the cramming attacks described above. We are not the first to identify the billing and accounting systems as a ripe target for attacks. Since a company may lose customers or face costly financial credits due to over-billing, there are obvious incentives to pay for protecting accounting services and this fact has not been lost on the vendors of security devices. Network security products are now being advertised as providing protection against such attacks. For example, Juniper Networks claims that their security gateway for GPRS provides protection against over-billing attacks [10]. The security devices being aimed at this market are generally sophisticated stateful firewalls and we have seen that cramming attacks can evade such measures.

The tools available today for the management of IPsec tunnels remain relatively limited. Even when using IKE to dynamically establish tunnels, there must be some prior configuration of each node. Network managers usually do this from the router's command line interface. Centralized management tools such as Solsoft's Policy server [17] provide a network manager the capability of configuring IPsec tunnels from a central location. Fu and Wu have proposed a centralized management system that generates the detailed IPsec policy and association entries from high-level specifications [6, 7]. Centralized configuration is impractical in our scenario given that the NAS would have to be preconfigured to connect to all servers that a client can access. Others have developed protocols to set up IPsec connections. Cisco's Dynamic Multipoint VPN (DMVPN) [2] feature is a protocol for establishing tunnels between spoke nodes of a hub and spoke configuration. Each spoke node must maintain a permanent tunnel to a Next Hop Resolution Protocol (NHRP) [14] server acting as a hub. When a spoke wishes to communicate directly with another spoke, it queries the hub for the information it needs to set up a tunnel. Cisco's Tunnel Endpoint Discovery (TED) protocol [5, 3] is protocol for discovering the endpoint of a tunnel and setting up the association.

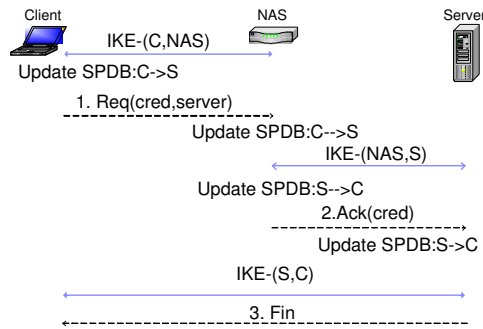


Figure 3: L3A Set Up

### 3. Protocols

In this section we present the two protocols for layer three accounting. The L3A set up protocol was described formally in [8] using term rewriting. This section provides a specification aimed at implementers and also provides an L3A tear down protocol.

#### 3.1. L3A Set Up

The L3A set-up protocol creates the configuration of tunnels given in Figure 2. The client will initiate the protocol. Although the client has a relationship with the NAS as well as with the server, the NAS is not assumed to have any relationship with the server. So the client must pass a credential to the NAS that it presents to the server on behalf of the client in order to establish the tunnel between the NAS and the server. IKE will be used to establish the three pairs of tunnels. Additional messages are required to pass credentials, to tell the NAS which server it should set up a tunnel with, and to tell the client that the protocol has completed. The protocol is illustrated in Figure 3 and the details are as follows.

**Client initiates protocol** The client C identifies the server S it desires to contact and the NAS that will provide access to the Internet.

**Establish Client-NAS SA** If there does not already exist a pair of associations between C and NAS, then the client invokes IKE to establish a pair of associations between the client and the NAS.

The client updates its SPDB with a policy saying that all traffic from the client to the server should flow through the C → NAS association. The client then forms a message containing the name of the server with which the client wishes to communicate and a credential that the NAS presents to the server on behalf of the client.

**Msg1** C → NAS : Req(cred, S)

**NAS Receives Req** If the NAS receives notification that a key exchange has completed with the client, it waits to receive a Req. On the other hand, a Req message may be received in a preexisting association. When the Req(Cred,S) message arrives, the NAS updates its SPDB with a policy saying that all traffic from the client to the

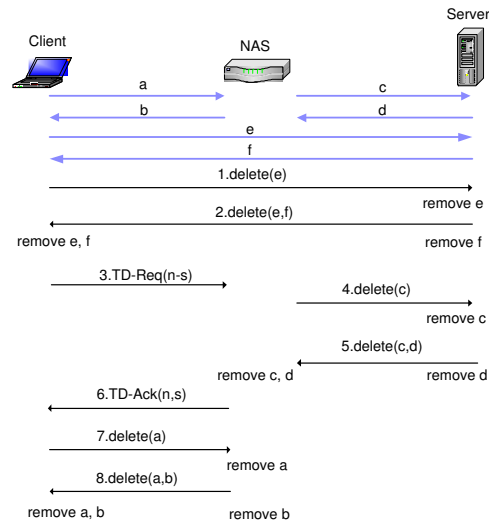


Figure 4: L3A Tear Down

server should flow through the C → NAS association. The NAS extracts the server address and credential from the Req message.

**Establish NAS-Server SA** If there does not already exist a pair of associations between NAS and S, then the NAS invokes IKE to establish an SA between the NAS and the server.

The NAS updates its policy database to reflect the fact that all traffic flowing from the server to the client should travel in the S → NAS association. The NAS then sends the message

**Msg2** NAS → S : ACK(cred)

**Server Receives Ack** Upon notification that a key exchange with the NAS has occurred, the server waits for an Ack message to arrive in the newly created tunnel. On the other hand, the Ack may arrive in a preexisting tunnel. Upon receiving the Ack message, the server extracts the credential that had been passed by the client. If the credential is valid, the server updates its SPDB with a policy saying that all traffic flowing from the server to the client should travel in the S → NAS association.

**Establish Client-Server SA** The server invokes IKE to set up a pair of end-to-end associations between the S and C. When the key exchange is complete the server notifies the client that the protocol has completed by sending a message.

**Msg3** S → C : FIN

#### 3.2. L3A Tear Down

We now give a brief description of the protocol that tears down the tunnels established by L3A. The steps are illustrated in Figure 4. We label the associations as  $a-f$  as in the figure. In practice these identifiers would be the SPIs for each association. The protocol works as follows.

**Client initiates protocol** The client initiates the protocol by sending to the server

**Msg1**  $C \rightarrow S$  : delete( $e$ ) Upon receiving a message of this form, the server deletes the association and policy for  $e$ . and sends the message

**Msg2**  $S \rightarrow PC$  : delete( $e, f$ ) and removes the association and policy for  $f$ . Upon receiving a message of this form, the client deletes the  $e$  and  $f$  associations as well as their policies. The client then sends the message

**Msg3**  $C \rightarrow NAS$  : TDRReq( $C, n, s$ ) Upon receiving a message of this form, the NAS sends a message

**Msg4**  $NAS \rightarrow S$  : delete( $C, c$ ) Upon receiving a message of this form, the server removes the policy for all traffic flowing from  $S \rightarrow C$ . If there are no policies for other clients, the association is deleted. The server sends the message

**Msg5**  $S \rightarrow NAS$  : delete( $C, S, c, d$ ) If the  $c$  association was removed, the  $d$  association is now removed. Upon receiving this message the NAS removes the policy for  $S \rightarrow C$  and removes  $c$  and  $d$  if the tunnel is not in use by another client. The NAS then forms the following message

**Msg6**  $NAS \rightarrow C$  : TDAck( $n, s$ ) Upon receiving a message of this form, the client checks if there are sessions with other servers and if not, then the client forms

**Msg7**  $C \rightarrow NAS$  : delete( $a$ ) Upon receiving a message of this form, the NAS deletes the association and policy for  $e$  and sends the message

**Msg8**  $NAS \rightarrow C$  : delete( $a, b$ ) and removes the association and policy for  $b$ . Upon receiving a message of this form, the client removes the  $a$  and  $b$  associations as well as their policies.

## 4. Implementation

In order to demonstrate the practicality of the L3A protocol, an actual system running L3A is needed. We implemented the three principals of L3A (client, NAS, server) on three different machines, each running FreeBSD 4.8 and connected with a megabit/second network link. In our testbed, the client and server both are Micron 600MHz Pentium IIIs with 128MB of memory, and the NAS is a Dell 1.3 GHz Pentium IV with 256MB of memory.

The cryptographic operations of L3A are performed using the standard OpenSSL library. Updates to the kernel's SADB made by L3A are communicated through the PF\_KEY Key Management API, as described in RFC 2367[15]. Interestingly, this RFC does not completely describe the manipulation of the SPD, so a trial and error approach was required in order to interface with L3A correctly.

Our biggest challenge in the implementation was getting a suitable implementation of IKE. We wanted to use IKEv2 but with support for nested tunnels on our FreeBSD platform. Unfortunately we did not find an off-the-shelf system satisfying these requirements so we set out to implement enough of it ourselves to do our experiments. The IKE protocol is rather complex in that there are many terms of negotiation to support the

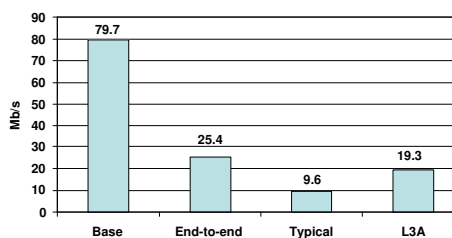


Figure 5: Throughputs

most general use. Such complexity is not needed for our purposes. Instead, we employ a somewhat stripped down version of IKE that we call IKE-. The simplifications are achieved by assuming that encryption is done with triple DES and that authentication is done using HMAC rather than negotiating which cryptographic mechanisms to use. A special flag is included in our protocol to indicate if the resulting tunnel should perform encryption and authentication or only authenticate traffic. Another simplification comes from eliminating IKE's two phase structure, where child associations are created in a second phase based on the shared key established in the first. IKE- has a single phase that terminates with the establishment of associations flowing in both direction. In the full version of this paper an appendix gives details of the IKE- protocol. The full paper is available at <http://securitylab.cs.uiuc.edu/contessans/papers/l3a.pdf>.

The experiments run on our testbed were designed to give performance results of the L3A protocol that can be compared to the performance of other solutions to the accounting problem. The first set of tests measures the raw throughput of client-server communication in 4 cases. The results appear in Figure 4. The first case, Base, is a baseline, with no IPsec at all, in order to quantify the maximum possible throughput of the connection. The second, End-to-end, utilizes IPsec end-to-end encryption and authentication, without accounting guarantees. Encryption has a significant impact on throughput, reducing it to barely a third of the unencrypted rate. This is particularly pronounced in the third case, Typical, which is the configuration of tunnels seen in most current accounting systems. The client maintains an encrypted tunnel with both the NAS and the server. This double encryption degrades performance to about a third of the End-to-end case because of the double encryption burden it places on the client. The final case, L3A, uses the tunnels set up by L3A. This entails three tunnels rather than the two used in the Typical case, but encryption is performed only end-to-end and the cost of the authenticated tunnels is much less than that of those that apply encryption. Consequently, the throughput performance of L3A is 101% better than that of the Typical configuration and only 32% lower than the case with no accounting. We did not measure the case where there are large numbers of clients, but in our tests with one client, the NAS was only lightly loaded.

The second set of tests measures the latency of the tunnel setup times. The results appear in Figure 4. For each scenario, the data reflects the time taken to both set up and tear down all appropriate tunnels. In the simplest case, End-to-end, just the

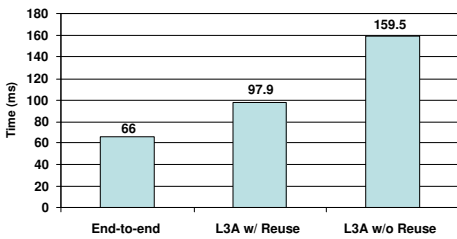


Figure 6: Latencies

client-server tunnel is set up. The next case to consider, L3A w/ Reuse, occurs in the common case, where a client-NAS tunnel already exists, and the remaining NAS-server and client-server tunnels are created and torn down by L3A, leaving the client-NAS tunnel for another L3A session. The final case, L3A w/o Reuse, describes the latency of L3A when the client-NAS tunnel is not reused, and all three tunnels are created and torn down by L3A. Thus the latency cost of establishing tunnels for accounting is 142% greater than that of end-to-end protection alone, but in the most common case, when there is already a tunnel between the client and NAS, it will be only 48% longer.

## 5. Conclusion

We have described cramming attacks and shown they pose practical threats to existing accounting protocols. We have described a way to address these threats by the coordinated establishment of a collection of IPsec tunnels using a protocol for layer three accounting called L3A. We implemented the L3A protocol on FreeBSD machines using our own implementation of a fragment of IKE and given evidence that the performance of L3A provides gains in bulk efficiency over more naive approaches while costing relatively little in increased latency. Perhaps the main contribution of the work is the demonstration of an elegant and efficient protocol for the dynamic establishment of a coordinated family of IPsec tunnels to achieve a multi-domain security objective (*viz.* secure accounting).

*Acknowledgements.* We would like to thank Tom Hiller and Pete McCann who first suggested we investigate the problem of securing the wireless accounting infrastructure. We would also like to thank Steve Bellovin for useful comments during several discussions on this work. This research was supported by the Office of Naval Research under MURI Research Contract N00014-02-1-0715.

## References

- [1] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple Network Management Protocol. RFC 1157, IETF, 1990.
- [2] Dynamic Multipoint VPN (DM VPN). Cisco White Paper. <http://www.cisco.com/>.
- [3] Tunnel Endpoint Discovery Enhancement. Cisco White Paper. <http://www.cisco.com/>.

- [4] J. Cushnie and D. Hutchison. Charging and Billing Challenges for Wireless Internet Networks. Technical report, Lancaster University.
- [5] S. Fluhrer. Tunnel Endpoint Discovery. Technical report, IPSP, 2001. <http://quimby.gnus.org/internet-drafts/draft-fluhrer-ted-00.txt>.
- [6] Z. Fu, S. Wu, W.H. Haung, K.Loh, F. Gong, and C. Xu. IPsec/VPN Security Policy: Correctness and Conflict Resolution. In M. Sloman and J. Lobo and E. Lupu, editor, *Policies for Distributed Systems and Networks*, pages 39–56, 2001.
- [7] Z. Fu and S.F. Wu. Automatic Generation of IPsec/VPN Security Policies in an Intra-Domain Environment. In *International Workshop on Distributed Systems: Operations and Management*, Lecture Notes in Computer Science 1995, pages 279–290, October 2001.
- [8] Alwyn Goodloe, Mark-Oliver Stehr, and Carl A. Gunter. Formal Simulation of Layer 3 Accounting. In *Workshop on Issues in the Theory of Security (WITS '05)*, Long Beach, CA, January 2005. IFIP.
- [9] Evolution of Charging and Billing Models for GSM and Future Mobile Internet Services, 2000.
- [10] Infrastructure Security Gateway for GPRS Networks. Juniper Networks White Paper. <http://www.juniper.net>.
- [11] C. Kaufman. Internet Key Exchange(IKE V2) Protocol. RFC 2407, IETF, 2004.
- [12] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, IETF, 1998.
- [13] M. Koutsopoulou, A. Kaloxylos, A. Alonistioti, L. Merakos, and K. Kawamura. Charging, Accounting, and Billing Management Schemes in Mobile Telecommunications Networks and the Internet. *IEEE Communications Surveys*, 6(1), 2004.
- [14] J. Luciani, D. Katz, D. Piscicello, B. Cole, and N. Doraswamy. NBMA Next Hop Resolution Protocol (NHRP). RFC 2332, IETF, 1998.
- [15] D. McDonald, C. Metz, and B. Phan. PF\_KEY Key Management API, Version 2. RFC 2367, IETF, 1998.
- [16] C. Rigney. RADIUS Accounting. RFC 2866, IETF, 2000.
- [17] Solsoft Policy Server: Better Management for Network Security. Solsoft White Paper, 2003. <http://www.solsoft.com>.
- [18] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF, 1999.

# Mitigating DoS Attack Through Selective Bin Verification

Micah Sherr<sup>†</sup>, Michael Greenwald<sup>‡</sup>, Carl A. Gunter<sup>\*</sup>, Sanjeev Khanna<sup>†</sup>, and Santosh S. Venkatesh<sup>†</sup>

<sup>†</sup> School of Engineering and Applied Science, University of Pennsylvania  
{msherr,sanjeevk,venkatesh}@seas.upenn.edu

<sup>‡</sup> Bell Labs  
greenwald@research.bell-labs.com

<sup>\*</sup> Department of Computer Science, University of Illinois at Urbana-Champaign

## Abstract

*Despite considerable attention from both the academic and commercial communities, denial-of-service (DoS) attacks represent a growing threat to network administrators and service providers. A large number of proposed DoS countermeasures attempt to detect an attack in-progress and filter out the DoS attack packets. These techniques often depend on the instantiation of sophisticated routing mechanisms and the ability to differentiate between normal and malicious messages. Unfortunately, neither of these prerequisites may be practical or possible.*

*We propose and evaluate a defense against DoS attacks which we call selective bin verification. The technique shows promise against large DoS attacks, even when attack packets are able to permeate the network and reach the target of their attack. We explore the effectiveness of our technique by implementing an experimental testbed in which selective bin verification is successfully used to protect against DoS attacks. We formally describe the mathematical properties of our approach and delineate “tuning” parameters for defending against various attacks.*

## 1. Introduction

The increasing number and severity of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks have become a growing annoyance to network administrators and service providers [7]. Consequently, there has been a significant body of work in which countermeasures are proposed to alleviate or eliminate the effects of these attacks. Typically, human operators defend against DoS by augmenting their networks with additional equipment or

by employing some filtering mechanism that prevents malicious packets from reaching their target [5].

While these approaches may be appropriate for several types of DoS attacks, they unfortunately are not always applicable or practical. The deployment of additional computational and network resources may be technologically or financially infeasible. Techniques that discriminate between normal and malicious traffic assume that these two traffic streams are somehow distinguishable. For some protocols, it may be prohibitively expensive to reliably make this distinction.

In this paper, we study the effectiveness of a novel countermeasure to DoS called *selective verification*. First introduced in [4], our technique allows the receiver to tolerate large DDoS attacks while providing service to legitimate senders. Unlike more orthodox approaches in which success is measured by the ability to prevent bogus requests from entering the targeted network, selective verification is effective even when attack packets reach the receiver.

In particular, this paper investigates a class of selective verification called *selective bin verification*. While previous work [4] has illustrated the ability of *sequential selective verification* to protect authenticated broadcasts from DoS, we show that bin verification is an effective technique to protect point-to-point protocols.

The remainder of the paper is organized as follows. In the following section, we discuss related work. In Section 3, we describe the selective bin verification protocol. To test the efficacy of our approach, we describe and analyze various selective bin verification experiments in Section 4. We conclude and discuss areas of future research in Section 5.

## 2. Related Work

Much prior work has investigated the problem of protecting against DoS. Existing defenses generally fall into two categories: those that mitigate the attack by adding additional resources and those that attempt to differentiate between legitimate and malicious (or anomalous) traffic. While the former approach is often used in practice to mitigate the effects of an ongoing attack [2], most of the academic literature has focused on the latter category.

In particular, automated detection and response mechanisms present promising defenses. Network intrusion detection systems [8] are effective at detecting certain types of DoS attacks. However, approaches that rely on signature or anomaly detection make the assumption that the attack packets are distinguishable from the legitimate traffic. While this may be true for certain attacks (e.g., “smurf” attacks [3]), other DoS techniques merely inundate a service with requests that appear valid.

Additional work has focused on curbing DoS attacks at the source [5, 6]. Here, a router detects an anomalous traffic flow and actively responds to the attack. However, since distributed DoS attacks are often carried out by unwitting zombie hosts that may be dispersed throughout the Internet, this task is non-trivial without the widespread adoption of cooperating DoS-aware routers.

An important advantage of selective bin verification is that it reduces the effects of a DoS attack even in the case where the attack fails to be detected. It is a software-based approach and requires no hardware modifications. While our technique is not a panacea to the DoS problem, its use in conjunction with existing approaches may significantly reduce the effects of many DoS attacks. In the section that follows, we describe our technique more formally.

## 3. Selective Bin Verification

### 3.1. Sequential Selective Verification

Selective verification has previously been introduced as an effective approach to thwarting DDoS attacks [4]. Specifically, the focus of the prior work was to alleviate the effects of a DoS attack against an authenticated broadcast stream. The authors considered a protocol in which the receiver must exert significant processing resources to authenticate a message (for example, carry out a public-key operation). Furthermore, they imposed the restriction that the attack traffic attempted to overwhelm the computational resources of the receivers but was not able to prevent all legitimate traffic from being delivered. Under such an attack model, there is a disparity between the bandwidth used by the sender and by the attacker. While the sender broadcasts only the legitimate stream, the attacker must flood the

network with superfluous packets in order to mount an effective attack.

Selective verification exploits this asymmetry to protect the broadcast message stream from a DoS attack. Rather than process all arriving packets, a host using selective verification processes only a subset of the received packets. To ensure that legitimate messages are not lost, the sender transmits multiple copies of each message.

Upon first inspection, selective verification appears to worsen the DoS attack by increasing the amount of traffic received by the receiver. However, due to the asymmetry in the cost of sending versus processing messages, the large reduction in processing cost outweighs the increase in network traffic during an attack. For the sender, sending duplicate copies of a packet is cheap. Conversely, the receiver must exert significant computational resources to process messages. To alleviate this potential high cost, the receiver randomly drops packets with probability  $p$  (randomly dropping packets is, again, a cheap operation). On average, the receiver still may process multiple copies of legitimate messages (nonces may be used to quickly discard duplicates), however, DoS packets are also discarded with probability  $p$ . Thus, to sustain the same level of attack, the adversary must increase his traffic by a factor of  $1/p$ . While the legitimate senders must also send multiple copies of their messages, senders use only a small fraction of the receiver’s bandwidth during a DoS attack and transmitting duplicates adds little overhead to either the sender or the receiver. The primary cost of using selective verification is therefore borne by the attackers.

### 3.2. Selective Bin Verification

In this work, we explore the effectiveness of using selective bin verification to defend against DoS attacks. Here, we consider a typical client-server system in which multiple independent clients (senders) simultaneously access a single server (the receiver). Concurrently, the system may be subject to a DoS or DDoS attack in which the receiver is inundated with seemingly valid requests.

Like sequential verification, bin verification operates by processing only a randomly-selected subset of received messages. However, unlike sequential verification in which all messages are stored in the same queue, bin verification utilizes a set of  $n$  bins, labeled  $0, \dots, n - 1$ . When a legitimate sender transmits her request, she sends  $c$  copies of her message. Each copy includes a bin identifier ( $b$ ) and successive copies are numbered sequentially starting from a randomly chosen initial point. Upon receiving a copy of the request, the receiver queues the message in bin  $(b \bmod n)$ . Each sender hence “stripes” the  $n$  bins with packet copies. If the sender packets are lost with probability  $l$ , then the average number of sender packets per bin is  $(1 - l)c/n$ .

Attack packets are processed and assigned to bins in an identical fashion though, conservatively, we do not assume that attack packets are subject to loss.

After some fixed amount of time (which we call the *collection interval*), the receiver chooses  $k$  bins, where  $k \leq n$ . For example, a simple (and effective) technique is to choose the  $k$  bins that contain the least (but non-zero) number of queued messages. Regardless of how the bins are selected, the receiver proceeds by processing each message in the  $k$  bins. For each valid request (e.g., messages that are successfully authenticated through a digital signature), the receiver responds to the request according to the particular protocol being carried out<sup>1</sup>.

### 3.3. Sequential vs. Bin Verification

While sequential verification is effective at protecting an authenticated broadcast stream against DoS, bin verification is more appropriate for protocols in which multiple senders simultaneously send requests to a single receiver. In the latter case (which is the impetus of this paper), the binning technique increases the probability that a sender and the receiver will be able to participate in a successful exchange.

Consider a configuration in which we have  $n$  bins and  $m$  senders. Suppose each sender sends  $n$  copies. In the absence of network loss, we are guaranteed that by choosing a single bin we satisfy all  $m$  senders. The receiver's load is therefore  $m$ , or 1 packet per sender. On the other hand, in sequential verification, in order to generate an expected load of 1 packet/sender, the receiver needs to discard packets with probability  $(1 - 1/n)$ . Then the probability that none of the packets of a sender are received is roughly  $1/e$ . Thus, approximately  $m/e$  senders will have none of their packets received. In other words, with a comparable work load, we expect only 63.21% of the senders to now succeed as opposed to 100% with binning.

The calculations remain unchanged in the presence of a DoS attack. To scale down the attack by a factor of  $n$ , the receiver can inspect one of his  $n$  bins. Here, the load on the receiver is  $m + \gamma/n$ , where  $\gamma$  is the number of received attack messages. Assuming no network loss, all sender requests will be processed.

### 3.4. Memory Requirements of Selective Verification

For bin verification to be most effective, the receiver must have sufficient memory to enqueue all arriving messages during the collection interval. With diminishing prices for memory and increasing speeds of hard disks (i.e., for virtual paging), we believe that processing power rather

<sup>1</sup>Since multiple copies of a message may appear in the  $k$  bins, request identifiers may be used to both speed the processing of redundant messages and prevent non-idempotent requests from being executed multiple times.

than memory is more often the scarce resource. (Many commodity hard disks have transfer rates exceeding 100Mb/s and could therefore be used to buffer packets when memory becomes unavailable.) Moreover, for many protocols, particularly those that rely on expensive cryptographic operations and are thus more susceptible to DoS (e.g., X.509 [1] and SSL/TLS), the bottleneck is the receiver's computational resources. For example, according to OpenSSL benchmarks we conducted on a 2.4GHz Pentium IV, the maximum number of signatures that can be computed per second using a 2048-bit key is only 34.4 for RSA and 122.0 for DSA, far below the number of requests that can be transmitted in one second using a 100Mb/s connection.

While bin verification is effective for computationally intensive protocols, sequential verification is more appropriate for protecting against attempts to overwhelm the memory resources of the receiver. In such cases, the receiver can discard packets with a probability sufficient to ensure that the nondropped packets can be buffered in memory.

In the remainder of this paper, we make the assumption that there is sufficient memory to hold all incoming messages. That is, we concentrate on attacks in which processing power is the scarce resource and show how bin verification can alleviate the effects of these attacks.

## 4. Experiments

We have conducted a series of experiments to verify the ability of selective bin verification to mitigate the effects of DoS. In each experiment, multiple senders attempt to carry out a modified version of the X.509 two-pass protocol [1].

### 4.1. Modified X.509 Two-pass Protocol

$$\begin{array}{l} (1) \quad A \rightarrow B \quad : \quad cert_A, D_A, S_A(D_A) \\ (2) \quad B \rightarrow A \quad : \quad OK \end{array}$$

where  
 $cert_A$  is a certificate from a certificate authority used for authenticating  $A$ 's public key  
 $D_A$  is defined as  $(r_A, B, P_B(k_1))$   
 $P_B(k_1)$  denotes the encryption of  $k_1$  using  $B$ 's public key  
 $S_A(D_A)$  denotes the signing of  $D_A$  using  $A$ 's private key  
 $r_A$  is a nonce

**Figure 1. Modified version of the X.509 protocol**

The goal of the X.509 protocol is for a sender to securely transmit a symmetric key,  $k_1$ , to a receiver. As shown in Figure 1, a sender encrypts  $k_1$  using the receiver's public key and signs the result using its private key. The message is then sent to the receiver, where the receiver ensures that

the nonce is fresh, validates the signature (using the supplied certificate), and decrypts  $P_B(k_1)$ . If the receiver is successful in learning  $k_1$ , it transmits an OK message to the sender.

It is worthwhile to note that the cost of conducting the protocol is quite high for the receiver. For a request that contains a fresh nonce, the receiver must conduct an expensive signature check. An adversary who uses fresh nonces but sends random bits for  $D_A$  and  $S_A(D_A)$  forces the receiver to perform this operation. Thus, the protocol's high cost causes the receiver to be particularly vulnerable to DoS. We therefore use the modified X.509 protocol to measure the ability of bin verification to mitigate the effects of DoS.

## 4.2. Experimental Setup

The X.509 server (the receiver) was implemented using OpenSSL version 0.9.7. The server operated on a 800MHz Pentium III machine. Request initiators (senders) ran on a 2.4GHz Pentium IV and were constructed using POSIX threading and OpenSSL libraries. The sender software was capable of running approximately 80 concurrent connections with the receiver. In the cases where a greater number of senders were needed, a 700MHz Pentium III was also used. DoS attacks initiated from a 700MHz Pentium III computer. All machines ran Linux 2.4 and were connected using 100Mbps Ethernet network interfaces and switches.

All communication occurred over TCP/IP. Network loss was simulated at the sender by failing to send a request with probability  $l$ , which we call the *loss probability*. We chose to use TCP rather than UDP in order to more accurately control the loss rate visible to the application layer. DoS messages from the attack host experienced no loss.

The receiver software utilized 20 bins and a six second collection interval. Incoming messages were enqueued into a bin using the modulo technique described in Section 3.2. In all experiments, the collection interval was sufficient to capture all messages transmitted by the senders. A simulated loss probability of 0.20 was experienced by the senders. To compensate, each sender sent 25 copies of her requests. Each sender chose uniformly at random an initial bin from  $[0, 19]$  and proceeded by sending subsequent copies in a round-robin fashion.

After the collection interval, the receiver selected three bins. With the exception of the experiment described in Section 4.5, the three non-empty bins with the smallest number of messages were chosen. The receiver processed all messages in each bin (i.e., it verified the freshness of the nonce, validated the signature, and in the case of legitimate messages, decrypted  $P_B(k_1)$ ). If the message passed all checks, the receiver sent an OK message to the corresponding sender.

## 4.3. Failure Rate of Bin Verification

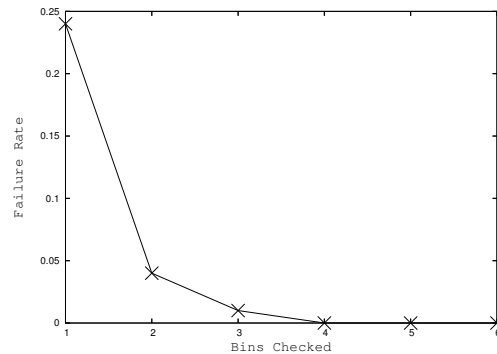
A sender's message may fail to be processed due to the simulated network loss probability (fixed in our experimentation at 0.20). After the collection interval has expired, the receiver processes messages only from a small subset of the nonempty bins. It is therefore possible that due to the loss probability, no copies of a particular request are present in any of the selected bins. We call this a *failure* and the fraction of unique sender requests (as opposed to sender copies) that end in failure the *failure rate*. Here, we show that by checking just a small number of bins, the failure rate becomes quite low.

For simplicity, we consider a bin selection algorithm in which the receiver chooses  $k$  random bins rather than the  $k$  smallest. Although this strategy is not optimal from the receiver's point of view (see Section 4.5 for an analysis of different DoS attacker techniques), it serves as a simple and useful baseline for understanding the anticipated loss rate of selective bin verification. Let  $l$  be the loss probability,  $c$  be the number of copies sent by a sender, and  $n$  be the number of bins used by the receiver. Additionally, we assume that the sender stripes her request across the  $n$  bins. We can then calculate the probability,  $f'$ , that a randomly chosen bin does not include a message from the sender as:

$$f' = \left(\frac{c \bmod n}{n}\right)l^{\lceil \frac{c}{n} \rceil} + \left(1 - \frac{c \bmod n}{n}\right)l^{\lfloor \frac{c}{n} \rfloor} \leq l^{\lfloor \frac{c}{n} \rfloor} \quad (1)$$

Since the probability that messages arrive in a given bin are independent, the failure rate,  $f$ , when  $k$  bins are examined is therefore

$$f \leq l^{\lfloor \frac{c}{n} \rfloor k} \quad (2)$$



**Figure 2. Failure Rate vs. Bins Checked (100 senders)**

To verify our analysis, we conducted an experiment using the X.509 testbed. For this experiment, 100 senders simultaneously sent X.509 requests. Here, however, the re-

ceiver used a more intelligent strategy and chose the smallest (rather than random) three bins. Although the failure rate is slightly higher for this technique as compared to the expected value calculated using Equation 2, the rate becomes quite negligible when checking a small number of bins with a sufficiently large number of senders. This trend can be seen in Figure 2, with the failure rate approaching 1% after only three bins are checked.

#### 4.4. Protection against DoS Attack

In this experiment, we show the efficacy of our selective bin verification approach against DoS attacks. We use as our metric the number of *inspections* carried out by the receiver. Here, we define an inspection to occur when the receiver processes a message, regardless of whether the message is legitimate, a duplicate, or invalid.

We compare the results of our binning technique against a straightforward implementation in which every message is considered. Let  $a$  be the attack rate (i.e., the number of attack messages arriving at the receiver per second),  $i$  be the collection interval (measured in seconds), and  $n$  be the number of bins. When binning is not used and each sender sends only one request, then the expected number of inspections required for  $m$  senders is

$$\text{insp}_{\text{nb}} = ai + m(1 - l) \quad (3)$$

where  $l$  is the loss probability of the sender.

For the binning technique, the  $m$  senders are expected to cause  $mc(1 - l)(\frac{k}{n})$  inspections, where  $c$  denotes the number of copies sent by each sender and  $k$  is the number of bins inspected by the receiver. Since the receiver only inspects  $k$  bins, the contribution of the DoS is  $ai(\frac{k}{n})$ . Thus, the expected number of inspections is

$$\text{insp}_{\text{b}} = \frac{k(mc(1 - l) + ai)}{n} \quad (4)$$

Thus, when the attack rate  $a$  is much larger than  $n$  and  $m = O(n)$ , then the attack rate is diminished by approximately a factor of  $k/n$ . For a receiver with 20 bins and  $k = 3$ , we would thus expect to process roughly 3/20 of the DoS packets. Furthermore, we note that Equation 4 is an upper-bound on the number of inspections since the receiver selects the  $k$ -smallest bins rather than  $k$  bins at random.

We conducted an experiment to confirm our analysis. Figure 3 shows the number of inspections for the binning and non-binning approaches. 50 senders simultaneously sent requests in both the non-binning and binning trials. For the non-binning trials, each sender sent only one copy of her message. As in all binning experiments, a six second collection interval was used. In addition, an adversary flooded the receiver with requests that appeared valid but that contained invalid signatures. These extraneous messages were striped

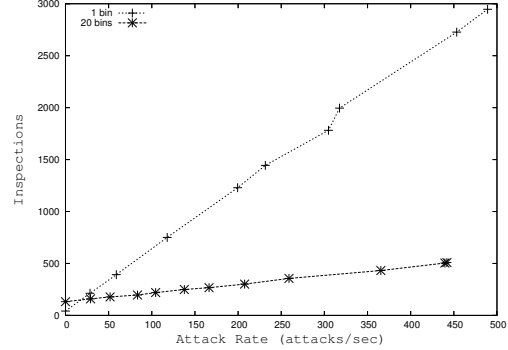


Figure 3. Inspections vs. Attack Rate

across all bins. Although both are linear, the slope resulting from the binning approach is significantly less than that of the non-binning technique. While the specific results shown in the Figure are a function of our chosen parameters (e.g., 50 senders, 20 bins, and 3 selected bins), we believe that our configuration is appropriate for real-world deployment and is effective at alleviating DoS.

#### 4.5. Bin Selection Algorithm

In the previous experiments, the receiver processed messages in the three smallest but non-empty bins. We call this technique the  $k$ -smallest approach. With a small number of senders, this approach can lead to a higher failure rate. For example, if a small number of senders each transmits 25 copies of their requests across the 20 bins with some loss probability, the selection strategy will favor bins that contain no messages from some of the senders. As a result, the failure rate for each sender is relatively high. With a large number of senders, the distribution of bin loads as a fraction of the total number of senders becomes more uniform, and hence the failure rate decreases.

To counter this effect and increase the probability that each sender's request is processed, we propose the  $\alpha$ -random approach for selecting  $k$  bins. Here, the receiver chooses the  $k - \alpha$  smallest but non-empty bins, where  $1 \leq \alpha < k$ . Let  $\mathcal{B}$  be the set of remaining, non-selected bins. The receiver proceeds by computing the mean bin load (i.e., the total number of received messages divided by the number of bins). It then uniformly at random chooses  $\alpha$  bins from the subset of  $\mathcal{B}$  which consists of bins that contain less than the mean bin load.

Figure 4 illustrates the failure rates for both bin selection strategies. As before, we let  $k = 3$ , and in the case of the  $\alpha$ -random selection algorithm, assign  $\alpha = 1$ . As can be seen in the Figure, the failure rate is significantly higher for the  $k$ -smallest selection technique with a small number of senders (e.g., less than 10). This effect is less pronounced

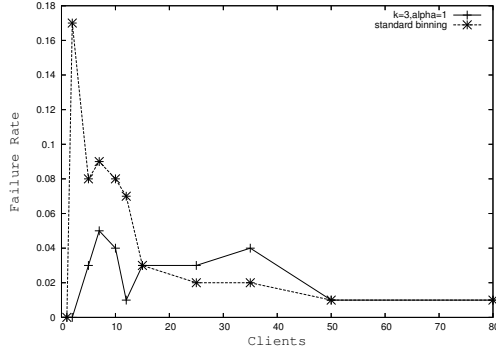


Figure 4. Failure Rate vs. Number of Senders

for the  $\alpha$ -random approach.

#### 4.6. Subset Attack

In Section 4.4, we described how selective bin verification can reduce the effect of DoS when the adversary stripes his attack across all bins. Here, we show that our binning technique is resilient even when the attacker constrains his attack to a chosen subset of bins. As before, we assume that the receiver uses the  $k$ -smallest approach. Furthermore, we denote the attacker's measure of success as the number of inspections he can force the receiver to undergo.

**Theorem:** The contribution of inspections due to DoS is maximized when the attack is evenly distributed across all  $n$  bins.

*Proof.* Let  $L(\sigma)$  be the total number of adversary packets in the  $S$  smallest bins, where  $\sigma$  is the attacker's chosen distribution function such that  $\sigma(i)$  denotes the number of DoS packets that the adversary sends to bin  $i$ . Also, let  $\bar{\sigma}$  be the equal distribution (i.e.,  $\forall i, j, |\bar{\sigma}(i) - \bar{\sigma}(j)| \leq 1$ ). For simplicity, we consider the case where  $\sum_{i=0}^{n-1} \sigma(i)$  is a multiple of  $n$  (that is,  $\forall i, j, \bar{\sigma}(i) = \bar{\sigma}(j)$ ). Since the  $k$ -smallest bins can never contain more messages than  $k$  times the average bin load (i.e.,  $\bar{\sigma}(i)$ ), then  $\forall \sigma, L(\sigma) \leq L(\bar{\sigma})$ .  $\square$

Lastly, we note that the senders' requests do not affect the efficacy of the attacker's chosen bin distribution technique. Legitimate senders evenly distribute their message copies among the  $n$  bins. Hence, the contribution of the senders' requests is constant and independent of the manner in which the DoS packets are distributed.

### 5. Conclusions and Future Work

Existing research concerning DoS has focused almost exclusively on preventing malicious packets from reach-

ing their intended targets. While such research is worthwhile and is useful for thwarting particular classes of attack, these approaches alone are not sufficient to protect service providers. Particularly for protocols in which the receiver has a high processing cost, traditional approaches fail to protect against attacks in which the adversary can overwhelm the receiver by sending malicious messages that appear entirely valid.

In this paper, we have investigated the ability of selective bin verification to protect services from attack, even when the DoS flood reaches the receiver. We show that bin verification is a promising technique for mitigating the effects of even large DoS attacks. Furthermore, we have analyzed the ability of bin verification to protect against attack under different configurations and attack strategies.

There are several interesting areas of future research relating to selective bin verification. Currently, the technique requires communication overhead even in the absence of an ongoing attack. Thus, an appropriate extension incorporates intrusion detection. For example, bin verification could be deactivated in the steady state and automatically enabled during an attack. A formal analysis of which protocols may best benefit from selective verification is another useful future research area. Finally, solutions that employ selective verification in concert with network-based DoS defenses are worthy of future study.

#### Acknowledgements

This work was partially supported by ONR Grant N00014-02-1-0715.

#### References

- [1] ITU-T (CCITT) recommendation X.509, Mar. 1988. Also ISO 9594-8.
- [2] S. Berinato. How a bookmaker and a whiz kid took on an extortionist - and won. *CSO Magazine*, May 2005. <http://www.csoonline.com/read/050105/extortion.html>.
- [3] CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks, Jan. 1998.
- [4] C. A. Gunter, S. Khanna, K. Tan, and S. S. Venkatesh. Dos protection for reliably authenticated broadcast. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2004.
- [5] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium*, February 2002.
- [6] P. Jelena and M. Greg. Attacking DDoS at the source, 2002.
- [7] D. Moore, G. Voelker, and S. Savage. Inferring internet denial of service activity. In *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., Aug. 2001. USENIX.
- [8] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23-24):2435-2463, 1999.

# On Design Tradeoffs between Security and Performance in Wireless Group Communicating Systems

Jin-Hee Cho, and Ing-Ray Chen  
 Department of Computer Science  
 Virginia Tech  
 {jicho, irchen}@vt.edu

**Abstract** - While security is of prime concern in secure group communicating systems in wireless networks, security mechanisms employed often have implication on the performance of the system. Recently model-based qualitative evaluation has been used for the evaluation of security protocols to quantify security properties in terms of intrusion tolerance using quantitative modeling techniques. However, most of the prior work focused only on measuring security properties, largely ignoring the performance impact of the security mechanisms introduced into the system. In this paper, we analyze the tradeoff between security and performance properties of an intrusion detection system (IDS) in a wireless group communicating setting. In particular, we analyze how often the IDS should perform intrusion detection to effectively trade security off for performance, or vice versa, for the system to satisfy the application security and performance requirements. Given the mean time to security failure (MTTSF) for the system to reach a failure state, and the response time per rekey operation for the wireless group communicating system as metrics, we identify the optimal intrusion detection rate under which the MTTSF metric can be best traded off for the response time metric.

**Key words:** Model-based evaluation, intrusion detection, key management, group rekeying, group communication, mean time to security failure, response time, performance analysis.

## 1. Introduction

Most of the early work in security emphasizes the prevention of attacks in system. Later most work focuses on system-level security mechanisms so that the system can perform its intended function through detecting and preventing malicious attacks. More recently, the notion of intrusion tolerance has been advocated to allow the system to continue performing its intended function despite partially successful attacks [1]. Most attempts to validate security mechanisms, however, have been qualitative by showing that the process employed to construct a system is secure. Since it is not practically feasible to construct a perfectly secure system, it is important to be able to qualitatively validate the efficacy of the system intended to be secure [1].

Many emerging mobile applications depend on the notion of secure group communication where mobile nodes can join or leave a group dynamically and group rekeying must be done so that only group members can communicate with each other using the secure key provided. In a wireless environment, it is important to reduce the communication overhead in group rekeying because of limited bandwidth and resources.

Researchers in the area of dependability analysis have attempted to extend quantitative analysis for dependability measures such as reliability and availability to security analysis for security measures with some success [2, 3, 4, 5, 6, 7, 8, 9]. Zhang et al. [16] analyzed several group rekeying algorithms in wireless environments and evaluated their performance characteristics. No intrusion was considered, however. Dacier et al. [9] proposed a novel approach to model the system as a privilege graph demonstrating operational security vulnerabilities and transformed the privilege graph into a Markov chain based on all possible successful attack scenarios. Jonsson et al. [8] presented a quantitative Markov model of attacker behaviors using data obtained from several experiments conducted over two years. They postulated that the process describing an attacker may be divided into multiple phases, such as learning, standard attack, and innovative attack. Goseva-Popstojanova et al. [6] presented a state transition model to depict the dynamic behaviors of intrusion tolerant systems. Their model includes a framework to define the vulnerability and the threat set. Madan et al. [3] employed a Semi-Markov Process (SMP) Model to evaluate the security attributes of an intrusion-tolerant system known as the SITAR system. Based on particular attack scenarios, states are related with failure of availability, data integrity, and data confidentiality. Steady-state analysis is used to obtain dependability measures such as availability. Transient analysis with absorbing states is used to obtain security measures such as mean time (or effort) to security failure (MTTSF) similar to the computation of the mean time to failure (MTTF) in reliability analysis. Wang et al. [7] utilized a higher-level formalism based on stochastic Petri nets (SPN) for security analysis of intrusion tolerant systems. In the area of intrusion tolerant systems, quantitative modeling techniques, particularly state-based stochastic methods [10], have been used to evaluate security properties.

All previous works cited above, however, often only focused on security measures without considering the impact of deploying security mechanisms on the performance of the system. We believe that the definitions and designs of security properties should reflect specific network and workload

environments and should take both security and performance requirements into consideration.

The objective of this paper work is to quantify security and performance properties of an intrusion detection system (IDS) in a wireless secure communicating system. Security and performance metrics are defined, based on which the effect of intrusion detection on security and performance attributes of the wireless group communicating system is analyzed, taking into account the presence of insider attacks. We analyze how often the intrusion detection activity of the IDS should be performed so as to effectively trade security off for performance of the system, or vice versa, to satisfy the application security and performance requirements.

This paper has two contributions. First, we develop quantitative analysis methods to analyze the tradeoff between security and performance in a wireless group communicating system in the presence of insider attacks and intrusion detection mechanisms, recognizing that security mechanisms often have great impacts on the performance property of the system. We develop a Stochastic Petri net (SPN) model to succinctly describe the attacker, the group communicating system, and the intrusion detection mechanism to evaluate the effect of intrusion detection on the security and performance properties of the system. We adopt SPN modeling so we can consider a general time distribution for an event, including using a fixed time interval to model the periodic intrusion detection. Second, when given a set of parameter values characterizing the operational conditions of the system, we identify the best intrusion detection rate under which we can effectively trade security off for performance (response time for providing services for secure group communications in this case), or vice versa, such that system designers can adjust the intrusion detection rate not only to satisfy the security and performance requirements, but also to optimize the metrics in the system.

The rest of the paper is organized as follows. Section 2 describes the system model, assumptions, and security and performance metrics defined. Section 3 describes the cost model and the parameterization process by which model parameters are given values. Further, we develop an SPN model in Section 3 to describe the behaviors of the group communicating system in the presence of attackers and intrusion detection so as to analyze the security and performance characteristics of the system. Section 4 presents numerical results obtained from evaluating the SPN model, and provides physical interpretations. Finally, Section 5 concludes this paper and outlines some future research areas.

## 2. System Models and Assumptions

### 2.1 Secure Group Communications in Dynamic Networks

An efficient way to achieve secure group communications is to use a symmetric key, called the *group key*, shared by group members. The group key can be distributed by a key server in wireless environments where a base station exists that can provide group key management services. The group key can also be agreed upon by group members by means of a group agreement protocol in a distributed environment where

there is no centralized key server. The group key is employed to encrypt the message sent by a member to the group; thus, only members of the group are able to decrypt and read group messages [13].

In a dynamic group setting where users can join or leave the group at any time, the group key needs to be rekeyed. There are the two main security properties commonly associated with rekeying [12, 14], namely, *forward secrecy* which ensures that an adversary who knows a contiguous subset of old group keys cannot identify subsequent group keys, and *backward secrecy* which ensures that an adversary who knows a subset of group keys cannot discover previous group keys. To maintain both backward secrecy and forward secrecy, the key server needs to perform rekeying (change the group key) whenever group membership changes [12,13] due to a new user joining or a current member leaving or being evicted.

### 2.2 Time for Performing a Rekey Operation

While the methodology developed in the paper can be generally applied to environments in which a centralized server does not exist, for ease of disposition we will assume that there exists a key server that applies a key distribution protocol to disseminate a new key upon group membership change events. This assumption can be relaxed by changing the parameterization process to consider a key agreement protocol for a set of distributed nodes to agree on a new group key. The only difference is to assign a different value to the “rekeying time” parameter representing the service time for performing a rekey operation. While many key distribution protocols exist, we consider the case that the key server uses the Logical Key Hierarchy (LKH) protocol [11] in a wireless environment by which the key server maintains a key tree to efficiently update the group key after a join or leave event.

A key update operation upon a member leave event requires a message of length  $2k \log_2 N$  bits (where  $k$  is the length of a key, and  $N$  is the number of members), while a key update operation upon a new member join event requires a message of length  $k(2 \log_2 N - 1)$ . The main benefit of LKH [11] is that it only requires a broadcast message size that is logarithmic in the number of group members.

### 2.3 System Assumptions

We make the following assumptions regarding the workload and operational characteristics in a wireless group communicating system:

- We assume that the interarrival times of the intrusion detection, compromising process, join and leave requests, are exponentially distributed with the rates of  $\theta$ ,  $\lambda c$ ,  $\lambda$ , and  $\mu$ , respectively. The assumption of exponential distribution can be relaxed easily by defining other time distributions and evaluating the model using SPNP v6.
- The time to perform a rekey operation upon a membership change event is dominated by the network communication cost for broadcasting the rekey message based on the LKH protocol in a wireless environment.

The computational time for maintaining the key tree and for calculating new keys along a new key path in the key tree in a rekey message is relatively small compared with the wireless network communication cost.

- The system enters a security failure state when a compromised but undetected member requests and obtains data using the group key. The system is in a failure state because data have been leaked out to a compromised node. On the other hand, if a member node is detected as compromised by the intrusion detection system (IDS), the system won't allow the member node to request data any more and will evict the member immediately. There is no recovery mechanism available in the system that can recover a compromised member into a trusted member node. Initially, all nodes are trusted member nodes.
- The system also enters a security failure state when more than 1/3 of member nodes are compromised but undetected by the IDS. We assume the Byzantine failure model [15] such that when more than 1/3 of member nodes are compromised, the system is compromised.
- We also consider that the IDS may not correctly detect compromised nodes. Thus, we consider the cases for false positive (detecting trusted member nodes as compromised member nodes) and false negative (detecting compromised member nodes as trusted member nodes) by the IDS.

## 2.4 Metrics

We define two metrics below to measure security and performance of secure group communicating systems in wireless environments:

**MTTSF (Mean Time to Security Failure):** This metric indicates the average time elapsed to reach a security failure state. A higher MTTSF is desirable. For secure group communicating systems, we say a security failure occurs when data have been leaked out to a compromised member node, or more than 1/3 of the member nodes have been compromised. Note that illegal data leak-out only occurs when a compromised but undetected member requests and subsequently obtains data using the group key.

**Service Response Time ( $\bar{R}$ ):** This metric indicates the average response time for a rekey operation, accounting for both the queuing and communication delays for the system to service a rekey operation due to a join/leave event. A lower  $\bar{R}$  is more desirable.

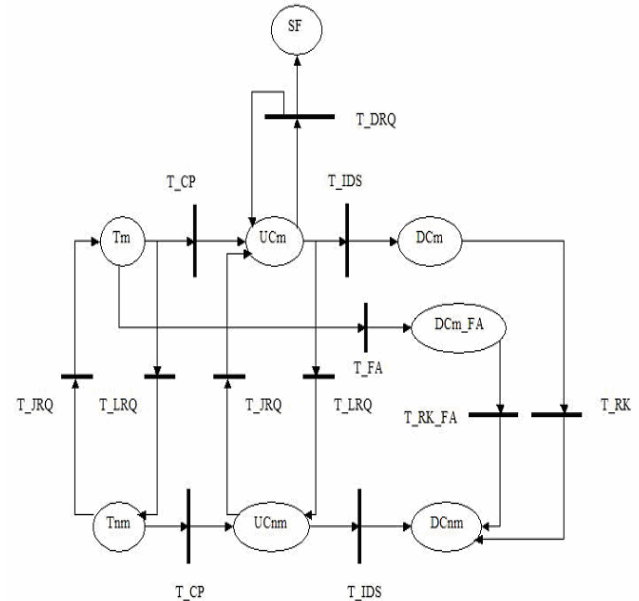
## 3. Performance Model

We develop a stochastic Petri net (SPN) model as shown in Figure 1 to describe the system behavior under insider attacks and periodic intrusion detection activities with the objective of assessing MTTSF and  $\bar{R}$  of the system. Table 1 summarizes the model parameters used. All transitions in the SPN model are timed transitions, meaning that the transition

time is non-zero and follows a probability distribution, e.g., exponentially distributed. Further, each transition can be associated with an enabling function that guards the firing of the transition depending on the current state of the system.

**Table 1: Model Parameters.**

Symbol	Meaning
$\lambda$	Arrival rate of join requests
$\mu$	Arrival rate of leave requests
$\theta$	Intrusion detection rate by the IDS
$\lambda_c$	Rate at which nodes are compromised
Rdrq	Rate for data request by compromised member nodes undetected by the IDS
Rfa	Rate for generating false positive by the IDS
Tcm	Communication time for broadcasting a rekey message
J	Length of each key value in the key tree (bits)
BW	Network bandwidth (Kbps)
N	Total number of member nodes initially.
MTTSF	Mean Time To Security Failure
$\bar{R}$	Average response time for a rekey operation



**Figure 1: SPN Model.**

Below we explain how the SPN model is constructed by considering the system's lifecycle. Initially, we assume all members are trusted; thus we place all members in place Tm as tokens. Trusted members may become compromised through insider attacks with a node-compromising rate of  $\lambda_c$ . This is modeled by firing transition T\_CP and moving one token at a time (if it exists) from place Tm to place UCm. Tokens in place UCm represent compromised but undetected member nodes. We consider the system as having experienced a security failure when data are leaked out to compromised members using the compromised group key. Thus, when a token exists in place UCm, the system is considered to be in a security vulnerable state. We assume that a compromised and undetected member will attempt to compromise data from other members in the group with a rate of Rdrq. This is

modeled by transition  $T\_DRQ$ , the firing of which will move a token into place SF, at which point we regard the system as experiencing a security failure, i.e., the system fails when  $mark(SF) > 0$  where the  $mark(SF)$  function returns the number of tokens contained in place SF.

A compromised node in place  $UCm$  may be detected by the IDS before it compromises data in the group communicating system. The intrusion detection activity of the system is modeled by transition  $T\_IDS$  with rate  $\theta$ . Whether the damage has been done by a compromised node before the compromised node is detected depends on the relative magnitude of the node-compromising rate ( $Rdrq$ ) vs. the IDS detection rate ( $\theta$ ). Thus, it is of interest to analyze the effect of  $Rdrq$  vs.  $\theta$  on  $MTTSF$  and  $\bar{R}$ . When transition  $T\_IDS$  fires, a token in place  $UCm$  will be moved to place  $DCm$ , representing a compromised, undetected node now becomes a compromised, detected node.

The IDS system has a non-zero probability of false-negative (failing to detect compromised members) and false-positive (incorrectly considering trusted member nodes as compromised member nodes). For the false-negative case, compromised nodes would remain in place  $UCm$  because the IDS could not detect them. A false-positive event is modeled by moving a trusted member in place  $Tm$  to place  $DCm\_FA$  after transition  $T\_FA$  fires with rate  $Rfa$ .

A compromised member node identified by the IDS, whether it is falsely identified or not, will be evicted by the group communicating system. This is modeled by firing transition  $T\_RK$  for evicting detected compromised members, and transition  $T\_RK\_FA$  for evicting falsely detected compromised members. The time for broadcasting a rekey message is  $Tcm$ , so the rate at which transition  $T\_RK$  or  $T\_RK\_FA$  fires is  $1/Tcm$  (by broadcasting a rekey message to evict a member). After an identified compromised member is evicted, it becomes a compromised non-member and is put in place  $DCnm$ . We assume the system does not contain recovery mechanisms to repair compromised nodes, so once a node is put in place  $DCnm$ , it remains there permanently. This is modeled by having no outgoing arcs from place  $DCnm$ .

The group communicating system is characterized by member join and leave events, with rates of  $\lambda$  and  $\mu$  respectively. Both trusted and undetected compromised nodes can join and leave the group at will. This is modeled by transitions  $T\_JRQ$  and  $T\_LRQ$  with firing rates  $\lambda$  and  $\mu$ , respectively.

Finally, the system is considered as experiencing a security failure if either of two security failure conditions described in Section 2.4 is met. This is modeled by making the system enter an absorbing state when one of two conditions is met. In the SPN model, this is achieved by associating each transition in the SPN model with an enabling function that returns false (thus disabling the transition from firing) when either condition is met, and true otherwise, i.e., if  $mark(SF) > 0$  or  $isBF()$  then return false; else return true. where  $mark(SF) > 0$  is true for the first security failure condition that data have been leaked out to compromised members, i.e., when there is a token in place SF, and  $isBF()$  is true for the second security failure condition when more than 1/3 of member nodes are compromised, i.e., when the following condition is true:

$$\frac{\text{The total number of UCm}}{\text{The total number of Tm} + \text{The total number of UCm}} > \frac{1}{3}$$

### 3.1 Parameterization

Here we describe the parameterization process, i.e., how model parameters are given values reflecting the operational and environment conditions of the system. First, we describe how  $Tcm$ , the reciprocal of which is the rate of transition  $T\_RK$  and transition  $T\_RK\_FA$ , is parameterized. Recall that  $Tcm$  is the communication time required for broadcasting a rekey message for a join or leave event. It is calculated with the following formula: if ( $N < 1$  or  $N = 1$ ) then  $Tcm = J/BW$ ; else  $Tcm = (2J \log N)/BW$ . Here  $N$  is the current number of member nodes,  $J$  is the length of each key, and  $BW$  is the wireless network bandwidth. We assume that the size of the rekey message is at least  $J$  when the current number of members is zero or one. The rekey message is of size  $2J \log N$  when the current number of members is greater than 1.

Certain model parameters are design parameters for which we vary their values to analyze their effects on system performance. We vary the intrusion detection rate ( $\theta$ ) between 0.5 and 32 and the node-compromising rate ( $\lambda c$ ) between 0.2 and 5 to analyze the sensitivity of system performance on these two parameters. While varying other less critical model parameters to analyze their effects also makes sense, in most cases we fix their values so as to manifest the effects of design parameters such as  $\theta$  and  $\lambda c$ . The default values used are  $\lambda = \mu = Rdrq = Rfa = 0.1$ ,  $J = 64$  bites,  $BW = 1$  Kbps (for wireless bandwidth) and  $N = 20$ .

### 3.2 Performance Metric Calculations

$MTTSF$  can be obtained using the concept of *mean time to absorption* (MTTA) in the SPN model. Specifically, we use a reward assignment such that a reward of 1 is assigned to all states except absorbing states. That is, the reward assignment is done with the following reward assignment function: if  $mark(SF) > 0$  or  $isBF()$  then return 0; else return 1.

Then the MTTA or the  $MTTSF$  of the system is simply the expected accumulated reward until absorption,  $E[Y(\infty)]$ , defined as:

$$E[Y(\infty)] = \sum_{i \in S} r_i \int_0^{\infty} P_i(t) dt$$

Where  $S$  denotes the set of all states except the absorbing states,  $r_i$  (reward) is 1 for those states, and  $P_i(t)$  is the probability of state  $i$  at time  $t$ .

$\bar{R}$ , the average service response time for a rekey operation, can be calculated by the time-averaged value of  $R(t)$  over a period of time  $t$ , where  $R(t)$  is the instantaneous response time at time  $t$ , which can be computed as follows:

$$R(t) = \sum_{i \in S} [P_s(t) \times R_s(t)]$$

where  $S$  is the set of all states,  $P_s(t)$  is the probability of state  $s$  at time  $t$ , and  $R_s(t)$  is the amount of time required to complete a newly arriving rekey operation given that the system is in state  $s$  at time  $t$ , computed as follows:

$$R_s(t) = Tcm + Tcm(a + b)$$

where  $a$  is the number of compromised member nodes correctly identified by the IDS, and  $b$  is the number of compromised member nodes incorrectly identified by the IDS due to false positive events.

Once  $R(t)$  is obtained,  $\bar{R}$  can be obtained by accumulating  $R(t)$  over time until the system has reached a security failure state and then dividing the cumulative  $R(t)$  by the lifetime of the system, i.e., the MTTSF, as follows:

$$\bar{R} = \frac{\int_0^{MTTSF} R(t) dt}{MTTSF}$$

## 4. Results and Analysis

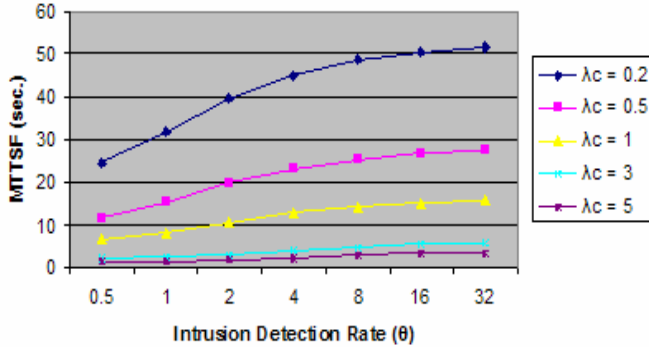


Figure 2: MTTSF vs. Intrusion Detection Rate ( $\theta$ ) with varying  $\lambda c$ .

We present numerical data obtained from evaluating the SPN model and discuss the physical meaning. We first examine the effect of the intrusion detection rate ( $\theta$ ) on MTTSF and  $\bar{R}$ . We then show that there is a range of  $\theta$  that could effectively trade the acceptable security level off for an improved response time ( $\bar{R}$ ).

### 4.1 MTTSF vs. Intrusion Detection Rate ( $\theta$ )

Figure 2 shows the effect of the intrusion detection rate ( $\theta$ ) on MTTSF under various node-compromising rate ( $\lambda c$ ) values. As we can see from Figure 2, higher  $\theta$  generates better MTTSF. This means that if the IDS detects compromised nodes more frequently, the longer time it would take for the system to reach a security failure state. In addition, as  $\lambda c$  becomes higher, MTTSF becomes shorter because a higher  $\lambda c$  causes more compromised nodes in the group, thus causing

the system more likely to reach a security failure state. Further, we observe that when  $\lambda c$  is low, the effect of  $\theta$  on MTTSF is more pronounced. Thus, the IDS is effective only when the node-compromising rate  $\lambda c$  is below a threshold value, e.g., when  $\lambda c = 5$ , the IDS is not effective over a range of the intrusion detection rate.

### 4.2 $\bar{R}$ vs. Intrusion Detection Rate ( $\theta$ )

Figure 3 shows the effect of the intrusion detection rate ( $\theta$ ) on  $\bar{R}$  as  $\lambda c$  varies. We observe that when  $\lambda c$  is high (top curves) and there are more compromising nodes in the system, increasing  $\theta$  actually would decrease  $\bar{R}$ . The reason is that as the detection rate increases, the system will detect compromised nodes and evict them with a higher probability. Thus, an incoming member join/leave operation will suffer a longer waiting time while the system is busy evicting compromised nodes detected. On the other hand, when  $\lambda c$  is low,  $\bar{R}$  is largely insensitive to the increase in  $\theta$ . Thus there is a threshold value of  $\lambda c$  below, which  $\bar{R}$  is relatively insensitive to  $\theta$ .

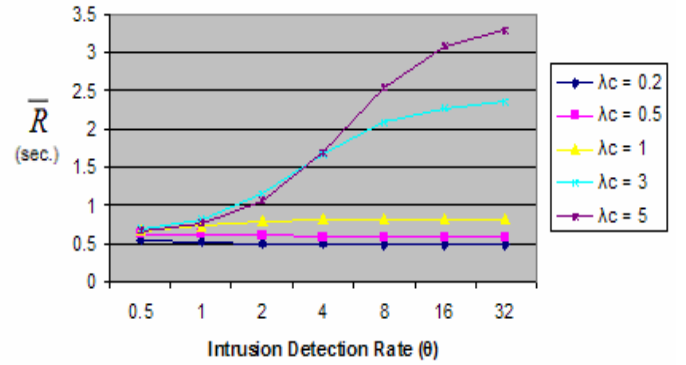


Figure 3:  $\bar{R}$  vs. Intrusion Detection Rate ( $\theta$ ) with varying  $\lambda c$ .

### 4.3 Identifying Acceptable Intrusion Detection Rate ( $\theta$ )

We could identify a proper intrusion detection rate ( $\theta$ ) to satisfy application-level security and performance requirements. The security requirement would be specified in terms of a lower bound of MTTSF, while the performance requirement would be specified in terms of the response time per member join/leave operation. The data shown in Figure 2 allow us to identify the lower bound of the intrusion detection rate ( $\theta$ ) to satisfy the imposed MTTSF lower-bound requirement. For example, when an application requires at least 12 sec of MTTSF at  $\lambda c = 1$ , the lower bound of  $\theta$  would be around 4. On the other hand, Figure 3 allows us to find the upper bound of the intrusion detection rate ( $\theta$ ) to satisfy the imposed response time requirement. For instance, when a minimum acceptable response time is 800 ms, the upper bound of  $\theta$  would be around 8 for  $\lambda c = 1$ . Therefore, the acceptance range for the intrusion detection rate  $\theta$  would be [4, 8] when  $\lambda c = 1$  that will meet both the security and performance requirements by a specific application. When  $\lambda c$

is low,  $\bar{R}$  is relatively insensitive to  $\theta$  while MTTSF is just the opposite, so we could use a higher value of  $\theta$  to increase MTTSF while satisfying  $\bar{R}$ . Conversely, when  $\lambda c$  is high, MTTSF is relatively insensitive to  $\theta$  while  $\bar{R}$  is just the opposite, so in this case, we should lower the value of  $\theta$  to satisfy  $\bar{R}$  while satisfying MTTSF.

## 5. Conclusion and Future Work

In this paper, by means of a Petri net model we demonstrated the intrinsic tradeoff between security (measured by the mean time to security failure or MTTSF) and performance (measured by the response time) of wireless group communicating systems. We showed that in general as the intrusion detection rate increases, MTTSF increases while the response time decreases. However, there exists a minimum threshold value of the node-compromising rate below which the response time is insensitive to the intrusion detection rate, and there exists a maximum threshold value of the node-compromising rate above which MTTSF is insensitive to the intrusion detection rate. We demonstrated how the system can adjust the intrusion detection rate to maximize MTTSF while satisfying the response time, or to minimize the response time while satisfying MTTSF.

To utilize the analysis results obtained, one can test a range of possible values of model parameters and build a table at static time listing the selection of the intrusion detection rate that can optimize the response time and/or MTTSF, when given the application requirements. Then at runtime, the system can perform a table lookup operation to select the proper intrusion detection rate based on statistical information collected periodically to parameterize the model parameters.

A future research direction is to apply hierarchical modeling techniques [1, 10] to allow us to solve a larger system. Another direction is to extend the research to systems without a key server such as in mobile ad hoc networks. We are currently investigating the effects of security vulnerability and the false alarm of the IDS on the response time and MTTSF.

## References

- [1] D.M. Nicol, W.H. Sanders, and K.S. Trivedi, "Model-Based Evaluation: From Dependability to Security," *IEEE Transactions on Dependability and Secure Computing*, Vol. 1, No.1, January-March 2004.
- [2] B.B. Madan, K.G.E. Popstojanova, K. Vaidyanathan, and K.S. Trivedi, "A Method for Modeling and Quantifying the Security Attributes of Intrusion Tolerant Systems," *Performance Evaluation*, Vol. 56, No. 1-4, 2004, pp. 167-186.
- [3] B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K. Trivedi, "Modeling and Quantification of Security Attributes of Software Systems," In *International Conference on Dependable Systems and Networks*, 2002, pp. 505-514.
- [4] S. Singh, M. Cukier, and W.H. Sanders, "Probabilistic Validation of an Intrusion-Tolerant Replication System," *International Conference on Dependable Systems and Networks*, June 2003, pp. 616-624.
- [5] F. Stevens, T. Courtney, S. Singh, A. Agbaria, J.F. Meyer, W.H. Sanders, and P. Pal, "Model-Based Validation of an Intrusion-Tolerant Information System," *23rd Symposium Reliable Distributed Systems*, 2004.
- [6] K. Goseva-Popstojanova, F. Wang, R. Wang, F. Gong, K. Vaidyanathan, K. Trivedi, and B. Muthusamy, "Characterizing Intrusion Tolerant Systems Using a State Transition Model," In *DARPA Information Survivability Conference and Exposition*, Vol. 2, 2001, pp. 211-221.
- [7] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi, "Security Analysis of SITAR Intrusion Tolerance System," *2003 ACM Workshop on Survivable and Self-regenerative Systems*, October 2003.
- [8] E. Jonsson and T. Olovsson, "A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior," *IEEE Transactions on Software Engineering*, Vol. 23, No. 4, April 1997, pp. 235-245.
- [9] M. Dacier, Y. Deswarte, and M. Ka nliche, "Quantitative Assessment of Operational Security: Models and Tools," Technical Report 96493, Laboratory for Analysis and Architecture of Systems, May 1996.
- [10] R.A. Sahner, K.S. Trivedi and A. Puliafito, *Performance and Reliability Analysis of Computer Systems*, Kluwer Academic Publishers, 1996.
- [11] Adrian Perrig and J.D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*, Kluwer Academic Publishers, 2002.
- [12] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam, "Secure Group Communications Using Key Graphs," *ACM SIGCOMM*, 1998.
- [13] Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda, Simon S. Lam, "Batch Rekeying for Secure Group Communications," *10th International World Wide Web Conference on World Wide Web*, 2001.
- [14] M. Steiner, G.Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 8, 2000, pp. 769-980.
- [15] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. on Programming Languages and Systems*, Vol. 4, No. 3, 1982, pp. 382-401.
- [16] C. Zhang, B. DeCleene, J. Kurose, and D. Towsley, "Comparison of inter-area rekeying algorithms for secure wireless group communications," *Performance Evaluation*, Vol. 49, No. 1-4, 2002, pp. 1-20.

# Detecting and Filtering Instant Messaging Spam – A Global and Personalized Approach

Zhijun Liu, Weili Lin, Na Li, and David Lee  
*Department of Computer Science and Engineering*  
*The Ohio State University, Columbus, Ohio*  
*{liuzh, linwei, lina, lee}@cse.ohio-state.edu*

**Abstract**— While Instant Message (IM) is gaining its popularity it is exposed to increasingly severe security threats. A serious problem is IM spam (spim) that is unsolicited commercial messages sent via IM messengers. Unlike e-mail spam (unsolicited bulk e-mails), which has been a serious security issue for a long time and a number of techniques have been proposed to cope with, spim has not received adequate attention from the research community yet, and traditional spam filtering techniques are not directly applicable to spim due to its presence information and real time nature.

In this paper, we present a new architecture for detecting and filtering spim. With the unique infrastructure of IM systems spim detection and filtering can be achieved not only at the client (receiver) side – for a personalized filtering – but also at the server side and various IM gateways – for a global filtering. Our technique integrates a number of mature spam defending techniques with modifications for IM applications, such as Black/White List, collaborative feedback based filtering, content-based technique, and challenge-response based filtering. We also design and implement new techniques for efficient spim detection and filtering, including filtering methods based on IM sending rate, content based spim defending techniques, fingerprint vector based filtering, text comparison filtering, and Bayesian filtering. We provide an analysis of their performances based on experimental results.

**Index Terms**— instant messaging, spam, spim

## I. INTRODUCTION

Instant Messaging [1] (IM), as an emerging communication technology, provides near real-time message delivery and rich presence information services. However, IM spam (spim) that is unsolicited commercial messages sent via IM messengers has begun to hit IM in large quantities since 2002. E-mail spam of unsolicited junk e-mails is known to be one of the major security problems of e-mail systems and a substantial amount of works has been done to defend against e-mail spam yet not spim. Between 2002 and 2003, the volume of spim has doubled to 500 million messages and more than 5-8% of all corporate IM today is spim [2]. By 2008, IM users are estimated to get an average of 25 IM spam messages per day. Since the IM messages are delivered almost in real time and often the user accepts the messages by default, IM spam may cause more severe damages than e-mail spam.

IM spam and e-mail spam have a lot in common in terms of content. Similar to e-mail spam, IM spam mainly consists of commercial advertisements. According to Radicati report, 70% of spim messages contain links to pornographic websites, around 12% convey “get rich” schemes; product sales account for 9%, and loans or finance messages are at 5%. However,

most of the existing anti-spam techniques are not suitable for spim filtering because of the differences in their underlying technology and system infrastructures.

In order to effectively eliminate spim we present a new architecture for its detecting and filtering. With the unique infrastructure of IM systems, spim detection and filtering can be achieved not only at the client (receiver) side but also at the server side and various IM gateways. Our technique integrates a number of mature spam defending techniques with modifications for IM applications. On the other hand, we design and implement new methods for efficient spim detection and filtering. We propose a personalized spim detection and filtering architecture that enables cooperates to apply different anti-spim technologies. Furthermore, in our architecture, filters can be deployed at IM client, server and gateway sides to eliminate spim more effectively. To the best of our knowledge, our work is the first that proposes an integrated architecture for spim detection and filtering.

## II. RELATED WORKS

Besides the proprietary protocols (MSN, Yahoo!, AOL etc.), current IM has two standards: XMPP [3, 4] (Extensible Messaging and Presence Protocol), which is a typical client-server protocol and defines an open XML-based protocol for extensible IMP application; and SIMPLE [5] (Simple Instant Messaging and Presence Service), which is a server broker architecture and is being developed by the SIMPLE Working Group at the IETF. We mainly focus on the spim filtering in server proxy architecture that is more popular in current IM systems; similar techniques can be used for SIMPLE.

Before we exploit the spim defending technique, we first investigate existing spam detection and filtering techniques. There are a broad range of solutions designed to mitigate the spam problem. However, no single technique could solve the spam problem completely and each stands out in its own ways.

- Black/White List: A general list-based spam blocking technique that eliminates e-mails from the Black List (known spammer addresses or malicious mail server IP addresses) and only allows the e-mails from the White List.
- Content based filtering: Keyword based filtering, simply spots any e-mail that contains some keywords that are commonly used in spam; Signature based filtering identifies spam by comparing the fingerprint [6] or digestion of incoming e-mails to that of known spam; Rule based Filtering analyzes e-mails and assigns a score to each keyword, where the total points the e-mail gets will determine its destiny; Statistics based filtering, also called Bayesian filtering [12], assigns frequency-based probability to words as spam indicators and aggregates the

single word probabilities.

- Response & Challenge based filtering: It tries to shift part of the responsibility to the sender. Whenever an e-mail is sent from an unknown person, it will challenge the sender by a small image, a piece of audio or a question before the e-mail delivery.
- Collaborative spam filtering: Collect spam information based on the user feedback.
- Distributed P2P based approaches [6, 10]: Peer-to-peer architecture is used to collaboratively share the knowledge about spam and control the spam information propagation.

On the other hand, spim began to attract people’s attention in the beginning of 2004. The current spim defending techniques are still in a primitive stage:

- Blocking incoming messages from unknown senders based on the White List
- User blocks spimmer based on the Black List
- Limit the automatic IM user registration by including a registration process and an image verification test that automated systems cannot pass.

Most current public IM services, such as MSN and Yahoo!, only allow delivering messages to the users from of While list of their buddies. This approach is helpful but is not an ultimate solution because it significantly constrains the applications of IM and it is not suitable for business users such as customer service representatives who are dealing with new customers not on the White List yet. Similar to e-mail spam, the individual blocking spimmer is not efficient and the limited IM user registration could only mitigate the spim problem to some extent. Hence, the current spim defending techniques are not adequate.

Although spam and spim have a lot in common, there are significant differences between them.

- 1) IM message is relatively short. In contrast, e-mail message can be long and contain large attachments. Hence, IM traffic is not as significant as e-mail traffic.
- 2) IM client maintains a TCP connection to IM server and the message delivery is almost in real time. Differently, e-mail has no such real time requirement. Therefore, lots of offline spam defending techniques are not appropriate for defending against spim. The challenge of defending against spim is how to effectively detect and filter spims while guaranteeing the real time services.
- 3) Due to the small message size consumption of storage space by offline spim on the IM servers is not as much of a serious concern as that in e-mail servers for spam.
- 4) Most of e-mail spams are caused by SMTP open relay where spammers take advantage of mis-configured SMTP servers for a free ride to send out a large amount of junk e-mails. In the case of IM, this is not a concern any more. However, sophisticated spimmers can “compromise” IM account with malwares, which will harvest the victim’s Buddy List and send out spims to the victim’s buddies. This kind of spim attack is more like malicious “chain letter”, but its damage is more serious because IM messages are sent out in near real time.

Given the significant differences between spam and spim, it deserves a serious investigation to defend against spim for IM system.

### III. OUR MODEL, ARCHITECTURE AND METHODS

We consider the XMPP IM and investigate an open IM architecture where an IM user allows messages from unknown users. We defend IM against the following spim:

- 1) Unsolicited junk IM messages, usually advertisements;
- 2) DoS attacks: flooding with unwanted messages, with the intent of crashing the IM client/server or making the client/server unstable or even hang.

After presenting the architecture, we discuss various spim filtering techniques, which can be implemented and deployed either at the server/gateway side or at the client side, providing the flexibility for both a global system-wide and also a personalized spim defending system.

#### A. Architecture

From an analysis of the characteristics of spim, we propose a hierarchical detection and filtering architecture as shown in Fig. 1. It integrates spam filtering techniques, including White/Black List, challenge-response based filtering, content-based technique, and collaborative feedback based filtering. More importantly, it supports our new techniques: filtering based on limiting IM sending rate and server/gateway side filtering.

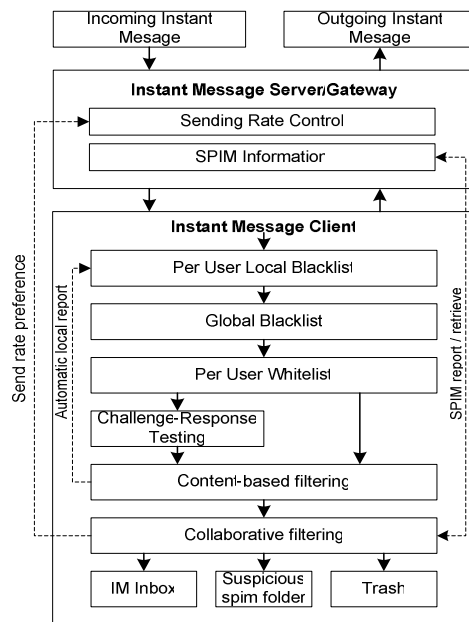


Figure 1: Personalized Spim Detection and Filtering Architecture

Note that e-mail server is on the client/user side and relays e-mail messages asynchronously and that IM client side system plays a role that is part of the roles of both e-mail server and client/user. However, IM server and gateway are the components in IM global systems and have no equivalent counterparts in e-mail systems. On the other hand, IM server/gateway and client forward messages in real time, and IM gateways forwards messages between different IM service systems.

Therefore, we can take advantage of the characteristics of IM architecture and deploy spim filters, which the architecture of e-mail system cannot support. For instance, we can enforce sending rate control at IM server/gateway side and that

provides an efficient way to eliminate spim and DoS attacks. On the other hand, due to the requirement of real time delivery of IM messages, most of the server based e-mail spam defending techniques are inadequate for spim. Instead, we install filters at IM client side, which is a counterpart of e-mail client yet with less storage requirements. As we shall show that the spim traffic is not a major concern since it can be managed by the sending rate control method at the server/gateway side, and we can handle spim at client side without consuming much network bandwidth.

### B. Server/Gateway Side Sending Rate Control

Often spim attacks are on destination IM users as well as IM servers/gateways by flooding them with unwanted messages. We apply a sending rate control filtering at the server/gateway side. It monitors its own workload and the traffic load of the network to alert and limit the abnormal users whose sending rate is greater than a certain threshold; messages sent out by abnormal users will be discarded by the server. This mechanism can reduce the instantaneous traffic burst in the network, constrain malicious users, and effectively defend against DoS attacks to IM server/gateway and client. Obviously, such effective global DoS attack defense mechanism can only be devised at IM system with its system architecture while it is infeasible to do so for e-mail system.

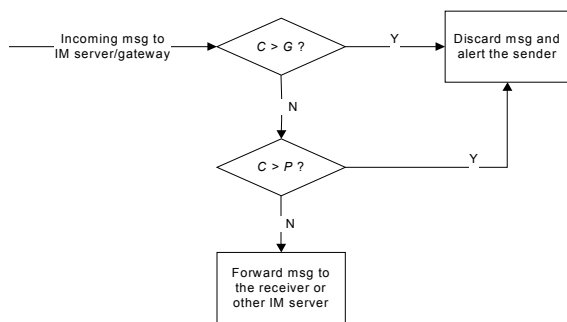


Figure 2: Sever/Gateway Side Sending Rate Control

Besides the sending rate control by the server/gateway, our architecture also provides users with personalized sending rate control mechanism that is highly desirable by PDA/mobile phone users with very limited available bandwidth. Each user can specify the corresponding sending rate threshold collaboratively for each buddy on his Buddy List. As shown in Fig.2, whenever a server/gateway receives a message, it will compare the sender's current sending rate  $C$  with the global sending rate threshold  $G$  and the receiver's personalized sending rate threshold  $P$ . One conservative global sending rate  $G$  could be roughly estimated by  $G = B / (N \times S)$ , where  $B$  is the maximum bandwidth for IM messages delivery,  $N$  is the number of online IM users and  $S$  is the average message size. If the sender's sending rate  $C > \min \{G, P\}$ , the message is discarded, and the server/gateway alerts the sender accordingly. Under certain circumstances, to avoid eliminating valid responding messages in high volume a sender's sending rate  $C$  may not include the responding messages; at server side initializing and responding messages can be distinguished and monitored.

### C. Client Side Filter Integration

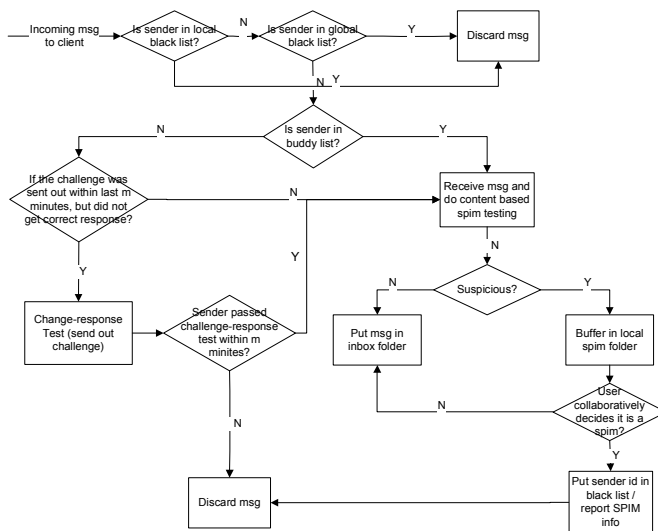


Figure 3: Client Side Spim Detection and Filtering

At the client side, five filters constitute a filtering pipeline, which processes the incoming message by passing it through. After analyzing the error rate and filtering efficiency, we determine the filtering order in our architecture as in Fig. 1. Among all of the filtering techniques, the Black List, which has the fastest running speed, is mainly produced by user feedbacks, thus it has the least error rate but is most efficient. In our simulation, we create a Black List based on the number of the occurrences of each e-mail address in a spim corpus; when the number of the occurrences reaches a threshold, the sender address is added to the Black List. Our experiment shows that up to 64% spims are filtered out when we set the threshold to 10. Hence the Black List is most effective to catch the well known malicious users at the very front. Content-based filtering is more time consuming due to the comparison operation, therefore, it is after the Black List filtering. Our analysis and experiments show that this ordering of different filtering techniques has the best performance to eliminate spim. A flowchart of the client side spim detection and filtering is displayed in Fig. 3. We now elaborate on some filtering process.

#### 1) Black List / White List Filtering

There are two different kinds of Black Lists, the local Black List and the global Black List. The global Black List comes from the server and maintains a list of malicious users who have or may send unsolicited junk or flooding messages. The users can also flexibly customize their own local Black List. Based on the Black List mechanism, the client can filter out all the messages sent by users who are on the Black List. This is a typical technique against either spam or spim. Only the users in the White List (Buddy List) who have been authorized can send message to the recipients and also obtain their presence information.

#### 2) Challenge-Response Filtering

In our approach, we distinguish the unknown users from users in the Buddy List. The messages coming from the users

in the Buddy List are accepted. All the messages from unknown users will go through the challenge-response test first. When a message comes from an unknown user, a challenge is immediately sent back to the sender with a message stating that the message will be delivered to the recipient only if the sender successfully meets a challenge. If the sender could not complete the test within a specified time limit, all the messages received during this period will be discarded. Challenge can be an image verification test that automated systems cannot pass. In order to save the bandwidth and prevent DoS attacks, instead of sending the exact image or audio challenge, a link containing the challenge can be sent. Similar technique is often used to limit the automatic e-mail/IM user registration. However, IM systems can provide more effective defense against such attacks. For instance, server side rate control mechanism can be applied to avoid wasting processing time and space on automated initializing messages in high volume, which is also a DoS attack.

The challenge-response testing limits the messages from the users not in the recipient's Buddy List. The recipient can set up a small time interval  $m$ , and only the messages received during this interval from unknown user are allowed to be delivered, provided that the sender passes the challenge-response testing. For those users not in the recipient's Buddy List, passing the challenge-response testing is similar to getting a temporary pass. By customizing the appropriate interval  $m$ , the recipient can leverage receiving messages from the unknown users

### 3) Content based Filtering

We exploit content-based filters, including signature based, Bayesian, and full text comparison approaches.

#### a. Fingerprint Vector Based Filtering

We apply the fingerprint vector based approximate text comparison algorithm [6, 7], which is one of the signature based filters, and all the messages are checked as follows. For any given text string, a set of fingerprints is generated. To calculate a block text fingerprint vector, the checksums of all possible consecutive substrings of length  $L$  will be calculated. There are  $(n - L + 1)$  such substrings for a given string of  $n$  characters. Calculating checksums of all such substrings is a fast operation with complexity  $O(n)$ . By sorting the set of all checksums and selecting a size  $N$  subset with the lowest values, we can use this fingerprint vector to represent the whole text. The number of common checksums between two vectors represents the similarity of these two texts. Therefore, by searching a message's fingerprint vector among the known spim fingerprint vector database and comparing the maximum number of common checksums with a chosen threshold  $t$ , we can easily tell whether it is a suspicious spim message or not.

This algorithm works well for large message/file comparison. Note that IM message is relative short and if the size of the spim message is less than  $L$ , this algorithm fails to filter it out. However, user can customize the parameter  $L$  to tune the granularity for a similarity match. For e-mail, a size  $L$  of 50 might work well [6]. In our experiments, we found when  $L=10$  and  $t=2$ , it works well for IM message filtering due to the fact that IM message is relative short. Also note that the goal of using content-based filter is to filter out the unsolicited

commercial advertising messages and the length of most of them are greater than 10. More details are in section IV.

#### b. Word Based Longest Common Subsequence Filtering

We investigate the full text comparison approach due to the small size of IM messages. Our approach takes advantage of the longest common subsequence (LCS) algorithm. Instead of matching the text letter by letter, we characterize a longest common subsequence of words to decide the similarity of two messages. LCS-based spim detection has high fault-tolerance because it is based on full context comparison.

We use dynamic programming to compute a longest common subsequence of words of two messages, which is similar to the LCS algorithm. The difference is that we do not search for the common characters shared by the two messages but the common words which are in the meaningful unit. Our experiments show that a threshold of 7 guarantees accuracy in terms of *FAR* (False Acceptance Rate) and *FRR* (False Rejection Rate).

#### c. Bayesian Filtering

We also examine Bogofilter [11], which is a fast Bayesian spam filter. It exploits Gary Robinson's geometric-mean algorithm with the Fisher's method modification to distinguish spam from ham. Bogofilter considers the input as a stream of tokens. Each token is checked against a word list database, which maintains the number of occurrences of each token in ham and spam corpus. These numbers are used as an estimate of the probability that a message containing the token is spam.

Bayesian filtering is one of the most efficient spam techniques. However, from the experiments, we are surprised to find that spim Bayesian filter is not as effective as we have expected, probably due to the small size of IM messages.

#### 4) Collaborative Feedback Based Filtering

Similar to e-mail spam, the collaborative user feedback based filtering is an effective mechanism to collect spim information, such as Black List or known spim database. This information can be distributed throughout the network and help other IM users filter out the spim messages.

Based on the content based checking, the suspicious spim messages is placed in a folder where the recipient can finally decide whether it is a spim message or not and take appropriate countermeasures - either discard it or report it to the server as a false acceptance. On the other hand, the client side periodically retrieves the latest spim information from the server and updates its local spim database. Note that either clients can periodically retrieve the spim information from the server or the server can push spim information to clients.

IM server/gateway takes the responsibility of collecting and propagating spim information. Whenever the normal IM clients receive the spim message they report to the server. IM server then collects and integrates the spim information, updates the spim database accordingly, and propagates the latest spim information to IM clients. On the other hand, the servers exchange the spim information among their service domains.

Spim information propagated can be a Black List or a spim signature. Each IM user is only allowed to submit a same report once. If the number of the users who submit a same report is

greater than a threshold, either the reported sender is classified as a spimmer or the reported message is classified as a spim itself. Similarly, the users can also vote to revoke the reports. The server traces all the reports and revocations and dynamically updates and propagates the spim information.

It is worth noting that [13] points out that 86% of the e-mail users simply delete e-mail spam instead of reporting spam to the server. Without the cooperation of users, the collaborative filtering loses its power. Since it is hard to force users to change habit, to make the whole system more efficient, we provide a client side option allowing automatic spim reporting, which is based on the interaction between the content-based filters and the local Black List.

#### IV. EXPERIMENTS

The criteria of a sound spim filter are: filtering out most of the spim yet without eliminating too many legitimate IM messages. That is, we intend to achieve a low false acceptance rate ( $FAR = n_{S \rightarrow H} / n_S$ ), which is the ratio of the number of incorrectly classified spim as ham to the total number of spim, and a low false rejection rate ( $FRR = n_{H \rightarrow S} / n_H$ ), which is the ratio of the number of incorrectly rejected ham messages as spim to the total number of legitimate messages. Due to IM's real time nature, low computation overhead should also be considered as another performance metric of spim filters.

In the experiment, large corpus of spim and good messages are used to evaluate our spim filtering techniques. However, there are few data in the public domain due to company privacy. Instead, we download a spam corpus [8] and choose all the short messages which are less than 1kb as spim; we believe the unsolicited IM message is similar to e-mail spam except for the size. Also we gain the corpus of short message service (SMS) from [9] and take it as our ham database since the IM message is similar to SMS. All the experiments are run on an Intel P4 machine with 1G RAM.

##### A. Fingerprint Vector Based Filtering

The fingerprint vector based approximate text comparison algorithm is originally used to look for the similarity between two large files. It works well in spam filtering when  $L=50$ . To adapt to spim filtering, we need to adjust the parameters.

Usually small number of users  $N$  is good enough and works well for e-mail spam. Since IM message is relative short, we can use large  $N$  to achieve better accuracy as well as to keep a relatively small spim database. In our experiment, we set  $N=50$  and investigate the impact of  $L$  and  $t$ .

The results are shown in Fig. 4 and Fig.5. This algorithm achieves good  $FAR$  and  $FRR$ . We can see the accuracy is a function of  $L$  and  $t$ . Note that one drawback of this algorithm is that it cannot filter out spim message of length less then  $L$ , thus choosing an appropriate  $L$  is very important. Since the message size for IM is short, when we decrease  $L$ , it achieves better  $FAR$ . On the other hand, when threshold  $t=2$ , it reduces the  $FAR$  significantly, although it might increase the  $FRR$  along with small  $L$ . From the figure, we can see the combination of  $L=10$  and  $t=2$  provides satisfactory results. Furthermore, the running speed of this algorithm is quite fast. On the average it takes less than 0.03s to search one message within a database with 2000 spim messages.

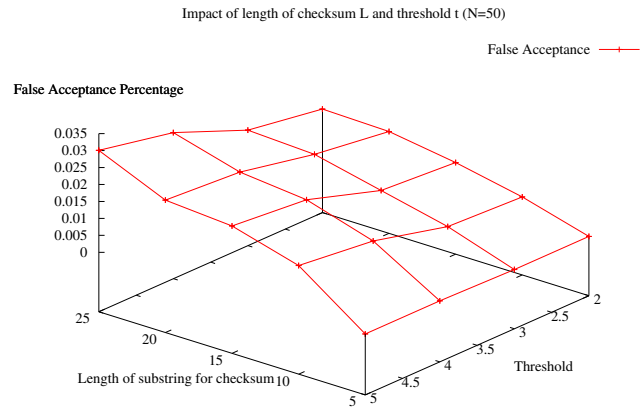


Figure 4: The impact of L and t (N=50) on False Acceptance Rate

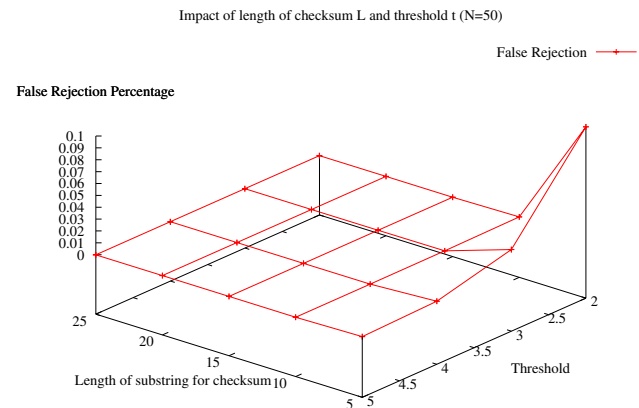


Figure 5: The impact of L and t (N=50) on False Rejection Rate

##### B. Word Based Longest Common Subsequence Algorithm

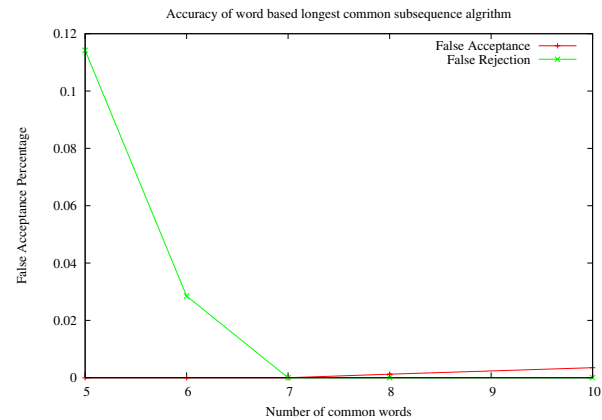


Figure 6: Accuracy of Word Based LCS Filter for Spim Filtering

To test this algorithm, we use the number of common words shared by two messages as our threshold. As shown in Fig. 6, the best performance is when choosing 7 as a threshold. When the number of common words shared by the message and a message in the spim database is more than 7, we consider it as spim. The  $FAR$  is changing gradually with similarity while there is a sharp increase in  $FRR$  when we decrease the threshold from 6 to 5. Usually there are certain common words in most of the messages, if the threshold is set too low, comparison does not provide useful information.

The time complexity of this algorithm is comparable to the fingerprint vector based filtering algorithm. On the average, the time for searching one message within a database of 2000 spim messages is less than 0.047s.

### C. Bayesian Filtering

Bogofilter is originally designed for e-mail spam, thus it takes into account e-mail header. To evaluate it with instant messages we attach to each message with an empty e-mail header and replace each e-mail spam's header with empty header so that it only considers the content of the message body.

As usual, training is crucial to achieve good performance. In the first experiment, we train Bogofilter with half of spam corpus which contains around 4000 small e-mail spams and half of ham database, which consists of 5000 good short messages. Surprisingly, when we test Bogofilter with the other half of our corpus, we find that *FAR* reaches up to more than 50%. This is because Bogofilter is more sensitive to hammish word than spammish word. The spam corpus used for training is not big enough to have probabilistic significance for Bogofilter to deal with sales pitch in the spam messages.

We retrain Bogofilter with 10000 spam messages and 5000 ham messages. The results are shown in Fig.7.

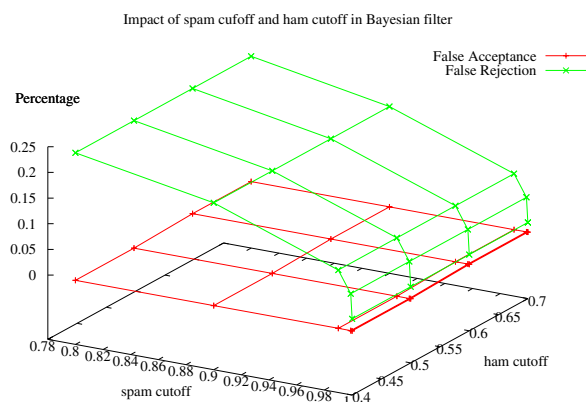


Figure 7: Accuracy of Bayesian Filter for Spim Filtering

From the figure, we can see that Bogofilter behaves well with respect to *FAR*. But even when we set *spam\_cutoff* to as high as 0.999999, we still get 2.3% *FRR*, which implies that two out of every hundred valid messages will be mistakenly classified as IM spam. It is too high to most users. Note that the high *FRR* is mainly due to the small size of IM messages.

Shorter message conveys less information that can be used to distinguish ham from spam. For Bogofilter, it chooses a certain amount of words of interests with *min\_dev* smaller than 0.2 to calculate the probability that a message is spam. If the message is too short, the maximum number of words of interest turns out to be the number of words contained in the message. Therefore, it greatly constrains the power of Bogofilter, and, consequently, it is inadequate for spim filtering.

## V. CONCLUSION

With the fast advancement of IM applications, the threats of spim have drawn serious public attention. Because of the

similarity between spim and spam, we could have resorted to the spam defending technology. However, due to the IM unique characteristics, such as small size and real-time nature, it turns out that most of the spam filtering techniques cannot be applied directly.

We have proposed a global and personalized spim detection and filtering architecture that integrates the enhanced spam detection techniques and also ushers our new methods, which are particularly effective for spim filtering, including the sending rate control for efficient and large scale spim filtering. With this architecture, the spim can be effectively detected and eliminated at the client side by personalized filters and also at the server/gateway side by a global spim defending mechanism.

Through experiments, we have investigated the performance of some content based filtering techniques and concluded that the fingerprint vector based algorithm and the word based LCS algorithm work well with the carefully chosen parameters. Although the Bayesian filtering is successful in defending spam, it is not as appropriate as we have expected in spim detection and filtering due to the small size of IM messages.

The collaborative filtering is another effective technique for spim filtering. In our design, the server mainly takes the responsibility of collecting and propagating spim information. Given that more and more P2P techniques are applied in spam detection and filtering [6, 10], we can introduce the IM volunteers or some existing P2P nodes to facilitate the spim information propagation. However, this highly distributed approach might raise legitimate security concerns.

## ACKNOWLEDGEMENTS

The insightful reviews and constructive comments by the anonymous reviewers of this workshop are deeply appreciated.

## REFERENCES

- [1] Mourad Debbabi and Mahfuzur RaHAMn, "The War of Presence and Instant Messaging: Right Protocols and APIs", Proceedings of IEEE Consumer Communications & Networking Conference, January 2004.
- [2] Thomas Claburn, *Spim, Like Spam, Is On The Rise*, Information Week, March 30, 2004
- [3] P. Saint-Andre, Ed., *RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core*, Oct 2004
- [4] P. Saint-Andre, Ed., *RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*, Oct 2004
- [5] B. Campbell, Ed., J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle, *RFC3428: Session Initiation Protocol Extension for Instant Messaging*, Dec, 2002
- [6] F. Zhou, L. Zhuang, B. Zhao, L. Huang, A. Joseph, and J. Kubiawicz, *Approximate object-location and Spam filtering on peer-to-peer systems*. In ACM/IFIP/USENIX International Middleware Conference, June 2003.
- [7] Manber, U., *Finding Similar Files in a Large File System*. In Proceedings of Winter USENIX Conference, 1994.
- [8] Spam corpus, <http://www.cs.nmt.edu/~janbob/SPAM>
- [9] SMS corpus, <http://www.comp.nus.edu.sg/~rpnlpit/downloads/corpora/smsCorpus/>
- [10] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati. *P2P-based Collaborative Spam Detection and Filtering*. In Proc. 4th IEEE Conf. on P2P, August 2004.
- [11] Bogofilter, <http://bogofilter.sourceforge.net/>
- [12] M. SaHAMi, S. Dumais, D. Heckerman and E. Horvitz, *A Bayesian Approach to Filtering Junk E-Mail*, in Proc. AAAI-1998 Workshop on Learning for Text Categorization, 1998
- [13] D. Fallows, *How It Is Hurting Email and Degrading Life on the Internet*, 2003.

# Analysis of IPSec Overheads for VPN Servers

Craig Shue, Youngsang Shin, Minaxi Gupta, Jong Youl Choi  
Computer Science Department, Indiana University, Bloomington, IN, U.S.A.  
{cshue, shiny, minaxi, jychoi}@cs.indiana.edu

## Abstract

*Internet Protocol Security (IPSec) is a widely deployed mechanism for implementing Virtual Private Networks (VPNs). This paper evaluates the performance overheads associated with IPSec. We use Openswan, an open source implementation of IPSec, and measure the running times of individual security operations and also the speedup gained by replacing various IPSec components with no-ops. The main findings of this study include: VPN connection establishment and maintenance overheads for short sessions could be significantly higher than those incurred while transferring data, and cryptographic operations contribute 32 – 60% of the total IPSec overheads.*

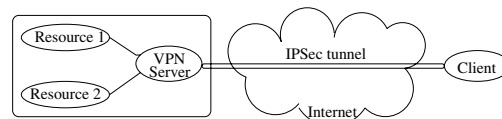
## 1. Introduction

IP packets do not have any inherent security, making it easy to forge and inspect their content, including the IP addresses contained in them. As a result, there is no guarantee that a received IP packet: 1) is from the claimed sender, 2) contains the original data that the sender put in it, or 3) was not sniffed during transit. *IPSec* [10], [5] (short for *IP Security*) provides a method to protect IP datagrams by defining a method for specifying the traffic to protect, how that traffic is to be protected, and to whom the traffic is sent. For both IPv4 and IPv6, it offers the choice of two protocols: Encapsulating Security Payload (ESP) [9] or Authentication Header (AH) [8]. AH provides data integrity, anti-replay protection, and proof-of-data origin on received packets. ESP provides data confidentiality in addition to all that AH provides. In order to establish and periodically refresh the necessary cryptographic parameters for both these protocols, IPSec defines another protocol called the Internet Key Exchange (IKE) [7] protocol.

Both ESP and AH protocols can be used either in *tunnel* mode or in *transport* mode. The transport mode leaves the original IP header untouched and is used to protect only the upper-layer protocols. As a result, it can only be used between two end hosts that are also cryptographic end points.

The tunnel mode protects the entire IP datagram by use of encapsulation and can be used to protect traffic between two end hosts, or two gateways (e.g. routers, firewalls), or between an end host and a gateway.

A popular and widely deployed use of IPSec is in establishing Virtual Private Networks (VPNs). Through the use of cryptographic primitives, VPNs allow off-site personnel to access organizational resources over the public Internet as if they were on-site. Figure 1 shows an example of a popular VPN configuration. In order to access a resource, say Resource 1, the client establishes an IPSec *tunnel* with the organization's VPN server. Typically, the *ESP* protocol is used to ensure the confidentiality of data traversing the Internet. To communicate with Resource 1, the client forms a IP datagram with the organization's local IP address (provided by the VPN server) as the source address and the IP address of Resource 1 as the destination address. It then encapsulates this datagram in another IP datagram with its present IP address as the source and organization's VPN server's IP address as the destination. Upon receipt of this IP-in-IP datagram, the VPN server strips off the outer IP header, decrypts the inner IP datagram, and sends it to Resource 1. To Resource 1, the datagram appears as though the client was on-site. Tunnel mode is preferred to the transport mode because the client can later utilize the same tunnel to access Resource 2 remotely as well.



**Figure 1. A popular VPN configuration.**

While performance of Transport Layer Security (TLS) has been characterized [2, 1], very little research has been conducted thus far in understanding the overheads involved in using IPSec. IPSec was the focus of work in [6], where the authors examined the ESP and AH encapsulation overheads. However, a comprehensive picture of IPSec overheads was not presented because the performance penalty associated with the IKE process was not examined. This aspect needs to be understood because the IKE protocol is

usually not just run once during the establishment of a VPN connection. It is periodically run during longer VPN sessions for stronger security. Also, frequent disconnections may cause IKE to be run multiple times for clients in wireless environments. Further, the difference in overheads due to different encryption algorithms and key sizes was not investigated in the previous IPsec work [6].

In this work-in-progress paper, we present the preliminary analysis of the performance overheads associated with IPsec using Openswan [3], an open source implementation of IPsec. We focus on the tunnel mode of operation and the ESP protocol because it is the most widely deployed configuration for creating VPNs. We utilize two methods in order to analyze the performance impact of the ESP protocol, the IKE protocol, various encryption algorithms, and various cryptographic key sizes: measuring run-times for individual security operations, and replacing various IPsec components with no-ops and recording the speed-up in the run-time of various IPsec phases. The main findings from this work include: 1) overheads of the IKE protocol are considerably higher than those incurred by ESP for processing a data packet, 2) cryptographic operations contribute 32 – 60% of the overheads for IKE and 34 – 55% for ESP, 3) digital signature generation and Diffie-Hellman computations are the largest contributor of overheads during the IKE process and only a small amount of the overheads can be attributed to the symmetric key encryption and hashing, and 4) symmetric key encryption is the most expensive operation during the ESP process.

## 2. Background

IPsec integrates security at the IP layer. In order to provide higher layer services that are IPsec oblivious, it defines two new protocols, *Encapsulating Security Payload (ESP)* and *Authentication Header (AH)*. Both ESP and AH protocols encapsulate IP packets using ESP and AH headers respectively. In the case of *transport mode* of operation, the original IP header is retained and the new ESP or AH header is inserted between the IP header and the header of the higher layer transport protocol like TCP. As a result, using transport mode implies using IPsec between the actual source and destination of the IP packets. In the case of *tunnel mode* of operation, the entire IP packet to be protected is encapsulated in another IP datagram and an IPsec header is inserted between the outer and inner IP headers. The communication end points in tunnel mode are those specified in the inner header and the cryptographic end points are those appearing in the outer IP header. Due to this flexibility offered by the tunnel mode, it is the most widely used in creating VPNs. Hence, we focus on the tunnel mode in this paper.

The choice between ESP and AH protocols depends on

the desired level of protection for the IP datagrams. As mentioned earlier, the AH protocol offers data integrity, anti-replay protection, and data source authentication but does not offer data privacy. The ESP protocol offers data privacy in addition to all the features offered by the AH protocol and is the protocol of choice for VPN deployment. Consequently, the subsequent description in this section focuses on the ESP protocol used for forming VPNs, even though much of it applies to AH as well. Also, the performance evaluation presented in this paper focuses on the ESP protocol.

The selection of a cryptographic mechanism is required before any IP data can be encrypted using the ESP protocol. The available primitives include using a symmetric key between the two cryptographic end points or the public keys of the end points. Since using public key encryption is computationally expensive, IPsec uses symmetric keys. But, before IPsec can use symmetric key to encrypt data, the symmetric keys must be exchanged. While out-of-band means can be used, it would not be possible to scale them for large networks. Moreover, they are not conducive to changing keys in the event the key is compromised. To accomplish this goal, IPsec defines the *Internet Key Exchange (IKE) protocol*. We now describe the IKE and ESP protocols in sections 2.1 and 2.2 respectively.

### 2.1. IKE Protocol

The goal of the IKE protocol is to establish and maintain shared security parameters and authenticated keys between the two IPsec end points. It uses a series of messages contained in UDP datagrams, typically directed to port 500. The IKE protocol consists of two distinct phases. The first phase establishes a symmetric IKE key between the *initiator* (VPN client in our case) and the *responder* (VPN server in our case). This key is used in the second phase to establish a symmetric IPsec key for use during ESP or AH encapsulation.

IKE defines two possible modes for the first phase of the protocol: the *main mode* and the *aggressive mode*. The main mode consists of 3 *exchanges* (6 messages in total) between the initiator and the responder. The aggressive mode requires half the number of messages but provides less negotiating capabilities; furthermore, it does not provide identity protection for the end points of the IPsec tunnel. The second phase is also called the *quick mode* and involves 3 messages.

IPsec uses the notion of *Security Association (SA)* in two different contexts. In the context of the IKE protocol, the SA defines the manner in which two end points communicate; for example, this involves agreeing on the algorithm used to encrypt traffic, the hash algorithm, and the mechanism to authenticate the other end point. IKE defines 3

categories of authentication methods (5 in total) for phase one: the first method involves the use of pre-shared keys, the next two methods use digital signatures (using RSA or digital signatures algorithms), and the last two methods use public key encryption.

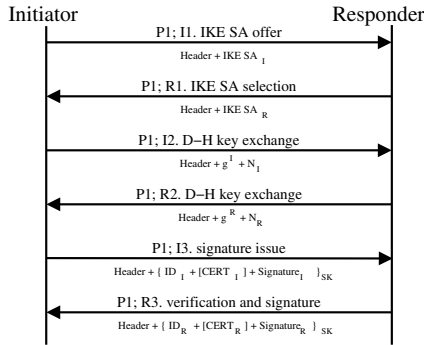


Figure 2. IKE Phase 1, Main Mode

Figure 2 shows the message exchanges for the first phase of the IKE protocol in main mode with digital signature scheme for authentication. In the first exchanges of I1 and R1, the initiator I suggests multiple SAs for the IKE protocol ( $IKESAI$ ) along with its cookie  $C_I$  and the responder R chooses one ( $IKESAR$ ) of them with its cookie  $C_R$ . In the second exchanges (I2 and R2), the initiator and the responder exchange Diffie-Hellman (D-H) public values,  $g^I$  and  $g^R$  respectively, and nonces,  $N_I$  and  $N_R$  respectively, to prevent replay attacks. In the last I3 and R3, each party computes independently the shared key  $SK$  from the previous exchanges – cookies, nonces, and D-H values – sends encryption of each identity ( $ID_I$  and  $ID_R$  respectively), a certificate for the public key verification (which is optional), and a signature on the hashed value of cookies, D-H values,  $SK$ , SAs, and identities. As a result, two parties will agree on the IKE symmetric key ( $SK$ ).

When using a pre-shared secret as the authentication mechanism for phase one, the first two exchanges are identical to the digital signature approach described in figure 2. The only difference is in the third exchange, when only the identity of the end-point and a hash are sent encrypted, instead of the signature.

When aggressive mode is used instead of the main mode for phase one, the first two messages on the initiator’s side are combined into a single message. Also, the responder’s first, second, and third messages are combined into a single message. This cuts down the total number of messages required to establish the IKE key using phase one in half.

The second IKE phase is done in *quick mode* and accomplishes two tasks: 1) it establishes an IPsec Security Association (SA)<sup>1</sup> and 2) it produces an IPsec key for use

<sup>1</sup>The IPsec SA is different from the IKE SA. In the context of IKE, the SA meant agreeing on the authentication mechanism, encryption and

during the ESP or AH encapsulation. Quick mode uses the IKE key derived from the first phase exchanges to encrypt its messages.

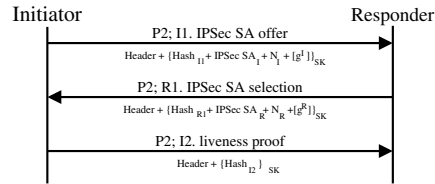


Figure 3. IKE Phase 2, Quick Mode

As shown in figure 3, the second phase of the IKE protocol is done by three message exchanges. For the first and second messages I1 and R1, the initiator I and the responder R exchange IPsec SAs ( $IPsecSA_I$  and  $IPsecSA_R$ ), nonces  $N_I$  and  $N_R$  (for replay attack protection), optional D-H values  $g^I$  and  $g^R$ , and hashes of these values and message IDs (to prove liveness),  $Hash_{I1}$  and  $Hash_{R1}$ . After the initiator sends one more hash ( $Hash_{I2}$ ) using both  $N_I$  and  $N_R$  and the message ID as the third message, both parties will agree on the IPsec symmetric key for use during ESP or AH encapsulation.

For better security during longer VPN sessions, IPsec provides a mechanism to periodically refresh both IKE and IPsec keys. Refreshing the IKE key entails running both IKE phases but refreshing the IPsec key only requires running the second phase (quick mode) again.

## 2.2. ESP Protocol

We now describe the processing of IP packets when ESP protocol is used in tunnel mode in IPsec VPNs. For processing outbound packets, the transport layer forwards data to the IP layer, which has been enhanced with the IPsec functionality. The IP layer consults a locally maintained Security Policy Database (SPD) that defines the security services afforded to this packet. The output of the SPD dictates whether the IP layer drops the packet, bypasses security, or applies security.

If security is to be applied, the appropriate IPsec SA is consulted by looking up the SA database (SADB)<sup>2</sup> and the entire IP packet is encrypted and placed inside another IP packet. To facilitate the processing of this packet at the other end, an ESP header containing SA mapping information and sequence number (to prevent replay attacks) is inserted between the new IP header and the original encrypted IP packet. An ESP trailer containing Integrity Check Value

hash algorithms. But, in the context of IPsec the SA defines the processing done on a specific IPsec packet by choosing between tunnel and transport modes, and between ESP and AH protocols.

<sup>2</sup>An IPsec SA negotiated through IKE second phase should exist at this point, if not IKE is invoked to establish it.

(ICV) is also inserted at the end of the new IP packet before sending it out.

Upon arrival of an inbound ESP packet, the SADB is consulted. If no SA exists, the received packet is discarded. Otherwise, it is de-capsulated and the appropriate IPsec key to decrypt the original IP packet is retrieved using the information contained in the ESP header of the received packet. Also, the sequence number contained in the ESP header is used to prevent replay attacks and the ICV value contained in the ESP trailer is verified to guarantee that the packet was not modified during transmission.

### 3. Methodology

We used Openswan [3], an open source implementation of IPsec, to examine various aspects of the IPsec protocol. This section describes the methodology used to understand the IPsec overheads, including the software and hardware used and the tests performed.

#### 3.1. Experimental Environment

To conduct our experiments, we used two x86 Dell Optiplex GX Pentium IV machines. The first, used as the VPN server, had a 1.66GHz processor and a 100Mbps network interface card, while the second, which was used as a VPN client had a 1.8GHz processor and a 1000Mbps network card. Both the machines had 512MBytes of RAM and were connected to each other through a 100Mbps Ethernet switch.

The machines ran Debian Linux [4] with a 2.6.8 kernel. We disabled the native IPsec support due to compatibility issues and instead used Kernel Level Internet Protocol Security (KLIPS), the shim provided by Openswan for IPsec support. The Openswan version used was 2.3.1*dr*3, the latest version at the time our experiments began.

For measuring both IKE and ESP overheads we used two different approaches: 1) measuring the time taken for individual security operations (referred to as *timing measurements* subsequently) and 2) replacing various IPsec components with no-ops (referred to as *no-op measurements* subsequently). In order to characterize the amount of time spent on various cryptographic operations, we inserted assembly codes<sup>3</sup> to capture the CPU cycle count at which a cryptographic function started and the count at which it completed. We determined the elapsed time for each operation by gathering the elapsed cycles and multiplying this cycle count with the processor's clock speed. This technique yields nanosecond resolution, allowing for highly accurate results and minimum overhead. To get an alternate view

<sup>3</sup>On Intel Pentium processors, RDTSC (Read Time Stamp Counter) instruction returns the number of clock cycles since the CPU was powered up or reset. This value is stored as a 64-bit number.

on the overheads we replaced the computationally expensive cryptographic operations with no-ops. For both IKE and ESP, we experimented with AES and 3DES symmetric encryption schemes with key sizes of 128 and 256 bits for AES and 192 bits for 3DES. For the hashing algorithm, we tested the MD5 hashing algorithm.

#### 3.2. Tests Performed

**IKE Testing.** Out of the three categories of authentication methods prescribed for IKE phase one, Openswan supports only two: digital signatures and pre-shared secret. We examined the overheads of both of these.

The pre-shared secret authentication mechanism is the most commonly used method in real-world VPN deployments using IPsec because it does not require that the clients possess public keys. It does require additional authentication measures because organizations often keep the pre-shared secret used to establish the IKE key in public domain for easy access for their personnel. Since the overheads for the digital signature authentication mechanism are a superset of those incurred when pre-shared secret approach is used, we focus on the overheads for the digital signature approach and elaborate where the two methods diverge.

We evaluated the overheads for both *main mode* and *aggressive mode* for IKE phase one and for *quick mode* for phase two. The public and private keys required for the digital signatures method were manually configured for both the client and server. To communicate with the running IKE daemon, Openswan uses a wrapper that connects to the daemon. We timed this wrapper to measure the time to complete the IKE exchanges. We conducted 25 trials of IKE daemon start-up, connection initiation, connection termination, and tear-down.

**ESP Testing.** The IPsec ESP module uses the symmetric key obtained through IKE to encapsulate and de-encapsulate outgoing and incoming IP packets respectively. For testing its overheads, we conducted 25 trials of the time it took the IPsec VPN server to process a ping request packet from the client connected via an ESP tunnel.

### 4. Experimental Results

This section reports on the experimental results obtained for timing and no-op measurements for ESP and IKE protocols for AES (key sizes 128 and 256) and 3DES (key size 192) encryption schemes with MD5 hashing algorithm.

#### 4.1. Timing Measurements

**IKE Timings.** Table 1 shows the cryptographic timing measurements for phases one and two of the IKE proto-

col for the responder (VPN server in our case) when main mode with digital signatures as the authentication method was used for phase one. The reported numbers are averaged over 25 trial runs. Here, and subsequently, the encryption algorithms and key sizes are denoted by a concatenation of the name of the algorithm and the key size (3DES192, AES128, AES256 respectively). The cryptographic operations are labeled in the same manner as in figures 2 and 3, with *P1* and *P2* prepended to disambiguate between the two IKE phases.

**Table 1. IKE cryptographic overheads for VPN server (in ms).**

MSG	OPERATION	3DES192	AES128	AES256
P1, R2	D-H comp.	17.59	17.64	17.55
P1	IKE key gen.	0.18	0.13	0.13
P1, R2	Decryption	0.17	0.06	0.06
P1, R2	Signature verif.	1.07	1.09	1.06
P1, R3	Hash verif. + gen.	0.04	0.04	0.04
P1, R3	Signature gen.	78.82	79.10	78.96
P1, R3	Encryption	0.06	0.16	0.16
P2	Decryption	0.11	0.03	0.03
P2	Hash verif.	0.02	0.02	0.02
P2, R1	D-H comp.	17.91	17.95	17.95
P2, R1	Hash gen.	0.03	0.03	0.03
P2, R1	Encryption	0.08	0.18	0.18
P2	Decryption	0.07	0.02	0.02
P2	Hash verif.	0.01	0.02	0.01
P2	IPSec key gen.	0.09	0.10	0.10
	Total:	117.59	117.93	117.66

As table 1 shows, the biggest contributor to the cryptographic overheads at the VPN server was the RSA signature generation. Out of the total 373.82ms recorded at the server for the IKE process for 3DES192<sup>4</sup>, including 117.59ms for the cryptographic operations as shown at the end of table 1, this one operation took approximately 21% of the total time required for the IKE process for all key sizes and encryption algorithms tested. However, verification of signatures sent by the client was much more efficient (1.07ms, 1.09ms, and 1.06ms for 3DES192, AES128, and AES256 respectively).

The D-H computations were the second biggest overhead at the server, consuming a total of 35.50ms (17.59ms and 17.91ms during phases one and two respectively) for 3DES192. These overheads varied little across encryption algorithms. The overheads associated with symmetric key encryption and decryption operations in both phases are quite low and vary with the size of the data being encrypted. Due to the small difference in the key sizes available for testing, no clear conclusion about the relationship of overheads and key sizes or encryption algorithms can be drawn. Finally, hashing contributed the least to the cryptographic overheads.

The details of the overheads recorded at the initiator

<sup>4</sup>This includes the IKE message processing at the client (including the cryptographic operations) because IKE messages happen in lock-step.

(VPN client) were similar to those presented here and are omitted due to space constraints. Overall, the client overheads were less since the machine used as the initiator had a 200MHz processing edge. In particular, the cryptographic overheads at the client were 107.27ms, 107.53ms, and 107.41ms for 3DES192, AES128, and AES256 respectively.

Changing the authentication method in IKE phase one main mode to pre-shared secret changed the total IKE overheads at the server to 214.99ms, 213.71ms, and 214.30ms for 3DES192, AES128, and AES256 respectively. These overheads differ from the digital signature overheads only by twice the digital signature generation overheads, one at the client and one at the server. This is expected because the remaining operations are very similar in both authentication methods. For conciseness, we have omitted the details of these results as well.

Next, we replaced the main mode by aggressive mode for IKE phase one and measured the total IKE overheads for both phases for digital signature authentication mechanism. This reduced the number of messages exchanged between the client and the server but did not change the cryptographic operations performed. The results reflected this and no difference in cryptographic overheads was observed. The overall IKE overheads decreased by only about 5ms.

Overall, the cryptographic operations contributed about 60% to the total running time of the IKE protocol for all encryption schemes and key sizes when digital signature authentication was used in main mode (32% when pre-shared key was used in main mode and 60% when digital signatures were used in aggressive mode). The remaining time was spent in the actual transmission of the messages, memory management, and other non-cryptographic operations.

**ESP Timings.** To infer the ESP overheads we measured the processing time of a ping packet at the server. Figure 4 shows the average overheads for 3DES192 over 25 trial runs when the client sends a 736 byte ping request to the server. The top to bottom of the y-axis in figure 4 shows the journey of an inbound ping request until the server prepares an outbound ping response packet. The inbound and outbound ESP overheads are demarcated by the dashed vertical lines and the rest of the overheads are the standard ping processing overheads incurred by the Linux TCP/IP implementation.

The key observation from figure 4 is that ESP processing for a packet is considerably faster than IKE. Also, out of the total 148.61 $\mu$ s inbound processing time, around 58% was contributed by symmetric key decryption. Similarly, 55% of the outbound processing time was contributed by the encryption operation. While the overheads for the hash computations were negligible in comparison with other overheads for IKE, this is not the case for ESP.

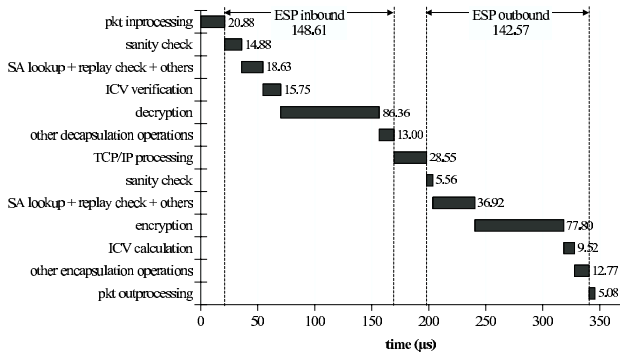


Figure 4. Server ESP overheads (3DES192).

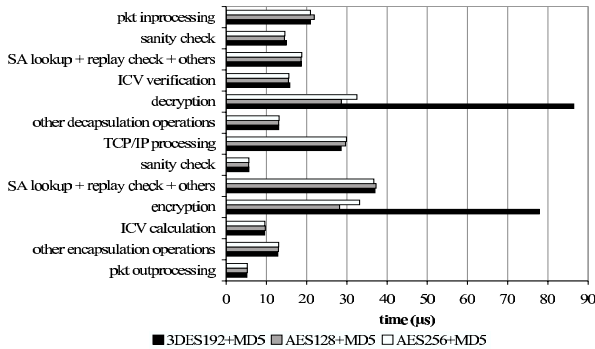


Figure 5. Comparison of ESP overheads.

Figure 5 shows the comparison between 3DES192, AES128, and AES256 for the same ping packet processing. While all other operations incur the same overhead as expected, AES256 is almost two and a half times faster than 3DES192 in spite of the bigger key size. It remains to be investigated whether this observation is hardware dependent and whether it holds true for larger transfers.

## 4.2. No-op Measurements

In order to infer the non-cryptographic IPsec overheads we no-oped cryptographic operations selectively and measured the speed-up. This section reports on the no-op experiments conducted.

**IKE No-ops.** Table 2 compares the server overheads between the full IKE implementation and its no-oped skeleton implementation where the cryptographic operations have been stripped off. Digital signatures were used for authentication during phase one, which was run in main mode.

Table 2. Comparison of full and skeleton IKE code (in ms).

	3DES192	AES128	AES256
Full IKE	373.82	370.86	371.06
Skeleton IKE	141.48	141.16	141.05

The difference between skeleton and full IKE implementations for all encryption schemes is very close to what we

observed using the timing measurements and the minor deviations can be attributed to compiler optimizations that occur when the code is no-oped. The results for pre-shared secret in main mode and digital signatures in aggressive mode have been omitted due to space constraints.

**ESP No-ops.** Table 3 compares the processing times at the server for native TCP/IP ping processing, the skeleton ESP version where all the cryptographic operations have been no-oped, and the full ESP implementation (3DES192, AES128, AES256). It is noteworthy that even when all cryptographic operations are no-oped, substantial ESP overhead remains, just like in the case of IKE. The cryptographic operations contribute 55%, 34%, and 37% to the overall ESP overheads for 3DES192, AES128, and AES256 respectively. 3DES192 performs the worst of three but it remains to be investigated whether this observation holds true for larger transfer sizes as well.

Table 3. Comparison of full and skeleton ESP code (in μs).

	3DES192	AES128	AES256
Full ESP	345.69	244.13	252.16
Skeleton ESP	182.37		
Native TCP/IP	46.44		

## Acknowledgments

We would like to thank Tom Zeller and Rob Henderson for insights on deployed VPNs and assistance in conducting the experiments respectively.

## References

- [1] G. Apostolopoulos, V. Peris, and D. Saha. Transport Layer Security: How much does it really cost? In *IEEE INFOCOM*, June 1999.
- [2] C. Coarfa, P. Druschel, and D. Wallach. Performance analysis of TLS Web servers. In *NDSS*, February 2002.
- [3] X. Corporation. Openswan Web-site, 2004. <http://www.openswan.org/>.
- [4] Debian Linux Web-site. <http://www.debian.org/>.
- [5] N. Doraswamy and D. Harkins. *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall, 1st edition, 1993.
- [6] G. C. Hadjichristophi, N. J. Davis IV, and S. F. Midkiff. IPSec overhead in wireline and wireless networks for web and email applications. In *22nd IEEE IPCCC*, April 2003.
- [7] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard), Nov. 1998.
- [8] S. Kent and R. Atkinson. IP authentication header. RFC 2402 (Proposed Standard), Nov. 1998.
- [9] S. Kent and R. Atkinson. IP encapsulating security payload. RFC 2406 (Proposed Standard), Nov. 1998.
- [10] S. Kent and R. Atkinson. Security architecture for the internet protocol. RFC 2401 (Proposed Standard), Nov. 1998. Updated by RFC 3168.

# Practical Security for Disconnected Nodes

Aaditeshwar Seth     Srinivasan Keshav  
*School of Computer Science*  
*University of Waterloo, Canada*  
*{a3seth, keshav}@uwaterloo.ca*

## Abstract

*Endpoints in a delay tolerant network (DTN) [5] must deal with long periods of disconnection, large end-to-end communication delays, and opportunistic communication over intermittent links. This makes traditional security mechanisms inefficient and sometimes unsuitable. We study three specific problems that arise naturally in this context: initiation of a secure channel by a disconnected user using an opportunistic connection, mutual authentication over an opportunistic link, and protection of disconnected users from attacks initiated by compromised identities. We propose a security architecture for DTN based on Hierarchical Identity Based Cryptography (HIBC) that provides efficient and practical solutions to these problems.*

## 1. Introduction

The emerging field of delay tolerant networks (DTN) [5] has recently attracted much attention. Such networks arise in inter-planetary Internets, sensor and ad-hoc networks, and other ‘challenged’ scenarios when connectivity is intermittent, such as in rural and underwater communication networks. In a DTN, client applications running on mobile or fixed devices opportunistically exploit connectivity over intermittent links. Mobile routers can also provide connectivity by acting as ‘data mules’ to carry data to and from servers that may reside in the Internet. Provision of security in such situations is a daunting task: traditional mechanisms are not well suited to environments where nodes may be disconnected for long periods of time and end-to-end communication is usually not possible. In this paper, we describe practical solutions to three problems faced by such disconnected nodes: (1) establishing a secure channel with another node (2) mutual authentication over an opportunistic link and (3) protection from users and infrastructure nodes whose credentials have been revoked or compromised. We solve these problems by extending well-known techniques in Hierarchical Identity-based Cryptography (HIBC) [3]. We start by describing a use case and explaining problems with the use of traditional security mechanisms for this use case. This is followed by an overview of the Delay Tolerant Networking architecture in Section 3, and the threat model in Section 4. Section 5, 6, and 7 describe in detail our proposals to mitigate these attacks.

## 2. Use case

Consider the following typical DTN scenario: suppose a bus with a WiFi-based router and local storage drives past a user with a Personal Digital Assistant (PDA) with wireless capabilities. Recent studies [11] have shown that during the

short time that the bus and the PDA are within range of each other, it is possible to opportunistically transfer tens of megabytes of data on the wireless link. However, to make this practical, the following three problems need to be solved. First, the PDA user must be able to establish a cryptographically-strong secure channel with some endpoint, for instance, with a mail server to download or upload mail. Second, the user and the bus must authenticate each other, so that the user is assured that the bus is not a rogue, and the bus knows how to bill for usage. Finally, if the permissions of either party to the exchange are revoked, these guarantees should continue to hold. Note that a solution for this use case is generally applicable broadly to any communication involving opportunistic links.

The state-of-the-art techniques to provide these assurances include a combination of Public Key Infrastructure (PKI) certificates issued by trusted third parties and Certificate Revocation Lists (CRLs) [13]. With PKI, a sender can establish a secure channel by encrypting data with a one-time session key, and encrypting the session key in turn with the recipient’s public key. Mutual authentication is assured by means of certificates issued to the bus and the user by a mutually trusted third party. Finally, CRLs allow any entity to become aware of other entities whose private keys have been compromised.

However, not all these mechanisms work well in a disconnected environment. A disconnected sender cannot efficiently use PKI because finding out the recipient’s public key requires an end-to-end round trip to a central or replicated lookup database, substantially delaying actual data transmission. Mutual authentication by means of certificates is certainly feasible using PKI certificates, but it requires authenticating parties to carry certificates from mutually trusted authorities. Finally, certificate revocation lists are unsuitable when updates can be excessively delayed and there are severe resource constraints on storage and link capacities. In our work, we propose solutions that eliminate these problems.

Our contribution is the development of practical cryptosystems for disconnected environments using HIBC [3] for creating secure channels, providing mutual authentication, and key revocation. Our solution is novel in that it explores the practical aspects related to deployment of DTN in remote rural or disconnected areas. This includes procedures for initial key establishment and roaming among different service providers. We also describe a simple technique to prevent a

user's identity from being compromised due to the loss or theft of a mobile device.

### 3. DTN overview

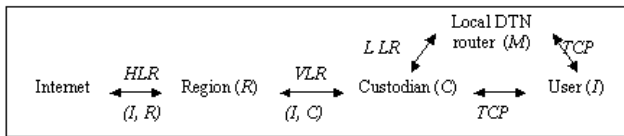
We use the Delay Tolerant Networking (DTN) [5] architecture as the basis of our design. This architecture has the following salient features:

1. Intermediate persistent storage
2. Use of opportunistic links
3. Data is sent in the form of self-identifying *bundles*

We now present some definitions relevant to our work.

1. *Region*: A region is a collection of mutually reachable DTN routers, determined by administrative policies, communication protocols, naming conventions, or connection types. The Internet is a single DTN region.
2. *Gateway*: This is a DTN router with interfaces on more than one region. An *Internet gateway* is a DTN router with at least one interface to the Internet region.
3. *Custodian*: This is a DTN router that acts as always-available proxy for intermittently connected hosts. Custodians opportunistically receive bundles from disconnected hosts, forward them to other custodians, and deliver them to a receiver whenever the receiver connects to the network.
4. *Local DTN router*: This is the DTN router that communicates directly with an endpoint. A local DTN router may or may not be a custodian as well.

We now outline some extensions to the basic DTN architecture to support user mobility; these are described in more detail in Reference [1]. We assume that mobile hosts are identified using an opaque globally unique identifier (GUID). We also assume that every DTN router has a 'default' entry that allows bundles to be forwarded (eventually) to an Internet gateway. The Internet region maintains a registry called the *Home Location Register (HLR)* that maps the mobile's GUID ( $I$ ) to its current region ( $R$ ). Each region maintains a *Visitor Location Register (VLR)* that stores a mapping and path from the GUIDs ( $I$ ) of all hosts currently in the region to that host's custodian DTN router ( $C$ ). Finally, each custodian maintains a *Local Location Register (LLR)* that maps from the GUID to the best last-hop fixed or mobile DTN router ( $M$ ) for each mobile.



**Fig. 1: Three stage hierarchy of lookups**

Routing tables in a region are established using reverse-path-forwarding when a REGISTER message sent by a host propagates towards its closest Internet gateway. To send data, a host sends an 'unbound' bundle that is propagated using default paths to an Internet-accessible gateway, which locates

the host's current region using the HLR and forwards the bundle to one of the region's gateways. Routing within a region from the gateway to the host uses the routing tables established using during host registration

This three-stage lookup hierarchy is shown in Fig. 1. When a mobile device moves, its location information is updated, if necessary, in the appropriate location registers using a REGISTER message.

### 4. Threat model

We assume the following threat model:

1. Rogue DTN routers may pretend to be valid DTN nodes.
2. DTN routers may be physically hijacked and compromised, but this is eventually detected.
3. End-systems can be hijacked or can turn malicious.
4. Eavesdroppers can potentially overhear wireless communication and break WEP-like mechanisms.

Given this threat model, consider a user who would like to conduct a secure transaction, such as a bank transaction, over a DTN. Because infrastructure nodes cannot be trusted, every opportunistic link must include a phase of mutual authentication. Second, users will want to establish end-to-end secure channels to prevent eavesdropping. Third, the infrastructure must protect itself from rogue routers. Finally, all nodes should protect themselves from nodes that were detected as being hijacked or declared malicious (the actual detection is outside the scope of this paper). This requires techniques to establish end-to-end secure channels, perform mutual authentication, and prevent communication with revoked nodes. We now describe these mechanisms.

### 5. Establishing secure channels

#### 5.1. Hierarchical Identity Based Cryptography

Boneh and Franklin [4] proposed the first practical Identity Based Cryptography (IBC) scheme and many variations have subsequently been described in the literature. Unlike traditional PKI, where a user obtains the public/private key pair from a certifying authority, public keys in IBC can be any string, but private keys are obtained from a trusted authority called the Private Key Generator (PKG). Hierarchical IBC extends IBC by establishing a cooperative hierarchy of PKGs. The top-level PKG is called the root PKG, and the other PKGs are called domain PKGs, each of which inherits the first part of its public ID from its parent. A detailed description of Hierarchical Identity Based Signature (HIDS) is given in [3, 10]. In the rest of the paper, we represent the public key of a user at level  $t$  in the key hierarchy as  $username@ID...ID_{t-1}$ . This indicates that the parent PKG of the user is the domain PKG at level  $t-1$  with a public identifier of  $(ID_1 ... ID_{t-1})$ .

Identity Based Cryptography (IBC) is ideally suited for creating a secure channel in a disconnected environment because the public key of an entity can simply be its public ID, and hence a lookup step is not required. For example, the

public ID for a user can be the email address of the user itself. Another advantage is that the possession of a valid private key implies that the certification authority has certified the identity. Therefore, a valid signature serves as an assurance of authentication. Finally, a user can freely self-generate certificates signed with its private key that are universally verifiable.

It is well known that HIBC suffers from lack of forward secrecy, meaning that the compromise of a key can compromise earlier transactions as well, which were conducted using that key. A forward-secure HIBC scheme is proposed in [8]. We propose a simpler algorithm for forward-secure HIBC in Section 5.4.

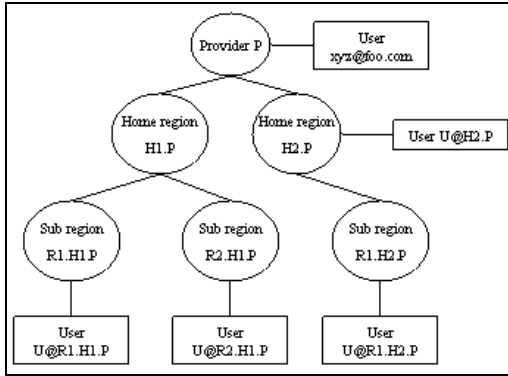


Fig. 2: Hierarchy of layout of regions

HIBC also suffers from the problem that if a PKG is compromised then it can yield all the private keys generated for lower level PKGs and users, which can be maliciously used to decrypt messages. A combination of IBC and PKI in [7] has been shown to avoid this problem, but the scheme cannot be used in a DTN [10]. In this paper, we assume that the PKG nodes are trusted and cannot be compromised. We are investigating alternative solutions to this problem in ongoing work.

## 5.2. Using HIBC in DTN

We incorporate HIBC into DTN as follows: As shown in Fig. 2, an arbitrary tree-like hierarchy is imposed on the DTN regions, based on administrative structures and policies. Each provider maintains its own top-level PKG, preferably in its partition of the DTN Internet region. Every sub region has its own domain-level PKG; alternatively, location registers in the sub-region should be able to default-route key pair requests to a parent PKG. A user can request a public ID and private key either from his nearest regional PKG (for instance, user  $U@R1.H1.P$  shown in Fig. 2, requests his public ID from the PKG in sub-region  $R1.H1.P$ ), or directly from the top-level PKG. The procedure of acquiring public-private key pairs is explained below in Section 5.3, and needs to be executed only once for new users who need a DTN identity. Each DTN router also maintains a unique identity for itself. The public ID for a DTN node in the region  $R1.H1.P$  can be written as  $DTN-IPaddress@R1.H1.P$ .

HIBC allows the creation of an end-to-end secure channel: the sender encrypts all data with the public key of the recipient, and the data can be decrypted only by the recipient. This provides confidentiality, integrity, and authenticated access. Besides allowing end-to-end secure channels, HIBC also protects the infrastructure from a class of attacks on the location management subsystem. Recall that a mobile host sends control messages whenever it changes its location. We use HIDS between the mobile host and the location registers being updated. This ensures safety from fabrication of control messages, redirection attacks, and the creation of dead-ends by unauthorized updating of location registers.

Finally, custodians in DTN send messages to the end-systems when custody has been transferred. We require custodian DTN nodes to sign these messages for the bundles that they take custody of ensuring safety from spoofing. End users can store these acknowledgements for auditing. Since the HIDS scheme itself ensures non-repudiation, the audit logs can be used as proof of custody transfer.

## 5.3. Establishment of system parameters

IBC requires each new user to obtain a public-private key pair from the top-level or a domain-level PKG. Users can form their public ID by concatenating a desired user-name with the region name ( $U@R1.H1.P$ ) of a domain PKG, or can request the root PKG to use any well-known ID like their email address as their public key. PKGs then push the new private keys to the users. Once the mobile host acquires the key pairs, it does not need to initiate any more interactions with the PKG, and only relies on the PKG to *push* time-based keys on a scheduled basis (see Section 7).

A new user that is directly connected to the PKG obtains its private/public key pair by communicating with the PKG over a standard secure channel mechanism like SSL. However, if the new user is in a disconnected region, it cannot communicate with the PKG. How then should it obtain its keys? We show this process in Fig. 3. We propose that USB storage devices (such as the popular ‘USB keys’) be used by the PKG to distribute keys through authorized distribution agents to disconnected end users. For instance, these pre-loaded USB keys could be given to a kiosk operator who authenticates a user first-hand and then hands over a USB key (similar to the way SIM cards are handed out for cell phones today). These storage devices carry a pair of ( $UID$ , *Symmetric key*) that has been generated by the PKG. During setup time users encrypt their desired public ID or username with the symmetric key found on the USB storage device and send it to the PKG, along with the ID of the symmetric key. The PKG looks up this ID to decrypt the message and determine the user’s public key. It then computes the public-private key pairs for the mobile host and sends them back to the user encrypted on the same symmetric key. Because the symmetric key is a one-time secret shared only by the new user and the PKG, this assures the security of this communication. To prevent the kiosk operator from tampering with the USB storage device, the device itself can be wrapped in a tamper-

resistant package (such as a sealed cellophane wrapper), which can be verified by visual inspection.

Note that because we require an authorized agent to distribute the USB keys, we assume that if a user desires to use a well-known ID like his email address as his public ID, then the authorized agent can verify that the email address being requested by the user is indeed the user's own email address.

Note that a new user is actually unreachable because no location table entries exist for that node's UID! How can the PKG send a reply to this user? If we allowed temporary unverified entries in the location registers, we would open a security hole that could be used for a DoS attack. So, for this special case, we use source routing. Specifically, when the new user's message is sent to the PKG, it accumulates a certified route. The PKG simply reverses this route and source routes this reply. This allows the new user to be added to the network without trust violations. Once the user is added, it can REGISTER its location with a signed message.

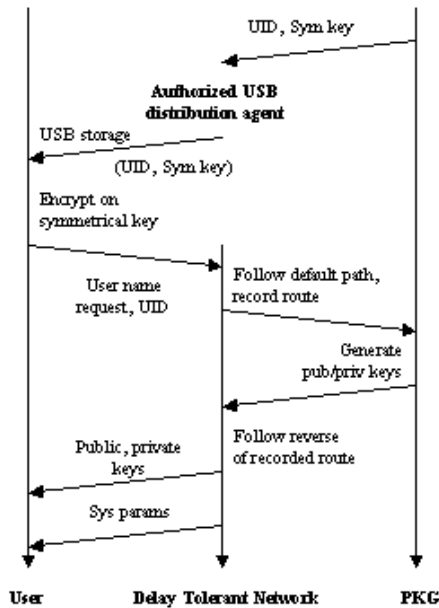


Fig. 3: Establishment of security parameters for new users

#### 5.4. Preventing identity theft due to loss of mobile devices

Users will generally access the DTN infrastructure through mobile devices like cell phones and PDAs. However, such devices can be easily lost, which also implies a loss of the identity. Our solution is to never keep the actual private key on the PDA, but to extend the key hierarchy by another level that is time-based. In other words, public and private keys for  $(ID_1 \dots ID_i)$  are extended to  $(ID_1 \dots ID_i, Date)$ . Note that this method of generating time-based keys is different from the key-revocation method explained in Section 7. Here, the tree is extended an additional level  $(ID_1 \dots ID_i \rightarrow ID_1 \dots ID_i, Date)$

which can be done by the user acting as a PKG for itself, but in the key-revocation method the public key is changed by concatenating a timestamp  $(ID_1 \dots ID_i \rightarrow ID_1 \dots Date-ID_i)$ , and hence this can be done only by the actual PKG. Furthermore, the granularity of change is likely to be finer for local time-based keys.

These time-based keys are generated for each new day in a secure location, such as on a desktop, and downloaded to the PDA periodically, say every few days. Thus, even if the PDA is lost, an intruder will gain access for only a limited amount of time (i.e. until the keys on the PDA remain valid). Even this access can be prevented by prompt action to quarantine all resources belonging to the user till the time-based keys have expired. The advantage with time-based keys is that the duration of compromise is limited. We believe this is an adequate practical solution for forward secrecy in IBC systems.

### 6. Mutual authentication

We now consider the case when a local DTN router meets a disconnected node. How can the two nodes mutually authenticate themselves? Two cases arise:

6.1. If the local DTN router belongs to the same provider as that of the user, a 1.5 RTT challenge-response protocol is used to verify the authenticity of the user and the local DTN router [13].

- a. User  $\rightarrow$  Local Router:  $\text{Pub}_{\text{DTN}}(\text{nonce}_1, \text{Pub}_{\text{User}})$
- b. Local Router  $\rightarrow$  User:  $\text{Pub}_{\text{User}}(\text{nonce}_1, \text{nonce}_2)$
- c. User  $\rightarrow$  Local Router:  $\text{nonce}_2$

This protocol verifies that both the user and the DTN router are who they claim to be, and they possess valid private keys because they are able to decrypt each other's random messages. The infrastructure is assumed to belong to a Trusted Computing Base (TCB), and hence it relies on the ingress Local DTN router to authenticate the user. Per-hop authentication can also be done if there is a high risk of the ingress routers to get compromised, and to push fake traffic into the network. In addition, all DTN nodes have the "DTN" string as a prefix in their public ID, which can be used as an additional safeguard.

6.2. The local DTN router can also belong to some other provider referred to as the *roaming provider*. This can occur if a bus drives past a PDA in a remote region, or if the PDA is taken into a remote kiosk, where the bus and the kiosk belong to a provider other than the home provider of the user. There are two cases in such a scenario:

#### 6.2.1. If guest access is allowed

We explain this through an example illustrated in Fig. 4 that uses the notion of chains of trust. It illustrates a scenario where Bob, who is a user of provider P1, roams to access service from a kiosk that is owned by provider P2. To authenticate Bob, the kiosk uses P1's system parameters

signed by P2, and Bob's public key signed by P1. Since the kiosk trusts P2, it infers that P2 has allowed access to P1's users because P2 signed the system parameters of P1. Now, since Bob is a valid P1 user, hence the kiosk grants access to Bob through the chain of trust. Similarly, Bob verifies that he can trust P2's kiosks by looking at P2's system parameters signed by P1, and the kiosk's public key signed by P2. This shows that our scheme works well despite the entities being disconnected from each other. Note that HIBC is not necessarily required for this scheme, and PKI is usable as well.

The mechanism for authentication and billing is as follows.

i. We slightly modify steps (a, b) in the 1.5 RTT challenge-response protocol shown earlier for the single provider case. As explained in the example above, along with their respective public keys, both entities also furnish their respective system parameters signed by a well-known signing authority like Verisign, or signed by the correspondent entity's home provider. Once the system parameters have been negotiated and verified, the same protocol can be used to authenticate both parties.

ii. Authentication tokens are created and signed by users through HIDS for each bundle or group of bundles. These tokens contain the user identifier, roaming provider identifier, and the identifier number of the bundles. The signed tokens and the bundles are sent by the end-host to the local DTN router of the roaming provider.

iii. The roaming provider verifies that the identifier numbers of all the bundles are included in the token, and sends a signed acknowledgement back to the user. Note that both entities can sign the tokens on their own respective private keys, and the signatures can then be verified by the correspondent entity irrespective of whether it belongs to another provider.

iv. The tokens are then relayed to the home provider by the roaming provider in order to impose charges on the user.

v. The signed tokens prevent any tampering attacks before presenting the tokens to the home provider. To avoid replay attacks, all authentication tokens carry the sequence number specified by the user as well. The roaming providers have to be careful in collecting and ordering the tokens according to the sequence number before submitting them to the home provider.

vi. The user stores the signed acknowledgement for auditing purposes to detect incorrect billing.

vii. This is to support a 'guest access' method to allow data based *post-payment* of roaming services [10].

Note that the above mechanism can be used only for sending data through a roaming provider's network. However, if data needs to be received in a roaming provider's network as well, then appropriate routing tables need to be set up in the roaming HLR, VLRs, and LLR as well. This can be achieved if the user is granted a temporary time-based identity by the local DTN router in the roaming network and is similar to the roaming token method explained next.

### 6.2.2. If roaming tokens are granted in advance

In this scheme, the soon-to-be-mobile user is given, in advance, a time-based private key and system parameters of a roaming provider. This can be done through an initial inter-federated secure communication between the user and the home provider, and the home provider and the roaming provider. The same 1.5 RTT challenge-response protocol is then used for authentication purposes between the roaming user and the local DTN router of the third-party provider. The temporary identity is used to set up routing tables in the roaming provider's network, so that the user can even receive data in the roaming network. The roaming token method is meant to support data based *pre-payment* of roaming services.

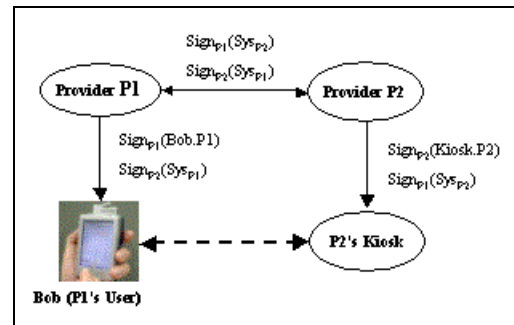


Fig. 4: Chain of trust

### 6.3. Mutual authentication of routing information

We can use HIBC to guarantee authenticity of routing information given to a mobile host by a mobile local router. Note that the information required by the user consists of the identity of the custodians that the mobile DTN router visits, the regions of these custodians, and routing and scheduling information of the mobile DTN router. We ensure that this information is correct and up-to-date by double signing tuples of  $(Information, Custodian DTN node, Mobile local DTN router, Current time)$ , first by the custodian DTN node, and then by the mobile DTN router (For example, the signature may look like  $Pvt_{Custodian-DTN} (Pvt_{Mobile-DTN} (Information, Custodian-DTN@RI.HI.P, Mobile-DTN@RI.HI.P, Time))$ ). The double signing and embedding of identities of the DTN nodes within the tokens makes the scheme secure against imposter attacks even if some eavesdropper or rogue DTN router intercepts the signed tuples and tries to replay them.

## 7. Distributed key revocation

Instead of using CRLs, we use time-based keys [2] for revoking the rights of compromised infrastructure nodes and malicious users. We assume that clocks of all the entities are synchronized within a small margin of error. Shortly before the end of a *refresh interval*, a PKG updates every endpoint with new timestamped private keys. The refresh interval should be much larger than the allowed margin of errors in clock synchronizations. The interval can be interpreted as representing a tradeoff between the scalability of refreshes and the level of security needed, with a faster refresh implying

a higher security level. The interval will also depend on the schedules of intermediate links, and can be adjusted to a maximum threshold limit that depends on the operational environment.

Our system automatically concatenates all public IDs with the last refresh time, transparent to the communicating entities, as described below. Provision is made for a small lag period before and after the time instance of change during which both old and new keys can be used. Each entity tells its correspondent node its public key and last time of change as well as a certificate signed by the root PKG (or a well-known third party certifying authority) with the value of the refresh interval and the public ID of the entity. This allows the correspondent node to correct decide if the time-based key of the entity has expired since the last time of change because both the time-based private key and the refresh interval are unforgeable.

A time-based key is considered expired one refresh interval after the last time of change. A user with an expired key is automatically refused access into the trusted computing base at the ingress local DTN router. Similarly, no valid user or infrastructure node accepts communication from an infrastructure node with an expired key. This means that once the infrastructure detects the compromise of any node in the system, that node can be excluded from the trusted computing base after one refresh interval.

Note that using time-based keys imposes an additional computational and communication overhead on the entire system. However, it has the property that it is fail-safe, that is, a failure in the system does not compromise security. In contrast, CRLs are less expensive, but open a security hole in case of a failure. Developing efficient techniques to disseminate time-based keys is an interesting area for future work. Also, note that these time-based keys are likely not the same as the keys used for end-to-end encryption, authentication, and integrity. End-to-end IBC keys will usually not be time-based because a sender has no way to know a recipient's last refresh time.

## 8. Discussion and related work

Our work presents practical solutions to providing secure channels, mutual authentication, and rights revocation in networks with disconnected nodes. We exploit several inherent strengths of HIBC, and, assuming security of the PKGs, provide simple solutions to these challenging problems. Security in disconnected environments has only recently been studied in the literature. Related work includes:

*HIBC for DTN:* The use of HIBC for DTN as well as the use of time-based keys for access control was proposed in [2]. In contrast, we provide *practical* schemes for key dissemination, mutual authentication, secure location updates, audited custody transfer, and time-based forward secrecy and revocation.

*Offline authorization frameworks for ubiquitous computing:* The 'Lobby' system proposed in [12] is typical of secure ubiquitous computing architectures like UPnP and Jini. These architectures provide an authorization framework that support offline mobile devices. These schemes require all policy-enforcement-points (or 'Lobby's) to periodically connect to a central database and renew their user and role based ACLs. Our work supplements such architectures by providing a general platform over which fine-grained trust models can be built to provide policy-based access control at the ingress points to the DTN TCB.

## 9. Acknowledgements

We gratefully acknowledge Kevin Fall at Intel Research, Berkeley for his seminal suggestions on the use of Identity Based Cryptography for security in disconnected environments. We also benefited from extensive discussions with Stephen Fung at U. Waterloo.

## 10. References

- [1] A. Seth, P. Darragh, S. Keshav. "A Generalized Architecture for Tetherless Computing in Disconnected Networks," *Work in progress: <http://mindstream.watsmore.net>*
- [2] K. Fall, "Identity Based Cryptosystem for Secure Delay Tolerant Networking," *Manuscript*. Intel Research, Berkeley, December 2003.
- [3] Craig Gentry, Alice Silverberg. "Hierarchical ID-Based Cryptography," In *Proc. International Conference on the Theory and Application of Cryptography and Information Security*, 2002.
- [4] D. Boneh, M. Franklin. "Identity Based Encryption from the Weil Pairing," In *Proc. Crypto 2001 Lecture Notes in Computer Science, Vol 2139, Springer Verlag*, 2001.
- [5] K. Fall. "A Delay Tolerant Network Architecture for Challenged Internets," In *Proc. SIGCOMM*, 2003.
- [6] B. Lampson. "Computer Security in the Real World," *IEEE Computer*, June 2004.
- [7] C. Gentry. "Certificate based encryption and the certificate revocation problem," *Proc EUROCRYPT*, pp 272-293, 2003.
- [8] D. Yao, N. Fazio, Y. Dodis, A. Lysyanskasa. "ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption", In *Proc. ACM Conference on Computer and Communications Security*, 2004.
- [9] J. Mirkovic, S. Dietrich, D. Dietrich, P. Reiher. "Internet Denial of Service: Attack and Defence Mechanisms", *Prentice Hall PTR*, 2005.
- [10] A. Seth, S. Fung, S. Keshav. "A Secure Tetherless Computing Architecture", *Tech report, University of Waterloo, Canada*.
- [11] J. Ott, D. Kutscher. "A Disconnection-Tolerant Transport for Drive-thru Internet Environments", In *Proc. Infocom*, 2005.
- [12] K. Zhang, T. Kindberg. "An Authorization Infrastructure for Nomadic Computing", In *Proc. ACM Symposium on Access Control Models and Technologies*, 2002.
- [13] A. Menezes, P. Oorschot, S. Vanstone. "Handbook of Applied Cryptography," *CRC Press*, October 1996.

# Achieving K-Anonymity in Mobile Ad Hoc Networks \*

Xiaoxin Wu and Elisa Bertino

Department of Computer Sciences, Purdue University  
wu@cs.purdue.edu, bertino@cerias.purdue.edu

## Abstract

*A zone-based K-anonymity routing protocol is proposed and investigated, the goal of which is to achieve destination anonymity in positioning routing algorithms. Under this protocol, the source sends the data to an anonymity zone, where the destination node and a number of other nodes are located. The data is flooded within the anonymity zone so that to a tracer is not able to guess which is the destination node. The paper presents the details of the protocol design and proposes two approaches for anonymity zone management. In particular, we have found out that initially setting anonymity zone large can help to meet the destination anonymity requirement for longer time at a relatively low control overhead.*

## 1 Introduction

Mobile wireless ad hoc networks (MANET) have several advantages such as fast deployment, low cost bandwidth, and flexible application environment. In a MANET, the network is self-organized, and its function heavily depends on the collaboration among ad hoc users, i.e., ad hoc nodes. In most ad hoc routing algorithms, nodes have to disclose their IDs. This makes the node activity, i.e., transmitting or receiving, highly traceable. A key concern in such a context is represented by anonymity, which is considered an important requirement for privacy, and has been widely investigated for database systems and conventional networks. One important approach for achieving anonymity, proposed independently for database systems and fixed networks, is to hide the entity of interest among a group of similar entities [6, 2, 3]. For example, under the  $k$ -anonymity [6] approach, the object of interest is always hidden among  $k$  entities. The application of such an approach to MANET is however very challenging because of the highly dynamic nature of these systems. To date no approach has been reported dealing with such issue.

This paper investigates mechanisms for achieving communication anonymity in ad hoc networks. We focus on applying position-based ad hoc routing algorithm to achieve a level of destination anonymity, which is more challenging than preserving source anonymity. The research is motivated by the fact that when an ad hoc user is receiving sensitive data, it may not be willing to let its peer ad hoc users know.

A typical position-based ad hoc routing algorithm is the Greedy Perimeter Stateless Routing (GPSR) [4]. In GPSR, node position information is available. The source or the forwarder knows its own position as well as the positions of its neighbors and the destination. The position of the destination is carried along the route; thus a forwarding node can determine its next hop locally, by selecting the node which is the closest to the destination, as the source does. In the position-based routing algorithm, the routing information carried in the data packets can only be positions. Node IDs are not required. Therefore, the destination anonymity can be preserved if the match between an ID and the position is kept private. However, because position information needs to be shared among the nodes for routing purpose, it is more likely that a node's ID can be obtained if its position is known. For example, an inside tracer can request all the node positions to get this match.

In this paper we propose an approach that addresses the above problem. The approach is based on the concepts of "false" position, instead of the real position of the destination, and of "crowd" of nodes, that is, a set of nodes in an area around the false position.

The "false" position is close to the destination to which a packet is sent. Because the destination is nearby and due to the broadcast nature of wireless communication, the destination can still receive the packet. Because the destination will be among a "crowd" of nodes in an area around the false position, the approach can assure anonymity for the destination even if at the time of communication, both its position and ID are known. It is important to notice that the anonymity of the destination depends on the crowd size. Since the area around the false position cannot be too large (i.e., the false position cannot be too far away from

\*The work is sponsored by I3P (Institute of Information Infrastructure Protection) Fellowship and by the sponsors of CERIAS.

the real position), otherwise the destination cannot receive the data packet, there may be only a few nodes residing in the area when node density is not high. The crowd size is small and consequently, a required anonymity set, e.g., the  $k$ -anonymity, may not be achieved.

To expand the anonymity set so as to achieve  $k$ -anonymity in networks with different node densities, the area where the crowd is located needs to be expanded. In the remainder of this paper, we describe a protocol for MANET, based on the above notions, that we refer to as zone-based  $k$ -anonymous routing protocol.

In section 2 we give the protocol details, whereas in section 3 we discuss in details the anonymity of the source and the destination. In sections 4 and 5, we analyze anonymity and overhead. Finally in section 6 we outline future work.

## 2 Zone-based K-Anonymous Routing Protocol: An Overview

### 2.1 Network and Privacy assumptions

Ad hoc network assumptions are as follows:

- We assume that nodes are uniformly distributed with a relatively high node density, and that the number of nodes within any area is not too small. We assume flooding is robust, that is, any node in the flooded area can receive the flooded message. For node mobility, we assume a node moves toward a random direction at a variable speed.
- We assume each node to know its own position, e.g., through a GPS system [1]. We also assume that a position service system such as those proposed in [5, 8] is available, through which a node is able to retrieve the position of any other node. Based on the position information, positioning routing algorithms such as GPSR can be used.
- We assume the radio channel to be symmetric. Once a node sends a message to another node through a multi-hop connection, the receiving node can send back a message to the sender following the reverse route. We assume that ad hoc nodes have the same transmission and receiving range, which in this paper is also called the ad hoc channel coverage.

Privacy assumptions are as follows:

- Each node has a public key that is known to all the other nodes. This public key is assigned to the node by an off-line certificate authority (CA) when the node joins the network. A node may store its public key in the same way as it stores its position, so that other nodes can obtain it.

- We assume that there are internal attackers able to trace or monitor the behavior of other nodes for malicious purposes. Such an attacker is also called a *tracer* or an *observer*.
- An attacker is able to eavesdrop the communication channel. However, we assume that it cannot identify a node or tell whether two transmissions are from the same node based on transmission signal signatures. If the ID of the sender is not carried in a transmitted packet, or the ID is encrypted and cannot be read the attacker, the attacker is not able to tell which the sender is.
- An attacker is able to obtain the updated position for every node. For an end-to-end packet delivery (not our approach), we assume that an attacker can obtain the path information. It knows which are the forwarding nodes and the destination, and where they are. However, the attacker does not know which the source is or where the source is, because this information is not required for correct positioning routing.
- We assume that an attacker does not know either which node has requested positions of other nodes or which positions have been requested.

### 2.2 Zone-Based Anonymous Routing Algorithm

Figure 1 illustrates the general idea of zone-based  $k$ -anonymity routing protocol for destination anonymity when positioning routing algorithms are used in MANET. The source sends data to an anonymity zone (AZ), instead of the destination. The AZ is normally a circle-shaped area where the destination and a crowd of at least  $k - 1$  other nodes are located. Since the destination may not directly receive the packet, a local flooding is used within the AZ. As the anonymity zone is generated for the destination, in the rest of the paper, we also call it the destination anonymity zone (D-AZ).

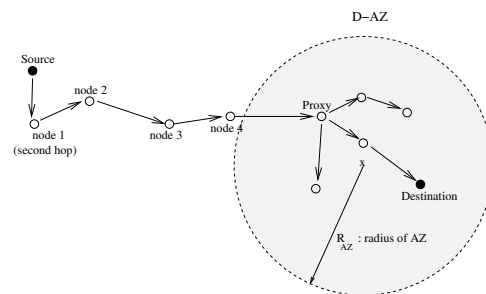


Figure 1.  $k$ -anonymity based private positioning routing.

When a source node needs to build a route to a destination, it obtains the position of the destination through the position services. It then randomly selects a point as the center of the D-AZ as well as the radius for the D-AZ. The radius should be large enough so that 1) the destination is located in the D-AZ; and 2) there are at least  $k - 1$  other nodes located in the D-AZ. The position and the radius of the D-AZ will be carried in the routing request (*rreq*) message in the plain text, which is forwarded toward the D-AZ using position-based routing algorithm. Every intermediate node which forwards the *rreq* will piggyback their IDs in the message.

To keep the destination private from the rest of the network, the *rreq* carries the destination challenging information. It includes a symmetric key encrypted by the destination's public key, and the IDs of both the source and the destination, which are encrypted by the symmetric key. The symmetric key is also used for the subsequent data transmission. Generating the destination challenge in this way instead of simply using public key to encrypt IDs can reduce information leakage.

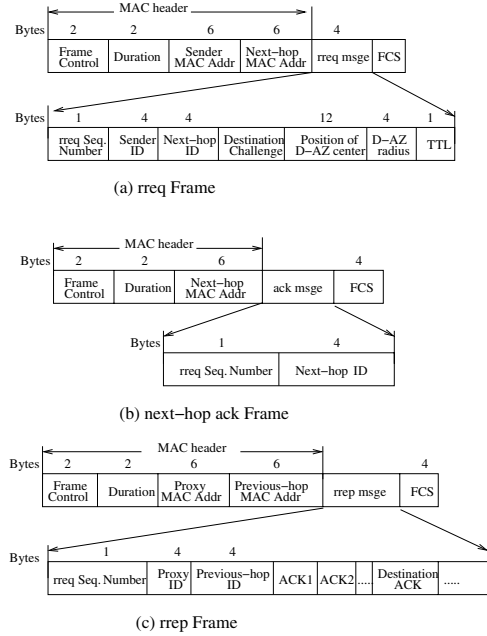
A node determines whether it is within the D-AZ by using the position information. The node in the D-AZ which first receives the *rreq* acts as a proxy and locally floods<sup>1</sup> the message to all the other nodes in the D-AZ. A node receiving the *rreq* replies to the proxy with an ACK using the reverse path it receives the *rreq*. The proxy then sends a routing reply (*rrep*) to the source following the reverse path, i.e., any node in the route forwards the *rrep* to its previous hop. Once the message reaches the next hop of the source, this next hop broadcasts the message so that the source can receive it. The *rrep* carries all the ACKs (including the ACK from the proxy) and the IDs of the intermediate nodes between the source and the D-AZ. The detailed protocol design for ACK collection is beyond the scope of this paper.

Upon receiving the *rrep*, the source checks the ACKs to determine whether the destination is in the D-AZ. If so, the source starts transmitting the data packet using the discovered route. As *rreq*, the data packet will be received by a proxy, and then be flooded in the D-AZ. All nodes including the destination can receive the packet. Data is encrypted by the symmetric key for privacy.

The packet frames for the major control messages in route discovery are illustrated in Figure 2. We leave the length of destination challenge open as it depends on what the symmetric cryptographic algorithm is used.

To a tracer, the information is that a group of nodes are receiving the packets. Yet the tracer cannot determine which node is the real destination. It should be noted that since the source normally does not know the exact network topology, and the D-AZ is generated based on node density,  $k$ -anonymity for the destination can only be achieved with

<sup>1</sup>Only a node located within the D-AZ will retransmit the message.



**Figure 2. Packet frames for major route discovery messages. Note that in *rreq*, if the sender is the source, pseudo MAC ID and node ID are used where the sender's MAC ID and node ID appear.**

a probability.

During data transmission, the entire D-AZ is treated as a single node. If the node density is not extremely low (which is the network scenario considered in this work), flooding is robust and highly reliable. Therefore, a packet delivery to the destination can be considered to be successful if it is received by the proxy in the D-AZ. Only an ACK from the proxy is needed for acknowledging the message. The source may occasionally trigger an ACK directly from the destination. In this case, an ACK collection by the proxy similar to that in the routing discovery stage will be executed. Since nodes are mobile, different packets may have different proxies.

A routing failure occurs if the data packets from the source cannot be received by any node (i.e., the proxy) in the D-AZ. This happens when the route between the source and the D-AZ breaks due to the mobility of the source or the intermediate nodes. In this case, a routing recovery can be executed immediately, where the same D-AZ is used for the destination.

A routing failure also occurs if the destination moves out of the D-AZ<sup>2</sup>. To find a new route, the source has to use a different D-AZ. The use of different D-AZs for the same

<sup>2</sup>The destination may make the source aware of its departure by sending it a message using the same anonymous routing algorithm.

destination may be prone to the so-called intersection attack, in which a tracer can reduce the anonymity size by finding the common nodes in these two D-AZs. To avoid the intersection attack, the source needs to make the two routing processes unlinkable. One possible solution is that after the destination moves out of the D-AZ, the source may wait for a period of time before it starts the new routing discovery.

### 3 Anonymity

#### 3.1 Source Anonymity

If a tracer intercepts a transmission (especially, a route discovery message) from the source, it can estimate the anonymity set for the source, which is the number of the nodes residing in the overlapped zone of the area covered by the receiving range of the tracer and the second hop. The anonymity size depends on the size of the overlapped area, i.e., the distance between the tracer and the second hop as well as node distributions.

A tracer can estimate the anonymity set of the source only if it is close to the source and receives a packet from the source directly. In particular, a tracer can make a judgment that a received message is from a source only if it stays very close to the source-like node, so that it can intercept every message delivered to it. Only in this way, it can be sure that the message it received is originally generated from this node (i.e., source), and not a message received and forwarded by it. However, because nodes are mobile, it is difficult for a tracer to always stay within the transmission range of a node. In this case, node mobility improves source anonymity.

Note that the above is the best case for the source anonymity. The anonymity decreases if the tracer is able to use advanced technique such as directional antenna.

#### 3.2 Destination Anonymity

The nodes in the D-AZ determine the destination anonymity. As the D-AZ information is carried along the entire route, it is more likely that a tracer intercepts the D-AZ information. It then can determine the anonymity set for the destination based on the information of the D-AZ and the network topology.

Because nodes are mobile, nodes may move in or out of the D-AZ. This may cause a well-known intersection attack. Node mobility will degrade destination anonymity as time elapses. Figure 3 shows an example of intersection attack. Suppose that two deliveries to the D-AZ are observed at different time, and the anonymity sets for the first and the second deliveries are  $set_1$  and  $set_2$  respectively. It is easy for an attacker to infer that the destination node is either  $e$

or  $f$ . The size for the anonymity set under the intersection attack is 2, instead of 6 for  $set_1$  or 5 for  $set_2$ .

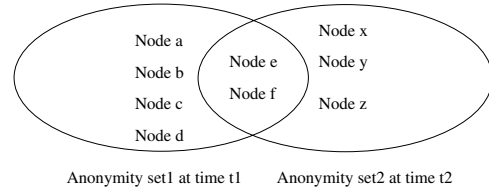


Figure 3. Example of intersection attack.

To address this problem, we propose two approaches for D-AZ management, where  $k$  is assumed to be the required anonymity size.

1. **Fixed D-AZ.** Under this approach, the source originally uses a large-sized D-AZ so that more than  $k$  nodes will be in the D-AZ. As time passes, even if some nodes move out of the D-AZ, there are still enough nodes staying in the D-AZ to keep the anonymity requirement.
2. **Adaptive D-AZ.** Under this approach, the source determines the size of D-AZ based on node density so that  $k$  nodes will be originally in the D-AZ. As the time passes, the size of D-AZ increases. A moving-away node that was in the original D-AZ then may still be in the enlarged D-AZ. The source estimates when to increase the D-AZ or how fast the D-AZ should be increased according to the anonymity requirement, node density, and mobility.

### 4 Analysis

This section presents the analysis for the proposed approaches. Especially, the destination anonymity is analyzed. The approaches are evaluated with respect to the probability of maintaining the anonymity size. We assume the desired size for the destination anonymity set is  $K_r$ .

#### 4.1 Anonymity Analysis on Fixed D-AZ

We assume nodes are moving randomly with a fixed average speed. Based on the results in [7], the probability density function for the time that a node may stay in an D-AZ, defined as  $f_{t_d}(t_d)$ , can be approximated as an exponentially distribution function with a mean time of  $\bar{t}_d$ , and

$$\bar{t}_d = \frac{\pi S}{E[v]L} = \frac{\pi R_{AZ}}{2E[v]}, \quad (1)$$

where  $S$ ,  $L$ , and  $R_{AZ}$  are the area, the perimeter, and the radius of the D-AZ respectively, and  $E[v]$  is the average speed of the mobile node.

Let  $p$  be the probability that after a time  $t_1$ , a node will stay in the D-AZ, then:

$$p = p\{t_d > t_1\} = \int_{t_1}^{\infty} f_{t_d}(t_d) dt_d = \int_{t_1}^{\infty} \frac{1}{\bar{t}_d} e^{-\frac{t_d}{\bar{t}_d}} dt_d = e^{-t_1/\bar{t}_d} \quad (2)$$

Let  $\rho$  be the node density. In this approach, as the radius for the D-AZ is  $R_{AZ}$ , the initial number of nodes residing in the D-AZ is  $n_0 = \rho\pi R_{AZ}^2$ . After time  $t$ , the probability that besides the destination, there are other  $i$  nodes staying in the D-AZ, denoted as  $P\{n = i\}$ , is:

$$P\{n = i\} = \binom{n_0 - 1}{i} p^i (1 - p)^{n_0 - i - 1}, \quad (3)$$

where  $p$  is obtained by Eqn. (2).

After time  $t$ , the probability that besides the destination, there are at least  $K_r - 1$  nodes staying in the D-AZ (i.e., the  $K_r$ -anonymity is achieved), denoted as  $P\{n \geq K_r - 1\}$ , is:

$$P\{n \geq K_r - 1\} = p \left( 1 - \sum_{i=1}^{K_r-1} P\{n = i\} \right). \quad (4)$$

$p$  on the right of the equation denotes the probability that the destination stays in the D-AZ.

## 4.2 Anonymity Analysis for Adaptive D-AZ

In the adaptive D-AZ approach, the initial radius for the D-AZ, denoted by  $R_0$ , can be calculated by:

$$R_0 = \sqrt{\frac{K_r}{\pi\rho}}. \quad (5)$$

Let  $P_k(t)$  be the probability that after a time of  $t$ , all these  $k$  nodes stay in the D-AZ, i.e., the probability that a  $K_r$  anonymity size can be kept, then:

$$P_k(t) = e^{-K_r t / \bar{t}_d}. \quad (6)$$

We assume there is a probability threshold value  $p_0$ , and if  $P_k(t) \leq p_0$ , the D-AZ has to expand. Assume that at time  $t_0$ ,  $P_k(t)$  decreases to  $p_0$ . According to Eqn. 6,  $t_0 = -\ln(P_k \bar{t}_d / k)$ . After time  $t_0$ , the size of D-AZ has to be increased. Let  $R_{AZ}(t_1)$  be the function for D-AZ expansion. The probability that there are  $k$  nodes staying in the D-AZ after a time equal to  $t_0 + t_1$  has elapsed, denoted as  $p(t_1)$ , is:

$$p(t_1) = e^{-\frac{2K_r v(t_0 + t_1)}{\pi(R_0 + R_{AZ}(t_1))}}. \quad (7)$$

We need to find a function of  $R_{AZ}(t_1)$  so that as  $t_1$  changes,  $p(t_1)$  remains constant. This can be solved by deriving Eqn. (7) and equating the result to 0, which induces to a problem of differential function. It is easy to find such a function for  $R_{AZ}(t_1)$ , which is:

$$R_{AZ}(t_1) = c(t_1 + t_0) - R_0 \quad (8)$$

where  $c = R_0/t_0$ .

## 5 Analytical Results

In this section we show some analysis results on destination anonymity. Unless otherwise specified, for the analysis, the network node density is set as  $100/km^2$ . The average node speed is  $1m/s$ . The radius for the anonymity zone is  $250m$ . The desired size for the anonymity set is 8.

Figure 4 shows the impact of the D-AZ size and node density on destination anonymity. As the size of the D-AZ or the node density increases, there is a higher probability of keeping the destination anonymity set to the desired value. The reason is that as either of these two parameter increases, there are more ad hoc nodes originally residing in the D-AZ. It then takes longer time for the number of nodes to drop below the desired value. When node density is low while the size of D-AZ is small, the time elapses, the destination anonymity degrades fast. However, except for the case when  $\rho = 50$  and  $R_{AZ} = 250m$ , there is a high probability of maintaining the desired destination anonymity for a relatively long communication duration time. The probability that the proposed scheme can meet different anonymity requirements is shown in Figure 5. The probability of achieving a low anonymity requirement is higher.

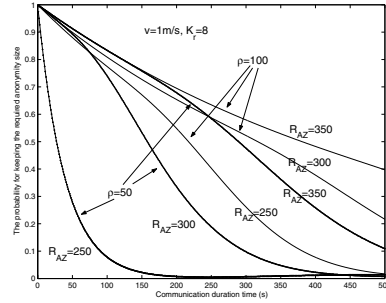


Figure 4. Anonymity vs. D-AZ size and Node density.

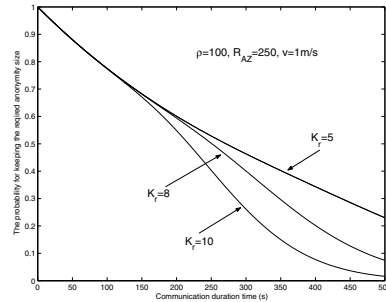


Figure 5. Anonymity achievement.

Figures 6 and 7 report the probability of providing the required anonymity size and the corresponding control over-

head (i.e., the number of nodes to which the packet has to be sent) in both the adaptive D-AZ and the fixed D-AZ approaches. The required anonymity size  $k$  is set to 8 and in the adaptive D-AZ approach, the probability threshold value  $p_0$  is set to 0.8. As we can see from the figures for short communication times, the fixed D-AZ approach results in better anonymity, with the overhead only marginally higher than in the adaptive D-AZ approach. When communication time is long, the adaptive D-AZ approach can still satisfy the anonymity requirement whereas the fixed D-AZ approach cannot. However, the tradeoff is a higher overhead. In summary, the fixed D-AZ approach has the better overall performance. The reason is that in the fixed D-AZ approach, the anonymity size can be achieved if a number of nodes is always present in the D-AZ. These nodes can be any nodes from a relatively large node set. Whereas the adaptive D-AZ approach requires that the nodes which are originally in the D-AZ remain in the expanded D-AZ. When the communication time is short, the adaptive D-AZ approach can be used. Otherwise, the fixed D-AZ approach or a joint approach should be used.

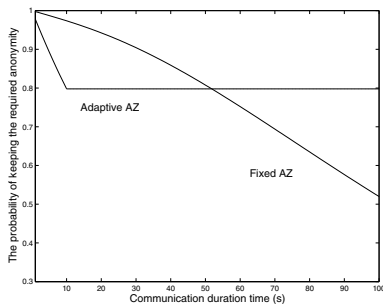


Figure 6. Anonymity for different schemes.

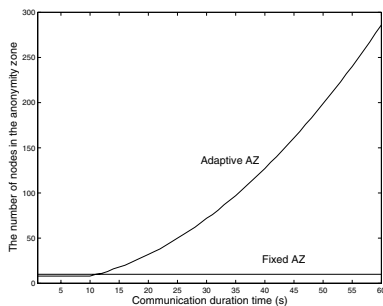


Figure 7. Overhead for different schemes.

## 6 Conclusion and Future Work

This paper proposes a zone-based anonymous positioning routing algorithm for mobile ad hoc networks. Node

anonymity, especially the anonymity of destinations, can be achieved with a trade off in communication overhead. In the paper we have analyzed the destination anonymity and communication overhead. An important conclusion from the analysis is that even when the node density is not very high and the D-AZ size is not very large, the proposed approach can maintain a relatively high anonymity set for the destination for a not short time. The adaptive D-AZ scheme can further improve destination anonymity. However, if the communication time lasts too long, the source should better restart a session to avoid a extremely high overhead for local flooding.

The detailed protocol, including the control message frames that are involved in route discovery and maintenance, are being developed. Future work also includes the development of a simulation model to evaluate routing performance such as network end-to-end throughput and delay. The performance degradation will be looked as the cost for anonymity achievement. If the cost is too high, advanced flooding algorithm will be developed for local flooding. The algorithm will use the position information to reduce redundant transmission while maintaining robustness.

## References

- [1] Navstar gps operation. In *Web site at <http://tycho.usno.navy.mil/gpsinfo.html>*.
- [2] Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):6–92, 1998.
- [3] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of ACM*, 24(2):84–88, 1981.
- [4] B. Karp and H. T. Kung. Gpsr: Greedy perimeters stateless routing for wireless network. In *Proceedings of MOBICOM*, 2000.
- [5] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *ACM Mobicom*, 2000.
- [6] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.
- [7] R. Thomas, H. Gilbert, and G. Mazziotto. Influence of the moving of the mobile stations on the performance of a radio cellular network. In *Proceedings of Third Nordic Seminar*, 1988.
- [8] X. Wu. Vpds: Virtual home region based distributed position service in mobile ad hoc networks. In *Proceedings of ICDCS*, 2005.

# Phyllo: A Peer-to-Peer Overlay Security Framework

William Heinbockel  
The MITRE Corporation  
Bedford, MA 01730  
Email: heinbockel@mitre.org

Minseok Kwon  
Department of Computer Science  
Rochester Institute of Technology  
Rochester, NY 14623  
Email: jmk@cs.rit.edu

## Abstract

*Despite the success of peer-to-peer systems, a majority of their overlay architectures are vulnerable to exploitation. Some of the features for improved performance have created security holes that attackers can breach to gain control of the network. De facto network security solutions (e.g., trusted servers, encryption, and firewalls) offer little assistance, as they are often not compatible with the open and decentralized structure of peer-to-peer networks.*

*To address overlay security problems, we propose Phyllo, a node-partitioning framework that isolates untrusted nodes from the core network. Yet, the isolated nodes can still participate in peer-to-peer communications. Our partitioning scheme also allows nodes to move between partitions, while introducing marginal performance overheads. Our experimental results indicate that Phyllo supports more reliable message delivery in the presence of malicious nodes.*

## 1. Introduction

Imagine the damages an attacker may cause if he controls an entire network. The attacker can filter and manipulate all inbound and outbound network traffic. Within peer-to-peer (P2P) systems, if there exist only a small percentage of attackers throughout the network, then the potential for malicious activity is relatively low. As the fraction of attackers increases, however, the attackers may potentially dominate larger portions of the network. The potential for the domination escalates even further if these attackers collude.

Peer-to-peer overlay networks (e.g., Chord [12] and Pastry [10]) currently offer few solutions for handling potential attacks [14, 2, 11]. Most overlays assume that malicious nodes will never comprise a considerable percentage of the network. To the best of our knowledge, however,

there are few efficient and reliable solutions for security issues in overlay networks.

We introduce our framework, *Phyllo*, to reduce the detrimental effects that untrusted nodes might cause on overlay networks. The basic idea of Phyllo is to partition the fundamental overlay structure in order to isolate damages from adversaries within a single partition. Promotion and demotion protocols determine which nodes are transferred between partitions. A customizable evaluation platform is available for each network to specify the conditions for this transfer. Our framework can be easily implemented on popular overlay network systems such as Pastry. Phyllo requires at most two additional routing hops, and minimal processing and communication overheads.

The remainder of this paper is organized as follows. Section 2 gives motivation for our framework. In Section 3, we discuss the basic architecture of Phyllo including its routing mechanism and promotion/demotion protocols. In Section 4, we present results from the experiments using our implementation of Phyllo on Pastry. Section 5 gives an overview of related work. Finally, we conclude the paper and summarize future work in Section 6.

## 2. Motivation

The primary motivation behind our framework is the architectural security problems in overlay networks detailed in [14, 2, 11]. An attacker may exploit such problems to control certain routing paths, or cause unnecessary communication and routing overheads. Our architecture aims at the five main security issues outlined in [14] and [11]:

- **ID assignment problem** occurs when an attacker exploits the assignment of node identifiers (IDs) to obtain a specific node ID. Networks such as Pastry, which can use a node's IP address to cluster IDs based on their locality, allow attackers to control specific portions of the network.

- **Message forwarding** is an overlay-centric version of the Byzantine Generals Problem. Since peer-to-peer networks rely solely on other peers for routing, we should ensure that a message is properly routed and not be modified in transit.
- **Routing table maintenance** in dynamically routed overlays such as Pastry, is susceptible to malicious nodes propagating bad routing information. Without any extra detection mechanism, all incoming data are assumed to be reliable in peer-to-peer overlay networks.
- **Lookup attacks** involve a node directing lookup requests to wrong or invalid nodes, ultimately returning an incorrect response.
- **Denial of service attacks** can easily be performed against overlay networks. These attacks are, however, not considered as a principal motivation in our framework.

The taxonomy described in [8] offers an overview of the procedures that peer-to-peer networks should adopt to alleviate rational attacks as described above. With such attacks, users act in their own interest rather than the interest of the entire network. We recommend using incentives instead of punishing new users to protect the network. These suggestions are taken into account to design a deployable secure overlay architecture.

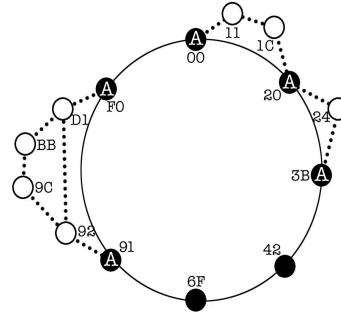
### 3. Phyllo Framework

The objective of Phyllo is to provide a flexible security interface for peer-to-peer overlay networks. In this section, we describe the details of the Phyllo framework. We first define a few terms and the basic architecture. Then, we discuss the routing algorithm and promotion/demotion protocols.

#### 3.1. Architecture

Phyllo utilizes a ring-like structure similar to Pastry [10] and Chord [12]. The framework is, however, not limited to only these two networks. The information presented here can easily be adapted to any structured overlay network with unique node identifiers.

To minimize the effect of malicious nodes, we suggest partitioning the nodes into leaf-like structures around the circumference of the ring (Figure 1). The nodes that comprise the major partition (original ring) are referred to as *major nodes* (e.g., nodes 00, 42, and 91 from Fig. 1), while the remainder of the nodes, existing in the minor partitions, are *minor nodes* (nodes 1C, 24, 9C, etc.). We also define

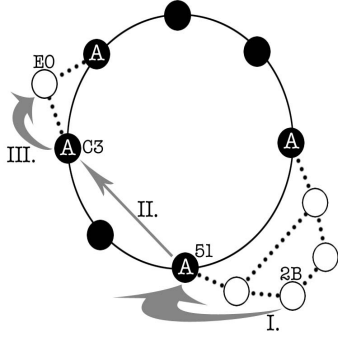


**Figure 1. Phyllo topology with major (black nodes) and minor (white nodes) partitions. Anchor nodes are shown as the black nodes with an A. The nodes are assigned node IDs in an ID space of  $2^8$ .**

*anchor nodes*, which are the two major nodes with the IDs that bound the ID range of a minor partition, in every minor partition (nodes 00, 20, 3B, 91, and F0). Anchor nodes belong to both the major partition and the minor partitions connected to the anchor nodes. Anchor nodes help filter incoming messages based on their senders. This separation essentially prevents malicious nodes from forcibly escalating from minor partitions to the major partition.

The partitioned ring topology employs a single node ID space. Initially, a new node is positioned in a minor partition as to preserve the original ordering of the ID space. Both major and minor nodes are updated with only the routing information for their partitions. For example, if a node (e.g., node 1C) broadcasts out new routing information, that broadcast is restricted to only its partition (i.e., node 11 and the anchors: 00 and 20). An anchor node maintains additional routing tables for the adjoined minor partitions. A node may move from a minor to the major partition when the node becomes qualified. Details of the promotion and demotion protocols are discussed later in Section 3.3.

Phyllo distinguishes two message types. An *application message* is the message sent by a peer-to-peer application (e.g., search, response, and file transfer messages for a file-sharing application). The other message type is an *overlay message*, which supports overlay network management independent of applications (e.g., join and routing update messages). The distinction of the two message types helps constrain overlay messages within their originating partition. An anchor node ensures that no overlay messages are propagated to other partitions, while allowing application messages to cross over partitions. This enhances security since all the attacks (Section 2), except message forwarding, rely on overlay messages. For additional assurance, each node can validate a sender ID. For example, if a sender



**Figure 2. The Stages of Minor-to-Minor Node Phyllo Routing: Phases I, II, and III.**

is part of a different partition, messages from this sender are discarded since that sender ID is unknown to the recipient.

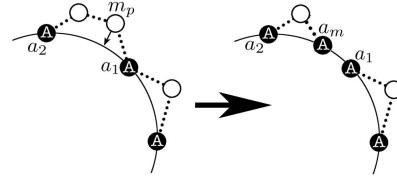
### 3.2. Routing

Within each partition, messages are routed following the network’s routing protocol. Since Phyllo uses Pastry for its basic routing protocol, every application message is routed towards the node with the ID closest to the message’s destination. To enable the partition-based routing, anchor nodes should be able to distinguish the routing in the major partition from the routing in minor partitions. The destination of a message routed in a minor partition should be between the two anchor nodes of that partition; the message is routed in the major partition otherwise.

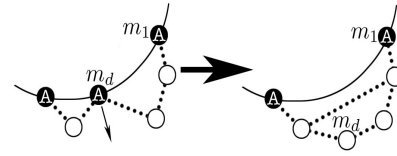
Some messages, however, require the transition between the major and minor partitions. For instance, the process of routing a message between two minor partitions (Figure 2) can be broken down into three separate stages. First, the message from a minor node is routed in the sender’s minor partition and eventually arrives at one of its anchor nodes (Phase I). The anchor node then forwards this message towards the destination (Phase II). The message continues to be routed within the major partition until the message reaches the major node closest to the destination. Finally, if the recipient major node is an anchor node and has minor nodes with IDs closer to that of the destination, the message is routed to the final destination in the minor partition (Phase III). All messages are routed using a combination of these phases.

### 3.3. Promotion and Demotion Protocols

Phyllo executes the promotion and demotion protocols to move nodes from a minor partition to the major partition, and vice versa. These protocols act as a means to reward or



**Figure 3. Phyllo Promotion Protocol: Minor node  $m_p$  is promoted by  $a_1$  to become a major anchor node,  $a_m$ .**



**Figure 4. Phyllo Demotion Protocol: Major misbehaving node  $m_d$  is demoted by anchor  $a_1$ .**

punish nodes for their behavior. The nodes are chosen to be promoted/demoted by the evaluation of a trust metric. We use the trust evaluation rate  $\alpha$  to denote the accuracy of this evaluation. For example,  $\alpha = 90\%$  indicates that the evaluation algorithm can distinguish good and malicious nodes with 90% accuracy. We assume that Phyllo estimates the trust value of a node accurately (we will investigate the trust evaluation algorithms further as future work). In addition, the promotion protocol is also used to control the size of minor partitions for lessening bottlenecks at the anchor nodes.

The promotion protocol (Figure 3) begins by an anchor node  $a_1$  identifying one of its minor nodes  $m_p$  to be promoted.  $a_1$  then contacts  $m_p$  as well as the other anchor,  $a_2$ , responsible for  $m_p$ . Once  $m_p$  receives its promotion message,  $m_p$  splits its current minor neighbor nodes’ information into two new minor neighbor sets – one for nodes between  $a_1$  and  $m_p$ , and the other for nodes between  $m_p$  and  $a_2$ . Meanwhile,  $a_1$  updates its minor partition that contains  $m_p$ , removing all nodes not between  $a_1$  and  $m_p$ . Similarly,  $a_2$  updates its minor partition. Afterwards, the anchor nodes send their minor partitions updates with regard to the promotion and new partition routing information.  $a_2$  also sends a routing update to the major partition informing of  $m_p$ ’s promotion. Once this procedure completes,  $m_p$  becomes a new anchor node  $a_m$  and divides the minor partition between  $a_1$  and  $a_2$ .

Demotion occurs in the same fashion as promotion. Instead of removing nodes and splitting partitions, the minor partitions are combined. A node  $m_1$  in the major parti-

tion targets another major node  $m_d$  for demotion (Figure 4).  $m_1$  updates the nodes in the major partition (including  $m_d$ ) of  $m_d$ 's demotion. The other major nodes remove  $m_d$  from their routing information once the demotion message is received.  $m_d$  deletes the major routing information and merges its minor partition routing information together to be new minor routing data.  $m_d$  finally notifies all the nodes in the new minor partition of  $m_d$ 's demotion. When an anchor node is lost, the demotion protocol allows Phyllo to recover by demoting the lost node and updating the partitions' routing information.

What happens if a node disobeys the promotion or demotion protocol? In either case, the disobedient node can only harm itself. Since a node is distinguished as a major or minor node by its peers, Phyllo does not care how a node sees itself. If a node behaves differently from its major/minor designation, all of that node's messages will be dropped because the sender is not known. In the worst-case scenario, the network loses its partitioned topology and functions as the original overlay.

#### 4. Performance Evaluation

In this section, we analyze the performance of our overlay security framework implementation via simulations.

##### 4.1. Implementation and Experimental Setup

We have implemented the techniques described in Section 3 for Phyllo on the Pastry overlay network. For this implementation, we use the two different kinds of promotion protocols: the original promotion protocol (Section 3.3) and a forced promotion as an extension of the original promotion. In the original promotion, the trust value of a minor node should be higher than a configurable threshold to be promoted to the major partition. In contrast, a minor node may be coercively promoted to the major partition in order to split a minor partition. The forced promotion is invoked when the size of the minor partition becomes extremely large (may cause bottlenecks at the anchor nodes). In this case, our implementation promotes the minor node with the highest trust value and with the ID closest to the center of that partition's ID range.

Several methods support the promotion/demotion mechanisms. Whenever a node receives a message, `evaluate(Message)` is invoked. This method allows the node to inspect the contents of the message to determine whether the message has been correctly routed. For promotion and demotion, the node calls `havePromotions()` and `haveDemotions()`, respectively. If either of these returns true, the corresponding node can be retrieved with `getPromotionNode()` and `getDemotionNode()`. Once a node has been promoted

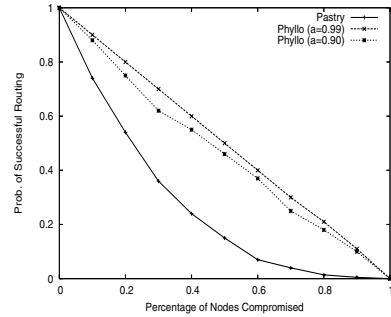


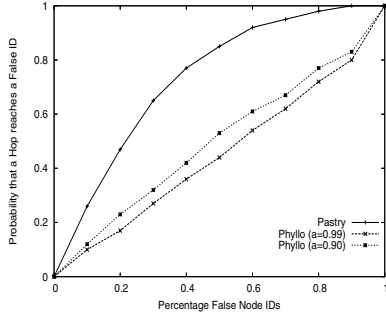
Figure 5. Chances for Successful Overlay Routing

or demoted, `promoted(Node)` and `demoted(Node)` complete the procedures. Note that while Phyllo supports the actual protocols, each overlay network implementation defines the conditions under which a node is promoted or demoted. Finally, `allowJoin(Node)` supports the join process of a new node. A request for join is accepted or rejected based upon some criteria such as partition size or ID distribution.

In our implementation, the minor partition size (including both anchor nodes),  $R$ , is strictly bounded by Pastry's leaf set size,  $L$ . Whenever  $|R| = |L|$ , the forced promotion is performed. Otherwise, Phyllo's configurations are equivalent to those of Pastry: ID base 16 ( $b = 4$ ), leaf set size  $|L| = 16$ , routing table entries  $|M| = 16$ , and ID space,  $2^{128}$ . Our experiments use overlay network sizes from  $N = 100$  to 10,000 and are run on a Dual 2 GHz G5 PowerMac with 1 GB memory and Java 1.4.2.

##### 4.2. Experimental Results

Figure 5 illustrates that Phyllo protects against message forwarding attacks (i.e., compromised nodes do not forward messages) by significantly improving the successful overlay routing ratio. The data show the routing success of 200,000 messages for Pastry and for Phyllo using trust evaluation schemes of  $\alpha = 99\%$ , and  $90\%$  where  $|R| = |L| = 16$ ,  $|M| = 16$  and  $N = 5000$ . All Phyllo data assume that the minimal number of major nodes is used. Here, the successful overlay routing ratio is the ratio of the number of successful routings to the total number of overlay routings. Pastry has a polynomial routing ratio – the addition of compromised nodes decreases the number of possible message routes by a polynomial factor. Phyllo yields an almost linear relationship, decreasing an attacker's effectiveness. Even with a trust evaluation rate of  $\alpha = 90\%$ , Phyllo still provides at least a 10-20% improvement over Pastry against forwarding attacks.

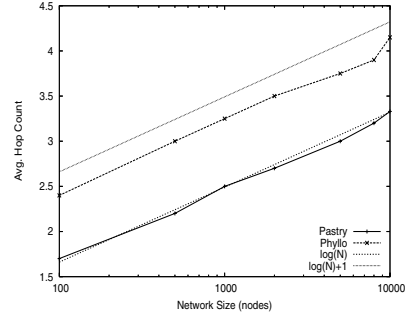


**Figure 6. The Probability that a Message is Routed To a False ID**

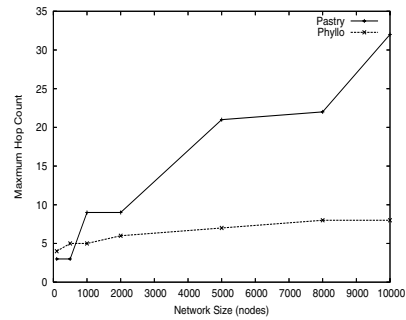
Phyllo also mitigates attacks involving bad routing updates (Figure 6). To compare Phyllo’s and Pastry’s resilience against such attacks, we simulate an attack where the malicious nodes use routing updates to propagate false node IDs throughout the network ( $|R| = |L| = 16$ ,  $|M| = 16$  and  $N = 5000$ ). We assume that a percentage of the total number of IDs are false and the trust evaluation scheme detects these bogus IDs with either 99% or 90% accuracy. 200,000 messages are routed through the network. We track how often a message reaches one of the false IDs. Since this attack resembles the message forwarding attack (except that the compromised nodes have spoofed IDs), the results of these two attacks should be similar. The results show that Phyllo reduces routing attacks by more than a factor of 2 for 20-80% false node IDs.

For network size  $n$ , Pastry can route a message between any two nodes in  $O(\log_{2^b} n)$  hops [10]. Figure 7 validates that Phyllo is upper-bounded by the same expression. (200,000 messages were sent between random nodes on various sizes of Pastry and Phyllo networks.) Since the major partition of Phyllo is inherently a Pastry ring, Phyllo requires at most as many hops as Pastry from one major node to another major node. To route between two different minor partitions, however, additional hops to route via an anchor node are necessary. Phyllo thus upper-bounds the longest path as the same route in Pastry (from the same source to the same destination) plus two extra hops. Our experiments indicate that the average overall hop count in Phyllo is  $O(\log_{2^b} N + 1)$ .

Despite having a larger average hop count, Phyllo actually reduces the maximal hop count significantly compared to Pastry. Figure 8 depicts that the maximum hop count of Phyllo is only 8 hops for a network of 10,000 nodes while the maximum hop count of Pastry is at least 32 hops. We analyze 200,000 messages routed from a random source node to a random destination ID when  $|R| = |L| = 16$ ,  $|M| = 16$ . As network size increases, the maximum hop count of



**Figure 7. Average Hop Counts for Overlay Routing**



**Figure 8. Maximum Number of Routing Hops in Pastry versus Phyllo**

Pastry increases significantly (from 3 to 32) whereas that of Phyllo increases marginally (from 4 to 8). For a network with unreliable connections (a common problem in the Internet), Phyllo provides more reliable connections with the lower maximum routing distance.

## 5. Related Work

Security issues in overlay architectures have been studied in several different contexts [2, 11]. These solutions require trusted servers or significant changes to underlying overlay protocols. In addition, none of the solutions can be added to other peer-to-peer systems transparently.

Several overlay systems employ hierarchical architectures for routing, but not for security purposes [4, 15, 5]. Architectures, such as [4], merge multiple overlays into a single peer-to-peer ring, in which a subset of nodes serve as gateways (similar to our anchor nodes) between different overlays. These overlays, however, are still vulnerable to a variety of malicious attacks.

HIERAS [15] divides a P2P ring into several subsections according to lowest link latency. Routing begins at the low-

est layers of the hierarchy and then proceeds to the higher layers. While this approach reduces network latency, the system is not protected from malicious nodes dropping or manipulating packets. In addition, neither [4] nor [15] features node transition mechanisms between the hierarchies after joining.

Phyllo bears some resemblance to the Jelly hierarchical framework [5]. Jelly divides nodes into hierarchical subsets, similar to our minor partitions, for load balancing and locality. Nodes in each subset can move from one subset to another in order to balance the hierarchy (similar to our promotion/demotion protocols). Jelly, however, does not incorporate any additional security mechanisms.

Node evaluation has also been an important topic in peer-to-peer security. Some schemes [13, 3] require a new node to contribute resources (“paying its due”) prior to active participation. A new node in Phyllo, however, can fully participate in the network without prior resource contribution. Phyllo can also offer the incentive of higher responsibility and marginally better routing performance in exchange of good behavior. Other peer-to-peer overlay trust evaluation algorithms include [1, 6]. Phyllo can adopt such mechanisms, especially for its `evaluate(Message)` method.

Secure Overlay Services (SOS) [7] is a peer-to-peer application that attempts to prevent denial-of-service attacks. Peer-to-Peer Security Layer Framework (P2PSLF) provides encryption methods for secure communication among nodes, access policies, and logging functions for grid networks. P2PSLF assures the confidentiality and integrity of a message; however, P2PSLF does not decrease the chance that a malicious node drops the message.

## 6. Conclusion and Future Work

We have presented a flexible and configurable security framework, *Phyllo*, that can be implemented on any structured overlay network. The framework mitigates attacks from malicious nodes when these nodes attempt to control message routes, corrupt routing information, and bring down overlay networks. Phyllo includes three essential mechanisms: partitioning, routing, and promotion/demotion. We have implemented a basic version of Phyllo on Pastry and performed experiments in a local machine. Our simulations show that our framework lowers the maximum number of routing hops and exhibits more reliable message delivery in the presence of malicious nodes.

We will conduct larger-scale experiments to evaluate the performance of Phyllo in more realistic environments (e.g., PlanetLab [9]). This includes the implementations of Phyllo on other overlays and their experiments. Analyzing the results from these experiments will allow us to evaluate the quality of the major partition and determine the best minor partition size. This will greatly assist in making recommen-

dations for the major partition with mostly trusted nodes. The best minor partition size should help not overburden anchor nodes, while providing small hop counts and reliable routing. More work will be done to allow Phyllo to work with highly dynamic networks. Currently, we cannot effectively recover from a failed anchor node and have trouble handling nodes joining during a promotion or demotion. We will also explore several different evaluation algorithms to detect potentially harmful nodes for demotion.

## References

- [1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proc. of ACM CIKM*, pages 310–317. ACM Press, 2001.
- [2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proc. of USENIX OSDI*. ACM Press, 2002.
- [3] E. Friedman and P. Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, June 2001.
- [4] L. Garcés-Erice, E. Biersack, P. Felber, K. Ross, and G. Urvoy-Keller. Hierarchical Peer-to-peer Systems. In *Proc. of IASTED PDCS*. Springer-Verlag, LNCS, 2003.
- [5] R. Hsiao and S.-D. Wang. Jelly: a dynamic hierarchical P2P overlay network with load balance and locality. In *Proc. of IEEE ICDCS*, pages 534–540, March 2004.
- [6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proc. of ACM WWW*, pages 640–651, 2003.
- [7] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proc. of ACM SIGCOMM*, pages 61–72, 2002.
- [8] S. J. Nielson, S. A. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *Proc. of IPTPS*, February 2005.
- [9] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. of the HotNets-I*, October 2002.
- [10] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of ACM/IFIP/USENIX Middleware*, pages 329–350, Heidelberg, Germany, November 2001.
- [11] E. Sit and R. Morris. Security Consideration for Peer-to-Peer Distributed Hash Tables. In *Proc. of IPTPS*, Cambridge, MA, March 2002.
- [12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable peer-to-peer lookup service for Internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, March 2001.
- [13] M. Waldman and D. Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proc. of ACM CCS*, pages 126–135. ACM Press, 2001.
- [14] D. Wallach. A Survey of Peer-to-Peer Security Issues. In *Proc. of ACM ISSS*, 2002.
- [15] Z. Xu, R. Min, and Y. Hu. HIERAS: a DHT based hierarchical P2P routing algorithm. In *Proc. of IEEE ICPP*, pages 187–194, October 2003.

# Identity Theft Protection in Structured Overlays

Lakshmi Ganesh and Ben Y. Zhao  
Computer Science Department, U. C. Santa Barbara  
{lakshmi, ravenben}@cs.ucsb.edu

## Abstract

*Structured peer-to-peer (P2P) overlays rely on consistent and robust key-based routing to support large-scale network applications such as multicast and global-scale storage. We identify the main attack in these networks as a form of P2P identity theft, where a malicious node in the path of a message claims it is the desired destination node. Attackers can hijack route and lookup requests to forge and destroy data to disrupt applications. We propose a solution where nodes sign proof-of-life certificates for partial node ids and distribute them to randomly chosen proof managers in the network. Source nodes can evade attackers by requesting proofs from multiple proof managers. Analysis and simulation show the approach is effective and imposes storage and communication costs that grow logarithmically with network size.*

## 1 Introduction

Structured peer-to-peer overlays [15, 8, 14] provide scalable and resilient infrastructures for Internet-scale applications. A variety of Internet-scale applications have been built on them, including application-level multicast [18, 10], distributed file systems [7, 9] and distributed query processing [5]. While many projects have studied these overlays, few have examined their security issues [12, 1]. Given their global scale, use of low cost identities, and distribution across independent network domains, we cannot treat the presence of malicious nodes as aberrations, but must expect them as part of normal operations.

The core functionality applications leverage is *key-based routing* (KBR) [2], where all messages with the same destination-key route to the same node consistently across changes in the network. Applications use this mechanism to store and locate data using location-independent names, much like a distributed hash table [2]. Since the overlays use large sparse namespaces (160 bits) to avoid name collision, nodes must deliver each message by choosing a single node closest to the destination key. This is often called the

key's *root* node.

For any route request to key  $K$ , a malicious node on the routing path can hijack the key-based routing primitive by claiming that it is  $K$ 's root node. Since nodes only keep state about  $\log N$  nodes for a  $N$  node network, they must rely on intermediate nodes to determine the key to root mapping. For example, an attacker with ID 12340 can hijack a message destined for node 12345 by claiming it is the only node with prefix 1234. We call this the *Identity Attack*, since the attacker is stealing the identity of the true root node. To attack a file system, several malicious nodes close to a target node can claim they (or their colluding neighbors) are the root nodes for all outgoing read requests, and return arbitrary data in response.

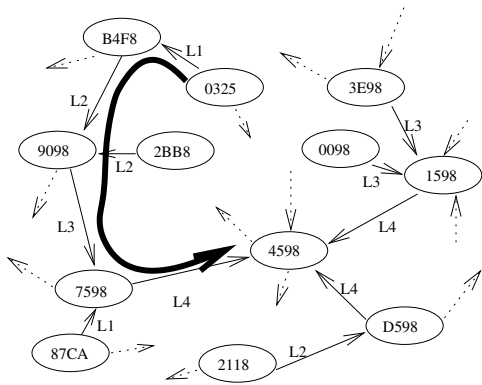
In this paper, we describe the Identity attack, and present a solution where client nodes find self-certifying proofs to verify the existence of their desired destinations. Nodes periodically push signed proofs of their existence out to a random subset of network nodes. Client nodes use their routing table to estimate namespace density and determine when a root node is suspicious. They verify authenticity of root nodes by requesting existence proofs for closer IDs. Our detailed simulations show that namespace density estimation is effective at detecting suspicious nodes, and existence certificates provide proof of an attack while requiring reasonable traffic overhead.

We begin in Section 2 with a discussion of structured overlay security and related work. We then describe the identity attack and our defense in Section 3. Next in Section 4, we explore its efficiency and cost tradeoffs via detailed simulations, followed by conclusions in Section 5.

## 2 Background and Related Work

In this section, we describe structured overlays and key-based routing. We then discuss known attacks on these systems and other work related to this paper.

**Key-based Routing** A structured overlay is an application-level network connecting any number of nodes, each representing an instance of an overlay participant. The nodes are assigned nodeIDs uniformly at random



**Figure 1.** Base 10 prefix routing in Tapestry from 5230 to 8954.

0	0	12021	10	0031	120	021	1230	02	12300	2	123000
1	1	23002	11	0321	121	301	1231	23	12301		123001
2	2	22031	12	3002	122	223	1232		12302		123002
3	3	30110	13	2100	123	002	1233	31	12303		123003

Node 123002

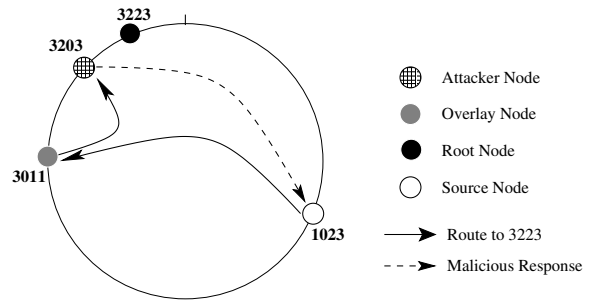
**Figure 2.** Base 4 routing table for node 123002.

from a large identifier space. Application-specific objects are assigned unique identifiers called keys from the same space.

Each key is dynamically mapped by the overlay to a unique live node, called its *root node*. While a key’s root can change with network membership, a single node is responsible for a key in a consistent network at any given time. To deliver a message to its root node (key-based routing), each node forwards messages using a locally maintained routing table of overlay links. Figure 1 shows an example of Tapestry’s prefix routing algorithm, and Figure 2 shows a node’s routing table.

Each system defines a function that maps keys to nodes. For example, keys can be mapped to the live node with the closest `nodeId` as in Pastry [8], or the closest `nodeId` clockwise from the key as in Chord [14].

**P2P Attacks and Defenses** Previous work describes two attacks on structured overlays, the Sybil attack [3] and the Eclipse attack [1]. In the Sybil attack, an attacker generates a large number of identities and uses them together to disrupt normal operation. In the Eclipse attack, attackers try to organize to disproportionately populate routing tables inside target nodes to affect routing operation. Both attacks can increase the probability that a malicious node can intercept a desired route request, resulting in an Identity attack. The Identity attack is more general, however, since it can be launched by a single malicious node, and affects every



**Figure 3.** In a network using digits of base 4, node 1023 routes a message towards key 3223. Before it reaches the root (3223), an attacker intercepts the message and responds as the root.

route, lookup or store operation.

Several approaches limit the Eclipse attack by constraining connectivity in the network [1, 4, 11]. However, these defenses can only limit attackers from attracting more than their share of normal traffic, but cannot protect traffic that routes to malicious nodes. They also require external mechanisms to verify node properties such as in-degree and location in the network. In contrast, our approach requires only key-based routing, and can significantly reduce the impact of malicious nodes in the routing path.

Finally, public-private keypairs based on prefix IDs was also used in the Cashmere [17] anonymous routing system. Network indirection across an overlay is generalized for mobility and resilience in the Internet Indirection Infrastructure (I3) project [13].

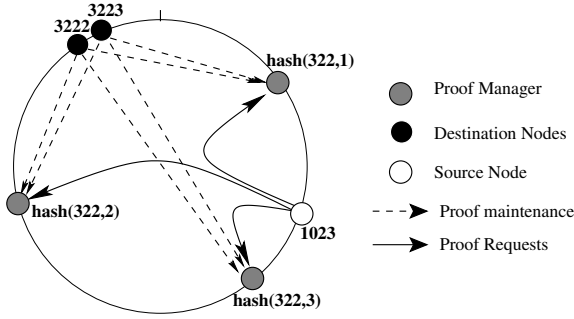
## 3 Defending Against the Identity Attack

### 3.1 The Identity Attack

To perform an Identity Attack, a malicious node hijacks an overlay connection at setup time by spoofing the destination node. When an overlay node routes a message to some key  $K$ , it wants to connect to  $K$ ’s root node (generally the node in the overlay with ID closest to the key). A malicious node on the routing path intercepts the message and responds to the source claiming that it is  $K$ ’s root node.

By claiming to be  $K$ ’s root node, the attacker can intercept application requests and return data of its own choosing. For example, the attacker can hijack a request for a block in a distributed file system and respond with arbitrary data. At worst, she can arbitrarily manipulate application behavior; at best, the application invalidates the data, reducing this to an effective denial-of-service attack. Figure 3 shows an example of the identity attack.

While a single node can perform the attack, malicious



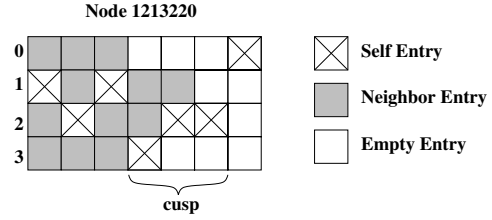
**Figure 4.** Nodes 3222 and 3223 periodically send signed existence proofs for prefix 322 to 3 randomly chosen proof managers in the overlay; node 1023 requests proofs.

parties can increase the effectiveness of the attack by using the Sybil attack [3] to generate a large number of colluding attacker nodes. Attackers can generate numerous identities to perform a *client-based* or *key-based* identity attack. In a client-based attack, multiple malicious nodes collude to fill a target node’s routing table using the Eclipse attack [11]. They can then isolate the node and hijack all outgoing application requests. In a key-based attack, the attacker targets a specific application-level key, and generates identities until it obtains a substantial number of identities close to the target key. Distributed across the network, these nodes will intercept most routing paths to the target, and effectively isolate the real root node (and content) from the network.

### 3.2 Existence Proofs

Our defense uses signed certificates to prove the existence of nodes with IDs in a namespace range. Online nodes periodically sign and send these *existence proofs* to a random subset of nodes, *proof managers*, for storage. When a node responds to a message with key  $K$ , the message source uses a namespace density estimate to determine whether the responder is a likely root. If not, the source node guesses the prefix that  $K$ ’s real root will share with  $K$ , and sends verification requests to the proof managers responsible for that prefix. If a better root exists, it will have signed a recent certificate with which the source can prove the identity theft. See Figure 4 for an example.

In a prefix routing protocol such as Tapestry [15], we use prefixes of different lengths to identify specific ranges of nodeIDs. In a namespace where nodes have IDs of  $L$  digits, we use a prefix of length  $l < L$  to define a *prefix group* corresponding to the set of all nodes whose IDs start with that prefix. Shorter prefixes will have prefix groups with more members. If nodes use  $L$ -digit IDs, each node potentially belongs to  $L$  different prefix groups. This corresponds to



**Figure 5.** Routing table for node 1213220, showing the *cusp* region.

different sized partitions of the namespace whose members are nodes with IDs in the range.

To certify that a prefix-group is non-empty, a member node uses a public key to sign an existence certificate embedded with a nonce. To simplify verification, a central offline CA distributes a unique public-private key pair to all members of each unique prefix group when they join the network. For example, a node ABCD would receive key pairs for prefix groups A, AB, ABC, and ABCD. Existence proofs are signed with the private key for the corresponding prefix. At regular intervals according to a network-wide parameter  $I$ , a node  $N$  signs certificates proving the existence of various prefix groups it belongs to. To determine the proof managers for prefix group  $P$ ,  $N$  applies a SHA-1 hash to  $P$  with several salts (*i.e.*, 1,2,3) to generate several random keys in the namespace. The proof-managers are the root nodes of those keys.

A node initiates verification when it thinks a responder to a message for key  $K$  is suspicious. It examines its local routing table, to find the longest prefix column for which it has all entries filled. This threshold  $T$  is a measure of the density of the network. The responder should match the desired key with at least  $T$  prefix digits. If not, the node tries to verify the existence of nodes matching longer prefixes of key  $K$ . It searches for nodes matching a prefix by calculating its proof managers using the salted hash as described above, and queries them for the relevant certificates. If any queries are successful, it has discovered an attempted identity attack.

### 3.3 Limiting Prefix Groups

Certifying every possible prefix group in the network would be effective, but prohibitively expensive. In this section, we show how to validate the entire network by certifying only a small number of prefixes.

Our goal is to provide existence proofs for all prefixes of a certain length  $L$ . A small  $L$  means many nodes will match the prefix, resulting in large bandwidth and storage overheads for verification. A large  $L$  means the client node may need to verify many prefixes in order to “find” a

suitable root node.

Recall that nodeIDs are chosen uniformly at random using a secure hashing function (e.g. SHA-1). To choose an appropriate prefix length to store proofs for, the client node examines its own routing table to measure the density of nodes in the namespace. It chooses several prefix lengths for which its routing table contains columns with a mixture of empty and nonempty entries. We call this region the *cusps*. Figure 5 shows a node’s routing table along with its cusp region. Choosing a prefix length  $L$  in this region means that each prefix of length  $L$  is likely to match a “small” number of nodes in the system.

Prior work [16] proved that with a high probability, such a cusp will have a size  $\leq 2$ , independent of network size. The proof is an application of the Coupon Collector problem, where entries in a row of the routing table are coupons, and collecting coupons is the act of assigning random IDs to fill a particular entry. The result says the probability of the cusp including more than 2 routing levels is  $P \leq b/e^b$ , where  $b$  is the base of the prefix digit.  $P$  is less than 0.07 for  $b = 4$  and less than  $1.8 \times 10^{-6}$  for  $b = 16$ .

As a result, we assume a cusp size of 3. A client node searches its routing table, and marks the start of the cusp as the first routing level that contains an empty entry. For example, node 1213220 in Figure 5 uses prefixes of length 4, 5, and 6 when sending certificates and requesting verifications. Verifications start from the longest possible prefix and work downwards. To test for the existence of a node 12301230, our node would first test prefix 123012, then 12301 if necessary, and finally 1230.

### 3.4 Replicating Proof Managers

Several factors can affect the success rate of verification requests. Node churn can limit the availability of proof managers for a given prefix. Malicious nodes on the path between the client and proof managers can hijack and drop the verification request. Finally, if proof managers themselves are compromised, they can simply deny ever seeing the requested existence certificate.

We can improve the verification success rate by increasing the number of randomly chosen proof managers. A larger replication factor means more managers will be online despite network churn and more of them will be non-malicious. Verification requests will take a larger number of random routes, increasing the number of requests that will avoid malicious nodes. We evaluate the impact of these factors on verification in Section 4.

### 3.5 Extension to Other Protocols

While much of our discussion assumes the use of prefix routing, our technique easily generalizes to other protocols.

Topology	Random
Length of run	7200s
Base	16
Prefixes certified (cusp size)	3
Certification interval	500s
Certificate time-out period	1500s

**Table 1. Simulation Settings**

For example, a range-based routing protocol like Chord simply routes “towards” a given value. Instead of certifying existence of nodes matching a given prefix 0123, we would certify existence of nodes in a certain value range (e.g. 12300-12399). Similarly, our mechanism for choosing prefix lengths to certify reduces to finding several range sizes that have the right node density. We are studying these mechanisms in ongoing work.

## 4 System Evaluation

In this section we describe some preliminary results based on detailed simulations on the P2PSim simulation platform. P2PSim [6] is a multithreaded discrete event simulator with full implementations of several protocols. Our experiments are run using the implementation of Tapestry [15] included with P2PSim. The simulation settings are listed in Table 1.

### 4.1 Overhead of Existence Proofs

In order to defend against Identity Attacks, each node in the network incurs bandwidth overhead in sending existence proofs (certificates) to proof managers, and storage overhead in storing existence proofs for other prefixes. We use simple analysis to quantify these overheads.

**Bandwidth Cost.** Nodes certify their prefix groups every  $T$  seconds ( $T = 500s$  in our simulations). Let  $v$  denote the number of prefix groups each node certifies ( $v = 3$  in our simulations), and  $r$  denote the replication factor (number of proof managers per prefix). Thus the rate that each node sends out certificates is  $\frac{vr}{T}$ . With our simulation parameters, we expect each node to send  $\frac{3 \cdot 4}{500} = 0.024$  certificates/second. This is confirmed by our measurements that show the rate to be 0.025 certificates per second for all network sizes. As expected, this overhead increases linearly with replication factor. Certificates should be no larger than 50 Bytes, resulting in bandwidth cost of 1.25 Bytes/second or 10 bps.

**Storage Cost.** If we assume that a new certificate replaces previous certificates from the same host, then each node generates  $vr$  certificates to be stored. Assuming the hash function to generate proof manager IDs spreads the load

evenly across all nodes, each node only needs to store  $vr$  certificates, for a total cost of 600 Bytes of storage per node using our parameters. Clearly, the overhead from generating and storing certificates is small enough to not impact overall system performance.

## 4.2 Resilience Against the Identity Attack

We evaluate our defense against the Identity Attack under a variety of conditions. We first consider the basic identity attack in a stable network where all mechanisms function normally. Next, we consider the case when compromised proof managers deny the presence of certificates. This models colluding malicious attackers, where colluding attackers cover each others' tracks by refusing to service verifications. Another factor is certificate hijacks, where malicious nodes intercept certificates on the path between the signer and a proof manager. This is a stronger attack, where we assume in-path routers has read the message payload to determine if it is a certificate (*i.e.* no overlay link level encryption). This attack also applies if malicious nodes indiscriminately hijack all messages they see without interpretation. Finally, we examine the impact of nodes entering and leaving the network (node churn).

**Performance Metrics** To quantify the effectiveness of our defense, we examine two metrics, the *trigger rate*, how often does an attack trigger a verification request, and the *verification rate*, how often do requests succeed in locating an existence proof proving the attack. Our simulations show that we get a trigger rate of 100% using our threshold detection scheme, with roughly 3 verifications requested per actual attack. We are currently tuning our threshold to reduce the verification overhead. Since the trigger rate is 100%, our experiments measure the verification rate under different conditions.

**Ideal Conditions** We assume that malicious nodes hijack non-certificate messages, but do not hijack certificates, and compromised proof managers perform normally. We vary the network size and assume no node churn. Figure 6 shows that for all network sizes, our trigger rate is 100%, and verification rate is over 99.8%.

**Verification Denials** Here we assume malicious nodes hijack messages and deny verification requests, but do not hijack certificates. We simulate a network of 4K nodes while varying the percentage of malicious nodes. Figure 7 shows that the verification rate falls as the proportion of malicious nodes increases. For a given proportion of malicious nodes, however, performance improves if we increase the number of proof managers. This improvement is significant when a large number of nodes is malicious. Note that even when 90% of nodes are malicious, over 80% of attacks are caught with just two proof managers.

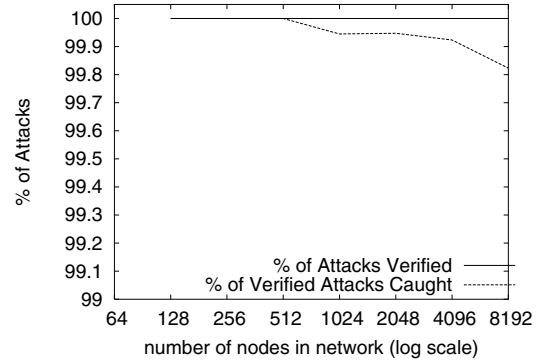


Figure 6. Effectiveness of the verification system under ideal conditions (no denials, no certificate hijacks, no node churn).

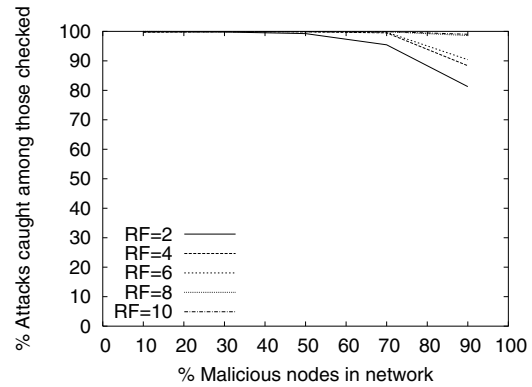
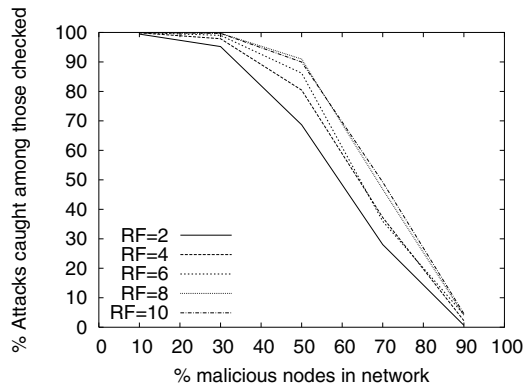


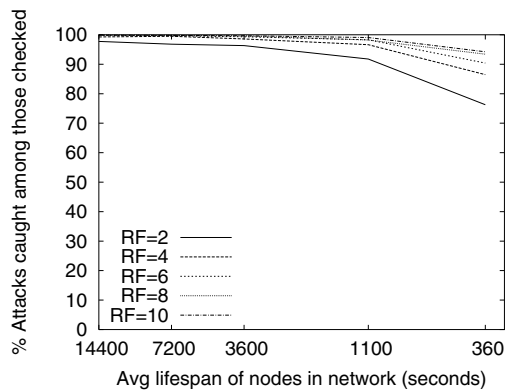
Figure 7. Use of replication factor to increase verification effectiveness (certificate denials, no hijacks, no node churn).

**Certificate Hijacks** This model is similar to the last model, with the addition that malicious nodes also hijack certificates en route to proof managers. Our results in Figure 8 show that this additional factor causes a steep fall in performance. For a given % of malicious nodes in the network, certificate hijacks have a much stronger impact than verification denials. This is because the likelihood that a malicious node is in the path of a certification is higher than the likelihood that it is a proof manager. For predominantly malicious (over 50%) networks, the percentage of attacks caught is very poor, even when many proof managers are used. But for networks with up to 40% malicious nodes, nearly 80% of the attacks are caught.

**Churn** Finally, we examine the effect of adding churn to the network. We simulate an attack model that includes Identity Attacks, verification denials as well as certificate



**Figure 8. Use of replication factor to increase verification effectiveness (certificate denials and hijacks, no node churn).**



**Figure 9. Use of replication factor to increase verification effectiveness (certificate denials, hijacks, and node churn assuming 20% malicious nodes).**

hijacks, and ran it on networks of 4096 nodes, 20% of which are malicious. Figure 9 shows that increasing churn degrades performance. With 8 or more proof managers, it is highly likely that at least one non-malicious proof manager can service each verification request; hence over 95% of the attempted hijacks are caught.

## 5 Conclusion

In this paper, we described the Identity Attack, a simple attack that subverts the fundamental key-based routing functionality in structured peer-to-peer overlays. Unlike the Sybil and Eclipse attacks, the Identity attack directly impacts application level behavior, and can leverage both prior attacks for increased effectiveness. We propose a defense

that uses the placement of signed existence proofs at randomized node subsets. After routing requests, source nodes use estimates of namespace density to trigger verification, where they determine whether “better” root nodes exist by searching for their existence proofs.

## References

- [1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *Proc. of OSDI*, Dec 2002.
- [2] F. Dabek, B. Zhao, P. Druschel, J. Kubiawicz, and I. Stoica. Towards a common API for structured P2P overlays. In *Proc. of IPTPS*, Feb 2003.
- [3] J. R. Douceur. The Sybil attack. In *Proc. of IPTPS*, Mar 2002.
- [4] K. Hildrum and J. Kubiawicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proc. of DISC*, Oct 2003.
- [5] R. Huebsch et al. Querying the internet with pier. In *Proc. of VLDB*, Berlin, Germany, Sept. 2003.
- [6] p2psim, a simulator for peer-to-peer protocols. <http://pdos.csail.mit.edu/p2psim/>.
- [7] S. Rhea et al. Pond: The OceanStore prototype. In *Proc. of FAST*, San Francisco, Apr 2003.
- [8] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, Nov 2001.
- [9] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. of SOSP*. ACM, Oct 2001.
- [10] A. Rowstron, A.-M. Kermarrec, P. Druschel, and M. Castro. SCRIBE: The design of a large-scale event notification infrastructure. In *Proc. of NGC*. ACM, Nov 2001.
- [11] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. In *Proc. of ACM SIGOPS European Workshop*. ACM, Sept. 2004.
- [12] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proc. of IPTPS*, Mar 2002.
- [13] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. of SIGCOMM*, Aug 2002.
- [14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM*, Aug 2001.
- [15] B. Y. Zhao et al. Tapestry: A global-scale overlay for rapid service deployment. *IEEE JSAC*, 22(1), Jan 2004.
- [16] B. Y. Zhao, J. D. Kubiawicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report CSD-01-1141, UCB, Apr 2001.
- [17] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proc. of NSDI*, Boston, MA, May 2005. ACM/USENIX.
- [18] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiawicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of NOSSDAV*. ACM, June 2001.

# An Effective Intrusion Detection Approach for OLSR MANET Protocol

M. Wang, L. Lamont,  
Communications Research Centre  
louise.lamont@crc.ca  
maoyu.wang@crc.ca

P. Mason, M. Gorlatova,  
DRDC  
Peter.Mason@drdc-rddc.gc.ca  
Maria.Gorlatova@drdc-rddc.gc.ca

## Abstract

*The Optimized Link State Routing (OLSR) protocol is a proactive Mobile Ad hoc Network (MANET) routing protocol. Security aspects have not been designed into the OLSR protocol and therefore make it vulnerable to various kinds of attacks. Recent research efforts have focused on providing authentication and encryption techniques to secure the OLSR protocol against attacks from outside intruders. A second line of defense is required to provide intrusion detection and response techniques in protecting the OLSR protocol against attacks from inside intruders.*

*In this paper, we describe security threats to the OLSR MANET routing protocol and present an intrusion detection solution based on protocol semantics checking. Our approach is based on semantic properties that are implied in the protocol definition and specify the correct OLSR routing update behavior. Conflict checking based on semantic properties is applied in every MANET node. Any abnormal protocol semantics will trigger an intrusion alarm. While we use OLSR as an example, we argue that the presented approach can be applied to any Multi-Point Relay (MPR) proactive MANET protocol.*

## 1. Introduction

Mobile Ad hoc Networks (MANETs) are autonomous systems of mobile nodes interconnected by wireless links. Any node in a MANET acts as a router to support connectivity to other mobile nodes that are out of range. The nodes are free to move randomly and organize themselves arbitrarily. The inherent flexibility offered by these networks, originally conceived for mostly military purposes such as battlefield communication and battlefield sensor monitoring network, allows for ease of deployment and appeals to various commercial applications such as convention meetings, electronic classrooms, search-and-rescue efforts, disaster relief, and law enforcement. A side effect of this flexibility is the ease with which a node can join or leave a MANET. Lack of any fixed physical and, sometimes, administrative infrastructure in these networks makes the task of securing them extremely challenging.

The Internet Engineering Task Force's (IETF) MANET Working Group (WG) identifies Optimized

Link State Routing (OLSR) [7] as one of four base routing protocols for use in MANETs.

Recent research efforts [3][4] have focused on providing schemes to block unauthorized users from joining an OLSR MANET and to prevent authorized nodes from compromising the security of the MANET. Our previous work [4] proposed an efficient approach to integrate a fully distributed certification authority (CA) in OLSR. A public-key infrastructure (PKI) was tightly coupled with an OLSR MANET at the network layer. In [3], the approach was based on authentication checks of the OLSR control messages. Signatures were introduced as a separate type of control message and served to validate the origin and integrity of information.

However, even when it is protected by the aforementioned authentication schemes, a MANET still needs intrusion detection. First of all, authentication techniques can prevent outsider attacks but cannot deal with insider attacks. Secondly, MANETs' features determine that the complexity and performance penalty of key distribution and management are a heavy burden in MANET environment. Thirdly, intrusion detection plays an essential role in detecting faults when the defensive line of prevention fails or unexpected faults occur [1]. To the best of our knowledge, all network-monitoring systems proposed in [5][6] are for reactive MANET protocols. No monitoring systems are yet provided for proactive MANET protocols such as OLSR.

In this paper, we present an intrusion detection system to prevent authorized nodes from compromising the security of proactive MANET routing protocols for particular types of routing attacks. While we use OLSR as an example, the same concepts can be applied to any MPR based proactive routing protocols for MANETs. As mentioned above, proactive MPR-based MANET routing protocols include OLSR and Topology Broadcast based on Reverse-Path Forwarding (TBRPF) [2]. Recent efforts are underway to extend the existing Open Shortest Path First (OSPF) routing protocol to support an MPR-based MANET [8].

Our approach is based on semantics properties-checking techniques [1] and involves a set of protocol-related properties for specifying correct OLSR behavior. A real test bed has been constructed, in which the existing implementation of CRCOLSRv6 is used to validate our approach. The paper provides valuable insight by detailing the OLSR vulnerabilities and intrinsic properties of OLSR message.

The rest of the paper is organized as follows. Section 2 provides a brief summary of OLSR. Section 3 analyzes vulnerabilities of the OLSR routing protocol and identifies various security attacks that can impact the integrity of OLSR routing. In Section 4, four semantics properties existing in the OLSR routing protocol are abstracted and proposed as criteria to be deployed in every MANET node for the purpose of checking for conflicts. Section 5 presents a case study. Finally, Section 6 concludes that conflict checking based on the properties of semantics in the protocol effectively enhances the security level and fault detection capability of an OLSR MANET.

## 2. The OLSR Protocol

The OLSR protocol is a variation of the pure Link State Routing (LSR) protocol and is designed specifically for MANETs. The OLSR protocol achieves optimization over LSR through the use of MPRs that are selected and designated by neighboring nodes. Unlike LSR, where every node declares its links, only MPR nodes declare links in OLSR. Also, unlike LSR, where each node forwards messages for their neighbors, the behavior in OLSR is as follows: only MPR nodes forward messages for those neighbor nodes that selected them as an MPR node.

Each node selects its MPR set of nodes in a way that, through them, it can reach all of its two-hop neighbors. A node learns about its one-hop and two-hop neighbors from its one-hop neighbors' HELLO messages. By exchanging HELLO messages, a node finds out which neighbors have chosen it as an MPR. The neighbors that select a node as an MPR form that node's MPR selector set. A Topology Control (TC) message is sent to the whole MANET periodically by each MPR in the network to respectively declare its MPR selector set and is used in the construction of routing tables in every MANET node.

In comparison to LSR, the optimization in OLSR localizes parts of routing selection; while computing the routing table to a destination, only MPR nodes are considered as the last hops to the destination instead of all neighbor nodes to the destination; each node picks its own MPRs instead of making a routing calculation to decide which neighbors serve as the last hop to the destination. This localization increases

the threat to the security of the network routing but also provides advantages for local control message validation because of the intrinsic relationship between the HELLO message and the TC message, which will be introduced with Property 1 in Section 4. The localization provides distributed intrusion detection locally.

## 3. OLSR Vulnerabilities

MANET application environments such as the battlefield or law enforcement situations are exposed to more threats than other environments such as electronic classrooms. A MANET node may be easily compromised if captured in the battlefield. Due to the open medium environment in a MANET, an intruder can join in the routing process without any attaching point. Dynamic membership and topology due to mobility are also big holes for security. To favor mobility and lower bandwidth, as well as lower computing power in the MANET, the OLSR protocol design achieves optimization, as mentioned in Section 2, over LSR through the use of MPR nodes. However, it also loses a security advantage that LSR has the fight-back feature. This feature makes LSR robust to malicious attacks and more promising for detecting faults [1]; each router floods its local connectivity to every other router, and each router receives the connectivity information from all other nodes and has the complete topology information for the whole network. In OLSR, only the connectivity used as routing computation (the connectivity to those MPR selectors) is flooded in the whole network. The fight-back feature is lost in OLSR because of the optimization required to tailor to a MANET environment.

As the existing IETF's RFC on OLSR [1] does not specify validation procedures and security mechanisms, numerous opportunities exist for intruding OLSR nodes to launch attacks. Malicious attacks to routing protocols can be cataloged as invalid update messages and router overload (Denial Of Service). DOS attacks are not strictly a routing protocol issue [1]. This paper focuses on invalid update messages in an OLSR MANET. Neighbor nodes know each other by exchanging HELLO messages, which reflect the local connectivity and are used to select the MPRs for routing connectivity. The TC message is different from the Link-State Advertisements (LSA) message because it propagates only partial local connectivity of MPR selectors instead of complete local connectivity of all neighbors. In order to support routing in MANET, a MANET node normally takes two responsibilities: generating control messages and forwarding control messages. To compromise the integrity of the routing protocol, an ac-

tive attacker can send incorrect control packets while the MANET node is generating control messages, or alter control packets while the MANET node is forwarding control messages. Furthermore, there are two ways to tamper with a control message: change its identity (identity spoofing) or damage its contents (link spoofing). [3] The following paragraphs catalog various attacks and describe them in detail.

A. Attacks to control message during generation

a) **Identity spoofing to HELLO message**

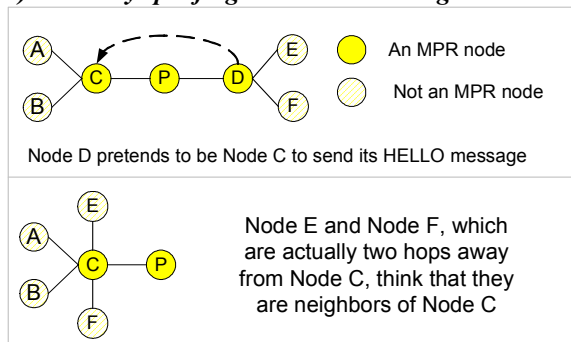


Figure 1. Node D spoofs identity of Node C in its HELLO message

An intruder pretends to be a well-behaved MANET node and spoofs the MANET node’s IP address as the source address to send HELLO messages in the intruder’s neighborhood. Figure 1 shows how the topology’s image of a MANET is distorted when Node D impersonates Node C. Node E and F, select Node C as their MPR. Traffic to Node E or Node F will be sent to Node C, which is believed to be the last hop to Node E or Node F, then lost at Node C because Node C does not have Node E or Node F in its neighborhood.

b) **Link spoofing to the HELLO message**

The damage to the HELLO message includes tampering with the content of the message such as inserting nonexistent neighbors, deleting existing neighbors, and modifying the status of the neighbors. Inserting non-existing neighbors can affect the MPR selection in the neighborhood of the intruder and increase the possibility that an intruder is selected as an MPR. As an MPR, an intruder can manipulate the traffic later on. Deleting existing neighbors makes those neighbor nodes unreachable. Modifying the status of neighbors achieves a similar effect to inserting or deleting a neighbor.

c) **Identity spoofing of a TC message**

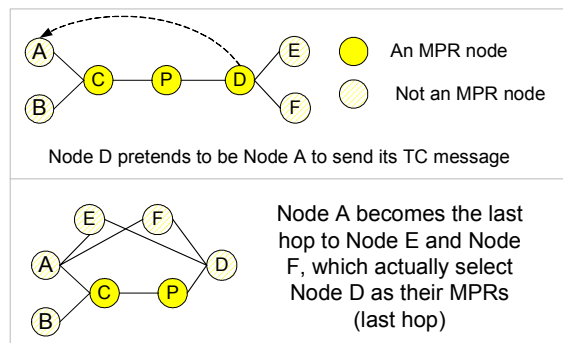


Figure 2. Node D spoofs identity of Node A in its TC message

An intruder pretends to be a well-behaved MANET node and spoofs this MANET node’s IP address as the source address for propagating the TC message in the MANET. As Figure 2 shows, Node D is selected as the MPR by Node E and Node F, and Node D is supposed to send a TC message to propagate that Node D is the last hop to Node E and F. Instead of sending a correct TC message, Node D spoofs Node A’s IP address in its TC message. The attack results in a distorted topology as shown in Figure 2. Traffic to Node E or F is delivered to Node A.

d) **Link spoofing in the TC message.**

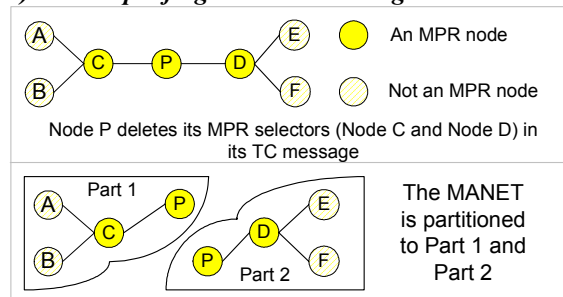


Figure 3. Node P deletes its MPR selectors ( Node C and Node D)

Link spoofing in the TC message includes inserting non-existent MPR selectors and deleting existing MPR selectors. Deleting existing MPR selectors results in those MPR selectors being unreachable as well as some nodes are left with connectivity to only those MPR selectors, as shown in Figure 3. Inserting non-existent MPR selectors creates fake links to those MPR selectors from the node that originated the TC message, which is similar to identity spoofing in the TC message, and distorts network topology. Distortion of network topology can cause routing loops or generate conflicts during routing table calculation.

A. Attacks to control message while forwarding

TC messages carry topology information critical to the routing calculation and need to be flooded in the MANET through the MPR nodes. While a TC message is being forwarded in the MANET, those relay nodes can tamper with the TC message by inserting or deleting MPR selectors in the TC set.

Since TC messages are flooded across the whole network, the intrusion can happen at the point of origin or at the point of forwarding. Damage to the TC messages can result in a much more serious problem than the one caused by damaging the HELLO messages: the TC messages are used globally by all of the nodes on the MANET for routing calculation. For example, an active attacker may simply send TC messages claiming to be the MPR for nodes where it is not the MPR. As the network depends on the MPRs for routing services, an intruder that manages to become an MPR can easily launch a black-hole attack on the network.

#### 4. Intrinsic properties of OLSR messages

The HELLO message carries the information of all of a node's one-hop neighbors. By exchanging HELLO messages in this neighborhood, a node acquires its local topology up to a two-hop range. The TC messages are propagated in the whole MANET, which ensures that all MANET nodes have the same view of the network routing topology; each MANET node calculates its routing table. The redundancy between the HELLO message and the TC message implies the following semantic properties and provides an opportunity to check for conflicting information and protect the OLSR routing protocol.

In the following description, a "HELLO set" is defined as all of the one-hop neighbor nodes carried in an HELLO message; a "TC set" is defined as all MPR selectors carried in a TC message.

**Property 1:** An HELLO message originating from a MANET node contains all one-hop neighbors of the node. A TC message originating from the same MANET node contains only MPRs selectors of the node. The TC set is always a subset of the hello set to the same MANET node. If  $TC_p$  represents the TC set of MANET node P,  $Hello_p$  represents the hello set of MANET node P; then the property  $TC_p \subseteq Hello_p$  exists.

A simple example of a MANET is shown in Figure 4. Nodes A, B, C, and D are all one-hop neighbors of Node P. The HELLO set sent from Node P includes neighbors: A, B, C, and D. Node C and Node D are not adjacent and choose Node P as a relay. The TC set originated from Node P contains

Node C and D that select node P as their MPRs. Property 1 indicates that TC set ( $TC_p$ ) is always a subset of the HELLO set ( $Hello_p$ ).

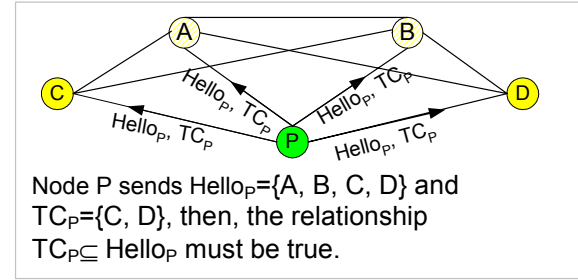


Figure 4. Relationship between  $TC_p$  and  $Hello_p$

If Property 1 is violated, it indicates that the originator of the TC message has inserted extra MPR selectors in the message and is pretending to be an MPR node for more nodes than it should be, which is described as link spoofing in the TC message in Section 3. An intruder launching identity spoofing in the TC message will violate the property  $TC_p \subseteq Hello_p$  and be detected by semantics checking with Property 1.

**Property 2:** If a MANET node receives a TC message that lists itself as an MPR selector, the originator of the TC message must be in the neighborhood of this MANET node.  $TC_p$  represents the TC set originating from Node P, and  $Hello_C$  represents the HELLO set of Node C; If  $C \in TC_p$ , then  $P \in Hello_C$  must be true first.

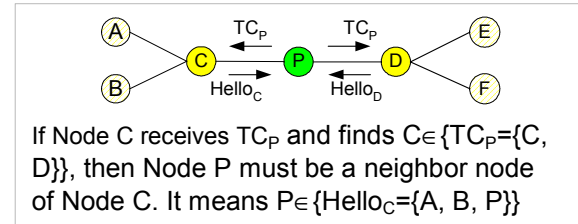


Figure 5: Relationship between  $TC_p$  and  $Hello_c$  or  $Hello_d$

Shown in Figure 5, Node C is the one-hop neighbor of Nodes A, B, and P. Node D is the one-hop neighbor of Nodes E, F, and P. Node C and Node D are not adjacent and select Node P as a relay. The TC set originating from Node P contains Node C and D: both select Node P as their MPR. Property 2 indicates that when the TC set from Node P is received, Node C or Node D finds itself included in the TC set and can verify that Node P must be in the neighborhood of Node C or Node D.

If Property 2 is violated, it may imply several attacks. The originator of the TC message is pretending to be an MPR --- the link spoofing of the TC message. An impersonation attack is being launched by

the originator --- the identity spoofing of the TC message. It is also possible that other intruders on the flooding path tamper with the TC message. Checking of semantics with Property 2 can detect these attacks. Further investigation is needed to probe if the attack is caused by the originator or by the forwarding nodes. Combining the checking results from Property 1, 2, and 3 can identify the types of the attack and determine who the intruder is.

**Property 3:** If a MANET node receives a TC message originating from its neighbors and notices that the TC message lists itself as an MPR selector, this MANET node must have listed the TC originator as an MPR in its HELLO message first. Node P and Node Q are neighbors;  $T_p$  represents the TC set originating from Node P;  $M_q$  represents the MPR set in Node Q; if  $q \in T_p$ , then  $p \in M_q$  must be true first.

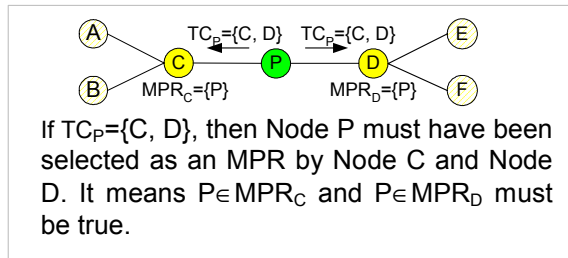


Figure 6. Relationship between  $TC_p$  and  $MPR_C$  or  $MPR_D$

In Figure 6, when the TC set originating from Node P is propagated in the MANET, Node C or Node D reveals that it is in the TC set. Property 3 indicates that Node C and Node D can verify that they must select Node P as their MPR and propagate Node P as the MPR in their HELLO message.

If Property 3 is violated, the originator of the TC message is pretending to be an MPR node for some of its neighbor nodes and tries to attract traffic destined to those MANET nodes in the future, which is typical link spoofing of a TC message. Semantic checking with Property 3 can detect the link spoofing to neighbor nodes of the intruder.

**Property 4:** The sender of a TC message will hear the same TC messages that are forwarded by all its MPRs. The MPR nodes only change the source IP address in the IP header and keep the IP payload; the TC message is unchanged.  $T_p$  represents the TC set sent from Node P;  $M_1, M_2, \dots, M_i, \dots, M_n$  represent all the MPR nodes of Node P;  $TP_{m_1}, TP_{m_2}, \dots, TP_{m_i}, \dots, TP_{m_n}$  represent the TC sets forwarded by  $M_1, M_2, \dots, M_i, \dots, M_n$  for Node P, then the property  $TP_{m_i} \neq \emptyset$ , for all  $i = 1, 2, \dots, n$ , and,  $TP_{m_1} = TP_{m_2} = \dots = TP_{m_i} = \dots = TP_{m_n} = T_p$  exists.

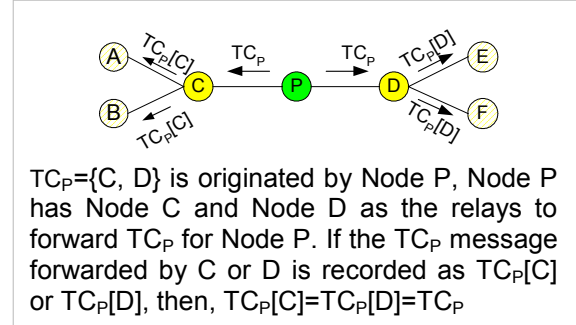


Figure 7. Relationship between  $TC_p$  and  $TC_p[C]$  or  $TC_p[D]$

As shown in Figure 7, Node P selects Node C and Node D as a relay to propagate a message to Node A, B, E and F. The TC set originating from Node P is recorded as  $TC_p$ , the TC message forwarded by Node C or Node D is recorded as  $TC_p[C]$ ,  $TC_p[D]$ . Property 4 indicates that Node P will monitor and ensure that the  $TC_p$  set forwarded by Node C and Node D and  $TC_p[C]$  or  $TC_p[D]$  is the same as the original  $TC_p$ .

If Node C or Node D is compromised and tries to modify the TC set originating from Node P, Node P will detect the attempt because Property 4 is violated. Semantics checking with Property 4 can detect attacks on the control message while forwarding.

A component, checking all aforementioned conflicts, is deployed in every MANET node that is running the OLSR routing protocol for the purpose of detecting any illegitimate update to the control message. A MANET node receiving a TC message applies the properties to validate that the TC message is a legitimate update message. A MANET node sending a TC message also applies Property 4 to detect if there is any anomalous activity while the routing information is exchanged. The checking of conflict information based on the properties of semantics in the protocol enhances the security level and also increases the robustness of the routing operation.

## 5. A case study

This case study provides a concrete example of how a compromised MANET node can easily trick other legitimate MANET nodes; it also shows how the conflict checking system can detect the illegitimate MANET node and its attack effectively.

Figure 8 illustrates 11 nodes across 5 hops in a real test bed. A compromised node can add non-existent neighbors to its HELLO or TC message; can change the messages it relays. A conflict checking mechanism, which checks the integrity of TC messages and sends warning messages if TCs are compromised, is implemented to verify its effectiveness.

MANET nodes will react to the warning messages by excluding the compromised nodes during their routing table calculation.

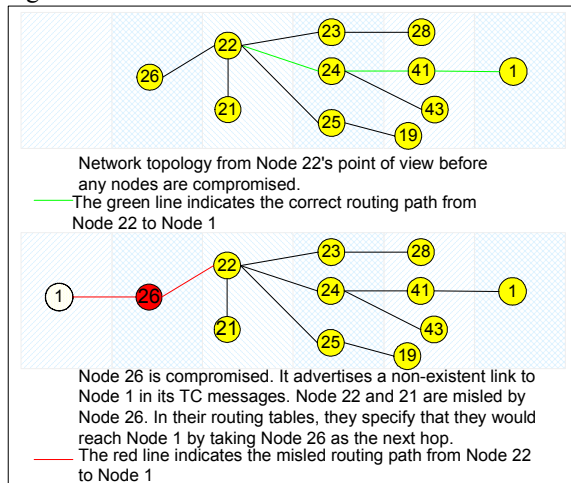


Figure 8. Illustration of link spoofing in TC message in a testbed

In the study, Node 26, is compromised, and starts to advertise a non-existent link to Node 1 in its TC messages. As a result, both Nodes 21 and 22 choose Node 26 as the next hop to reach Node 1 because it appears to be part of a shorter two-hop link. A compromised Node 26 successfully convinces surrounding nodes to incorrectly change their routing tables.

Destination	Next Hop	Hop Count
1	26	2
26	26	1
41	24	2

Under attack with Node 26 compromised and no defense mechanism

Destination	Next Hop	Hop Count
1	24	3
26	26	1
41	24	2

Under attack with Node 26 compromised and conflict checking is enabled

Figure 9. Routing table of Node 22

In the second test, we enable the conflict checking. When Node 1 receives the TC message originated from Node 26, it observes that Node 26 is not in its neighborhood. Node 1 then sends warning messages to other nodes, indicating that the TC messages coming from Node 26 are to be ignored. As soon as receiving the warning messages, MANET nodes ignore the TC messages coming from node 26, and recalculate their routing tables to the correct state. Figure 9 compares the routing table of Node 22 when it is under attack without and with the conflict checking mechanism.

## 6. Conclusions

A simple and effective approach to detect illegitimate routing control messages in OLSR MANET is proposed in this paper. Our solution for MANET security favors effectiveness, simplicity, low computing overhead and transmission overhead. The intrusion detection through conflict checking can also be generalized to protect other types of proactive MANET routing protocols that are based on MPR optimization. The case study shows that the approach effectively enhances the security level and fault detection capability of an OLSR MANET.

Future extensions of this work will include a complete implementation of all conflict checking and overall performance evaluation after the complete implementation is accomplished. Investigation to attackers' collaboration and convergence of counter-mechanism are necessary in the future. Cooperation, between MANET nodes for intrusion detection, and the verification procedure for intruders, both merit further research to extend the work.

## References

- [1] Dan Pei, Lixia Zhang, Dan Massey, "A Framework for Resilient Internet Routing Protocols", IEEE Network special issue on Protection, Restoration, and Disaster Recovery, April, 2004
- [2] R. Ogier, F. Templin and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", <http://www.ietf.org/internet-drafts/draft-ietf-manet-tbrpf-10.txt>, July 2003.
- [3] Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Muhlethaler, Daniele Raffo, "Securing the OLSR protocol", Proceedings of IFIP Med-Hoc-Ned 2003, June 25-27
- [4] D. Dhillon, T.S. Randhawa, M. Wang and L. Lamont, "Implementing a Fully Distributed Certificate Authority in an OLSR MANET", IEEE WCNC2004, 21-25 March 2004, Atlanta, Georgia USA
- [5] Y.-C. Hu, A. Perrig and D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks", MobiCom'02, September 2002.
- [6] M. Zapata, "Secure Ad Hoc On Demand Distance Vector Routing", ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 6, Issue. 3, June 2002.
- [7] Thomas Clausen et. al. , "Optimized Link State Routing Protocol", <http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-11.txt>, July 2003.
- [8] M. Chandra, "Extensions to OSPF to Support Mobile Ad Hoc Networking", <http://ietfreport.isoc.org/all-ids/draft-chandra-ospf-manet-ext-02.txt>

# The Attackers' Potential Influence on the Tactical Assessments Produced by Standard Alert Correlation Systems

Stephen W. Neville, *Member, IEEE*

**Abstract**—This work shows that knowledgeable attackers can influence the tactical assessments output by INFOSEC alert correlation systems solely through manipulating the timing characteristics of their attacks. The approach taken is to assume that the defender's goal is to thwart attackers by enact optimal tactical responses. It is then shown that, even in an idealized environment, the defender has no guarantee that the correlation system's estimates of the enacted attacks are correct. A theoretical path always exists by which the attacker can influence the contents of the correlation system's low-level alert clusters. As these low-level clusters form the basis of all higher level analyses, this is sufficient to show that the attacker has influence over the tactical assessments reported by correlation systems. In essence, the attackers can cause the defender to mis-correlate an attack's generated INFOSEC alerts in a manner which will go undetected and is to the attacker's advantage. This capability is shown to hinge on there being attacks whose identification requires the analysis of shared alerts (*i.e.*, alerts generatable by two or more distinct attacks).

**Index Terms**—Alert correlation, novel attacks, timing characteristics, formal models

## I. INTRODUCTION

As network security concerns have grown, there has been a commensurate growth in the R&D of security sensors and systems. This has resulted in a growing market of network and host-based security sensors. Typically, the available sensors are low-level devices which when deployed on large-scale heavily used networks produce voluminous alert streams, due, in part, to false alarm rates which can exceed 90%. Managing alarm volumes through sensor tuning is insufficient as the high data rates of modern networks leads directly to the base rate fallacy argument[1]. These voluminous alert streams compromise security since they overwhelm the human operators tasked with their analyses. Alert correlation systems have been developed to address this issue. Specifically, these systems automatically perform the low-level alert clustering and correlation analyses required to group alerts by cause. After further higher level processing, these groupings (or clusters) are then reported, in the form of prioritized intrusion reports, to the human security analysts tasked with enacting responses.

For corporations and governments, these low-level INFOSEC sensors acts as the primary sources of cyber-security situation awareness information. An organization's near-

term security rests on its ability to accurately detect, identify, assess, and respond to attacks as they are being enacted (*i.e.*, a good tactical defense). Strategic defense, implement through patching policies, user education, system hardening, *etc.*, is important to assure long term security; but, it operates under the inherent assumption that the attackers' behaviors will fall within presupposed bounds. Tactical defense addresses how to respond when the attacker succeeds in violating this assumption.

As network speeds, scales, and complexities continue to increase human response times will be insufficient to achieve good tactical defenses. It will become critical that attack response mechanisms be integrated into correlation systems if timely tactical defense is to be achieved. Currently, correlation systems do offer some automated response mechanisms. These, though, are typically disabled in operational deployments given their high propensity of causing self-inflicted damage. Automated response systems must mitigate their effects on normal network traffic, given that this is the corporate bread and butter.

Combined with the growing network speeds and complexities is the growing rise in the capabilities and complexities of attack tools and the use of the internet as an information sharing medium for attack communities. These issues lead to an interesting question: "When automated response mechanisms become integrated into operational correlation systems, can a knowledgeable attacker manipulate their attack process(es) such that the attack(s) either: (a) masquerade as normal traffic and go undetected or (b) masquerade as lower risk attacks such that the defender enacts sub-optimal responses? In particular, "Can this be done in a way that the defender cannot correct for?" If this is possible, then the attacker gains a significant advantage as the defender would believe they have strong security when this is in fact not the case. Such a potential would be well worth the attackers' efforts to identify.

Obviously, addressing these questions in a real-world context is important, but such an approach leads to issues that are inherently difficult to address. Specifically, assume a real-world network is identified (or simulated) which suffers from the above issues. For example, assume some previously unknown information comes to light which allows a previously missed or mis-assessed attack to be properly identified. If it is a novel attack then a new INFOSEC sensor can be designed to detect it. If it is a known attack then an existing sensors can be tuned to provide better attack separation. In either case, this specific instance of the above issue can be eliminated. But, this

This work was supported in part by the Canadian Natural Sciences and Engineering Research Council (NSERC).

S. W. Neville is with the Dept. Electrical and Computer Engineering at the University of Victoria, Victoria, BC, Canada (e-mail: Stephen.Neville@ieee.org).

falls short of solving the general problem. A new attack or attack methodology may occur that will re-express the same issues. This new missed attack will again go undetected (or incorrectly detected) until, by chance, some new information comes to light which provides evidence of its occurrence.

From the defender’s perspective the above questions are equivalent to: “What guarantee do I have that my correlation system is presenting me with a true picture of the network’s tactical state?”, or “Do alternate explanations exist for the observed evidence and, if so, does responding to the reported attack also mitigate these other possibilities?” It is prohibitive to answer these questions inclusive of the full complexity of operational large-scale networks. The approach taken in this work is, therefore, to assess whether the attackers possess such a capability within an idealized, defender advantageous, environment. In particular, all stages of the correlation system, with the expectation of the alert clustering (or alert fusion) sub-system, are assumed ideal. If the defender cannot guarantee tactical assessments are correct in this case, then they cannot do so in the real-world. Correlation systems are being deployed in real-world networks to direct tactical responses. Hence, assessing the potential for the postulated issues cannot be seen as secondary to solving the pressing false alarm issue. Both problems hinge, in part, on the ability to orthogonalize the space of network events within the composite feature space defined by the deployed sensors. Hence, the solution to both problems is interdependent.

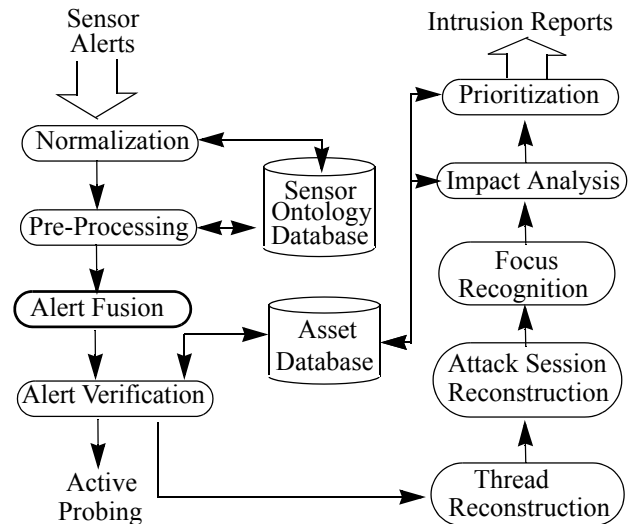
The purpose of this work is to lay the theoretical groundwork ahead of the future simulation and experimental validation work. This validation work has been left as an area of future work for three reasons. First, as detailed in [2], current alert correlation experimental testing methodologies fall significantly short of providing the required complete test coverage. Second, the modeling of real-world network traffic behaviors is known to be a hard problem [16][17]. Hence, under this domain, the question of whether the postulated issues are an artifact of a given simulation cannot be answered without first having developed a suitable theoretical framework. Third, as will be seen, the attackers’ ability to influence correlation systems in the manner postulated hinges on there existing attacks which can only be identified through analyzing shared alerts. This property depends on the exact nature of the deployed sensors. Hence, although the postulated problem may exist in general, the nature of its expression is likely to change from network to network, and therefore it would need to be assessed network by network.

Section II begins this work by reviewing the basic structure of alert correlation systems and reviewing the related works in the area. Sections III and IV then present game theoretic models of the attack and defenders behaviors, respectively, as these do not exist in the literature and are required to perform the subsequent analysis. Section V uses these models to shown that, even in an idealized environment, the attacker retains a capability to influence the low-level clusters’ contents in a

manner that is undetectable by the defender. Section VII then concludes the work. COTs-based networks and defenses are assumed throughout.

## II. STANDARD ALERT CORRELATION PROCESS

Alert correlation is a multi-stage process beginning with the normalizing of the generated low-level alerts into a common format and ending with the production of a prioritized list of intrusion reports (Figure 1). These reports provide the human analyst(s) with their viewpoint on the network’s tactical security state. Information flows only upwards through the hierarchy. Hence, a core component is the alert clustering (or alert fusion) stage as all further processing derives from its results. For the purposes of this work, all stages of the correlation process are assumed ideal in the sense that they perform optimally (or make the “best” decisions) based on the information they have at hand when the decisions must be made (*i.e.*, causality is maintained). If a higher level stage possesses information that could improve the alert clustering stage’s performance then this information is assumed to have been subsumed into the clustering decisions prior to the system’s deployment. Information coming to light after a cluster has formed cannot be used to restructure the cluster, as re-clustering is not currently supported alert correlation systems.



**Figure 1: The standard alert correlation stages, reproduced directly from [2].**

Figure 2 gives the details of the commonly used two-stage alert clustering algorithm, as abstracted from [2]-[10].  $a_k$  denotes a single alert generated by one of the deployed security sensors.  $C = \{c_j; j=1, \dots, N\}$  is the set of alert clusters currently existing in the clustering subsystem at the point when  $a_k$  arrives.  $C^* = \{c_k^*; k=1, \dots, M\}$  is the set of merged clusters in existence at the cluster merging stage.  $d(\cdot)$  is a similarity metric describing the degree of membership a given alert has with a given cluster in  $C$ .  $D(\cdot)$  is an inter-cluster similarity metric describing the degree of similarity

between two clusters existing in  $C^*$ .  $\varepsilon$  and  $\varepsilon^*$  are the respective inclusion and merging decision thresholds.  $t(\cdot)$  is a function that returns the set of detect times associated with a given alert set.  $L_j$  is a pre-defined threshold that fixes the maximum time duration over which alerts can be added to a cluster before the cluster is passed to the merging stage.

*Initialization:*  
 $C^* = \{\emptyset\}, M = 0$   
 $C = \{\emptyset\}, N = 0$   
*Cluster(alert<sub>k</sub>):*  
 For each  $c_j \in C$  {  
   if  $d(c_j, a_k) < \varepsilon$  then {  
      $c_j = c_j \cup \{a_k\}$   
     if  $\max(t(c_j)) - \min(t(c_j)) \geq L_j$  then {  
       Merge( $c_j$ ),  $C = C \setminus \{c_j\}$  }  
   }  
   if alert<sub>k</sub> has not been assigned to a  $c_j$  then {  
      $c_{N+1} = \{a_k\}, N = N + 1$  }  
  
*Merge( $c_j$ ):*  
   if  $\exists c_k^* \in C^*$  such that  $D(c_k^*, c_j) < \varepsilon^*$  then {  
      $c_k^* = c_k^* \cup c_j$   
   }  
   else  $c_{M+1}^* = c_j, M = M + 1$  }

**Figure 2: Alert clustering algorithm from [2]-[10].**

The cluster subsystem matches the asynchronously arriving alerts to existing clusters via the similarity metric  $d(\cdot)$ . For notational clarity, the time dependent nature of this matching has not been explicitly denoted but is assumed. If no existing cluster provides a suitable match for an arriving  $a_k$  then this  $a_k$  will initiate a new cluster in  $C$ . If the alert matches multiple existing clusters then it is added to all the clusters in  $C$  that it matches. If the age of the cluster, measured as the maximum difference in the detection times across a cluster's constituent alerts, exceeds  $L_j$  then the cluster is passed to the merging stage. At this next stage, the cluster is merged with the first cluster existing in  $C^*$  that it is found to be sufficiently similar to, as determined by  $D(\cdot)$ , if such a cluster exists. If no such cluster exists, then  $c_j$  becomes the newest member of  $C^*$ . All subsequent higher-level analyses are based on the clusters in  $C^*$ . Within the idealized defender model, all the parameters of the above model are assumed to be optimal as derived from the defenders' *a priori* knowledge of the possible attack set to which they may be subjected.

In keeping with Cuppen's work [5], it is assumed that a set  $\{L_j\}$  of clustering thresholds exist, with each element being the optimal threshold for its given attack type. Obviously, if an alert arrives which triggers a cluster's initiation and which

could be associated to more than one attack type then the defender has a fundamental problem in choosing which  $L_j$  would be assigned to the new cluster. The nature of this choice is outside of the scope of this work. For simplicity, it is assumed that the defender possesses an oracle that enables the optimal  $L_j$  to be selected.

It can be observed from the algorithm that the attacker's only avenue to influence an alert cluster contents is during the cluster's initiation. Once the cluster has been initiated all subsequent  $a_k$ 's for which  $d(c_j, a_k) < \varepsilon$  are included in the cluster. If  $d(\cdot)$  and  $\varepsilon$  are chosen appropriately, then the attacker has no post-initiation influence on the cluster. The attacker's goal, therefore, is to enact an attack process such that clusters will be incorrectly initiated. Obviously, in general, the merging stage cannot correct for such errors. To see this, assume  $a_k$  should have initiated cluster  $c_j$ , but  $d(c_k, a_k) < \varepsilon$  such that  $a_k$  is mistakenly included in  $c_k$ . If  $D(c_k, c_j) \geq \varepsilon^*$  then the merging stage will not re-combine the clusters. The remainder of this work will develop the notation and analysis necessary to show the conditions under which such "missed initiations" can occur. It should be noted that Julisch's work focuses solely on the correlation of non-malicious alerts [3]; hence, it does not suffer the issues addressed in this paper.

### III. ATTACKER MODEL

The notation and assumptions regarding the attackers are as follows. A set of atomic attacks  $\alpha = \{\alpha_1, \dots, \alpha_N\}$  are assumed to exist. All enacted attacks consist of a sequence of attacks chosen from  $\alpha$ . These composite attacks are denoted as  $\alpha_I = \{\alpha_k\}_{\forall k \in I}$ , where  $I$  is an index set on  $1, \dots, N$  which may possess non-unique entries. This is consistent with the available works in attack sequence analysis and attack graph generation, such as [11]-[14]. Multiple attackers are assumed to exist and single attackers are assumed capable of launching multiple attacks. Each atomic attack is enacted over a finite time period  $T(\alpha_k)$ . From the defender's perspective,  $T(\alpha_k)$  is a random variable drawn from an unknown distribution. No assumption is made as to whether the attackers are internal or external. This basic model is consistent with commonly used attacker models.

Additionally, it is assumed that the attackers are intelligent and rational as per the standard game theory definitions [15]. Rationality refers to the attackers enacting attacks that maximize their perceived utility in reaching their goals, where  $u(\alpha_k)$  denotes the perceived utility for attack  $\alpha_k$ . The attacker's utility maximization process can be viewed as a step by step trial and error search. Hence, the attacker need not know the complete composite attack *a priori*. The defenders' responses are assumed to be predicated on their estimates of the enacted attacks, as inferred by the observed evidence. Rationality directly implies that techniques capable of subverting the attack estimation process are used once they become known.

Intelligence, per its strict definition, denotes both the defender and attackers possessing the same knowledge about the network, its defences, and the set of possible attacks. Obvious-

ly, such complete knowledge on the attackers' behalf is generally unavailable. But, the attackers can gain information about the network and its defences through enacting attacks. Strictly speaking, computational complexity constraints would be required to formally bound an attacker's ability to gain such knowledge. Therefore, security must arise from intrinsic properties of the defensive systems and not through assumptions as to what the attackers do or do not know. Intelligence directly implies that security by obscurity is untenable.

#### IV. IDEALIZED DEFENDER MODEL

It is assumed that the defenders have deployed a heterogeneous set of cyber-security sensors  $\mathbf{s} = \{s_1, \dots, s_n\}$  throughout the network enclave(s) under their protection. Each sensor is assumed to contain a set of triggers  $\{x_k\}$ . These triggers are assumed to be binary decision functions that map the information space of the network events they observe, denoted as  $E_j$ , into  $\{0, 1\}$ . It is assumed that  $\bigcup_{\mathbf{s}} E_j = E$ ,  $E$  being the space of all events occurring within the network,  $e_m \in E$  being one such event. More precisely, each trigger  $x_k$  defines both a mapping  $g_k$  of the event space  $E_j$  into its own detection (or feature) space  $X_k$  (i.e.  $g_k: E_j \rightarrow X_k$ ) and a sub-space within this detection space,  $x_k \subseteq X_k$  that the event  $e_m$  must cover for the trigger to fire. Define an indicator function  $1_k(e_m)$  such that

$$1_k(e_m) = \begin{cases} 1 & \text{if } x_k \subseteq g_k(e_m) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

For simplicity, it is assumed that  $\forall \alpha_i \in \alpha \exists x_k$  such that  $1_k(\alpha_i) = 1$  (i.e. no attacks go undetected). Additionally, for simplicity, it's assume that  $\forall m, k$  if  $e_m \notin \alpha$  then  $1_k(e_m) = 0$  (i.e., there are no false alarms). If  $1_k(e_m) = 1$  then the trigger generates  $a_k$ . All  $a_k$ 's are assumed to report (i.e., contain) all of the information available from their respective  $x_k$  sub-spaces inclusive of the triggering event's source/destination information and the sensor's id. The arrival times at the correlation system of an attack's generated  $a_k$ 's are asynchronous random events as seen by the defender. It is assumed that  $\exists x_k$ 's for which  $1_k(\alpha_i) = 1$  and  $1_k(\alpha_j) = 1$  when  $i \neq j$ . Such alerts are denoted as *non-unique* alerts. If all attacks generate at least one alert which uniquely identifies the attack then optimal defence is trivially achieved solely by focusing on these unique alerts as, by definition, they provide an orthogonal mapping of  $\alpha$  into the attack detection space.

Alert uniqueness is a property of the deployed sensor suite. Hence, relying on a uniqueness assumption for defence entails a need to assess the composite deployed sensor suite. Typically, under COTs-based sensors this would require an extensive reverse engineering effort. Without a proof of uniqueness, a conservative engineering approach requires that one operate under the assumption that non-unique alerts exist and that optimal tactical defence must be sought in their presence. It should be noted that sensor characterizations, a requirement to determine alert uniqueness, are not typically performed within the cyber-security domain and commercial security sensor vendors

closely guard information about their trigger sets. Hence, the real-world problem of determining the extent to which non-unique alerts exist is a hard problem. To the author's knowledge the issue of non-unique alerts has not been addressed previously in the literature and it is a key determinant, along with false alarms and the general attack detection problem, in building large-scale network systems with assured levels of security.

Denote the set of alerts generated by a given  $\alpha_k$  as  $alerts(\alpha_i) = \bigcup_{\mathbf{s}} 1_k(\alpha_i) \cdot a_k$ , where  $1 \cdot a_k = a_k$  and  $0 \cdot a_k = \emptyset$ . This definition of triggers is inclusive of both misuse and anomaly detection approaches that use threshold-based decision functions. Without a loss of generality, all of the triggers' sub-spaces can be collected into a single composite information space  $X = \times_{\mathbf{s}} X_k$ . For simplicity,  $k$  can now be redefined as  $k = 1, \dots, N_x$  such that the  $x_k$ 's are now defined as the sub-spaces of  $X$  (i.e.  $x_k \subseteq X$ ), and  $N_x$  is the total number of such sub-spaces. The nature of  $X$  is a direct property of the deployed sensor suite. Without a loss of generality, all triggers are assumed to be neither redundant nor superfluous (i.e., for all collocated sensors  $x_k$  and  $x_m$  with  $m \neq k$  if  $1_k(\alpha_i) = 1$  then  $\exists \alpha_j \neq \alpha_i$  such that  $1_m(\alpha_j) = 1$  and  $\forall \alpha_k \in \alpha \neg \exists x_n \in X$  such that  $1_n(\alpha_k) = 0$ )

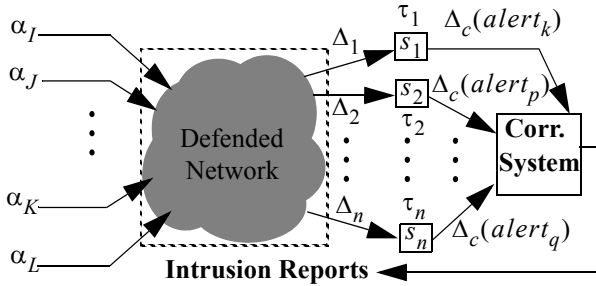
The goal of alert clustering is to generate, at the merging stage, a cluster  $c_k^*$  for which  $c_k^* = alerts(\alpha_k)$ . At issue is whether or not the attacker can influence the contents of these merged clusters such that an advantage is gained. More precisely, assume each  $c_k^*$  is used to generate an estimate  $\alpha_k$  of the associated enacted attack and that this estimate predicates a defensive response  $R(\alpha_k)$ . Denote the ideal defensive response as  $R^*(\alpha_k)$ . Assume the attacker gains an advantage if a response other than  $R^*(\alpha_k)$  is enacted for the given attack (i.e.  $\forall R(\alpha_k)$  with  $\alpha_k \neq \alpha_k^* u(\alpha_k | R(\alpha_k)) > u(\alpha_k | R^*(\alpha_k))$ ). If all higher correlation analysis steps are assumed ideal, then enacting an optimal response hinges on  $c_k^*$  containing sufficient information such that  $\alpha_k = \alpha_k^*$ . The rational attacker's goal is to identify attacks or attack processes such that  $\alpha_k \neq \alpha_k^*$ .

Obviously, the correlation system's higher level analysis steps can be structured to estimate the enacted composite attack  $\alpha_K$  instead of the individual atomic attacks, as done in the areas of attack sequence analysis and attack graph research. Under this extension,  $R^*(.)$  becomes the best response for the estimated composite attack  $\alpha_K = \{\alpha_k\}_{\forall k \in K}$ . The attacker's goal is to generate clustering errors for at least one  $\alpha_k \in \alpha_K$  such that  $u(\alpha_K | R(\alpha_K)) > u(\alpha_K | R^*(\alpha_K))$ . The attacker's fundamental intent therefore has not changed.

Under this extension, an argument can be made that erroneous  $\alpha_k$ 's can be removed in the higher-level analysis stages via approaches such as the identification of attack sequences in the presence of missing attacks [12][13]. This argument is valid provided that only one  $\alpha_K$  is plausible based on the observed evidence. If this is not the case then, by definition, the defender does not possess enough information to discount the possibility that an alternate  $\alpha_K$  is in fact the enacted composite attack. Hence, this approach reduces to needing to accurately estimate each  $\alpha_k$  in turn. Statistical likelihood methods cannot be used

to simplify the problem since, under the assumption of the attacker's rationality, the attacker then gains an advantage by enacting unlikely attacks. Discarding suspect  $\alpha_k$ 's does not provide a general solution since it also implies accepting the assumption that no alternate attack could have been generated the observed evidence. This is not to say that some errors may not be recoverable at higher-level analysis layers, only that such errors cannot be assumed to be generally recoverable. Obviously, not all of the alerts generated by an attack may be required to uniquely identify the attack. Denote  $I_c(alerts(\alpha_k))$  as the subset of alerts containing the critical information required to identify  $\alpha_k$  and assume that if  $I_c(alerts(\alpha_k)) \subseteq I_c^*$  then  $\alpha_k = \alpha_k$ . Then, without a loss of generality, those alerts for which  $a_k \notin I_c(alerts(\alpha_k))$  can be dropped from the analysis as they will not materially change the defender's response.

The above model is an defender advantageous idealized model in that it assumes: (a) all attacks are detected, (b) all of  $\alpha$  is *a priori* known to the defender, (c) no false alarms exist, (d) only critical alerts are considered. If, under these restrictions, the defender still cannot guarantee that optimal responses are always enacted then no such guarantee can exist in less idealized situations. Obviously, under this model the only option for the attacker is to manipulate their attack timing and/or coordination characteristics such that a critical alert is mis-clustered, thereby causing a sub-optimal response to be enacted.



**Figure 3: Effect of the network to time diffuse the enacted attacks.**

### V. ATTACKERS' INFLUENCE

The effect of the network is to diffuse the alert sets through time, as shown in Figure 3. Specifically, denote by  $\Delta_k$  the transit time of  $\alpha_k$  to each sensor.  $\tau_k$  denotes the decision time required by the sensor prior to generating  $a_k$ . Obviously, this time is dependent on the utilization of the system running  $s_k$  (i.e.  $\tau_k = \tau_k^0 / (1 - u_k)$  where  $u_k$  is the utilization and  $\tau_k^0$  is the average triggering decision time when  $u_k = 0$ ).  $\Delta_c(a_k)$  denotes the communication time required to transmit  $a_k$  from  $s_k$  to the correlator. For each enacted atomic attack the total time for  $alerts(\alpha_k)$  to reach the correlator is therefore,

$$T_{total}(\alpha_k) = T(\alpha_k) + \sum_{\forall k \in alerts(\alpha_k)} [\Delta_k + \tau_k + \Delta_c(a_k)]. \quad (2)$$

Assume for tactical defense that the defender must enact  $R^*(\alpha_k)$  prior to the end of  $T_{total}(\alpha_k)$  for the response to be effective. From the defender's perspective,  $T(\alpha_i)$ ,  $\Delta_k$ ,  $\tau_k$ , and  $\Delta_c(alert_k)$  are random variables drawn from unknown distri-

butions. Obviously,  $T(\alpha_k)$  by its nature can be influenced by the attacker, excepting cases where the structure of  $\alpha_k$  itself places a hard limit on  $T(\alpha_k)$ , as for example in single packet attacks. The attacker can also influence  $\Delta_k$ ,  $\Delta_c(alert_k)$ , and  $\tau_k$  through coordinated attacks targeting the intermediate network infrastructure, or attacks designed to overload  $s_k$ . Full DOS attacks are not required since such coordinated attacks need only to be sufficient to disrupt  $T_{total}(\alpha_k)$ . Obviously, variations in  $\tau_k$  can be tracked, and the attackers' influence on  $\Delta_c(alert_k)$  be eliminated by using out-of-band communication paths between the sensors and the correlator. But, unless all sensors are collocated, the attackers' influence on  $\Delta_k$  is not easily managed, as by definition the attacker can enact attacks on the network and, hence, has the access required to affect  $\Delta_k$ . In addition, modelling  $\Delta_k$  via its statistical moments is insufficient as it is its instantaneous value that is required to implement optimal clustering. Modelling network delays, such as  $\Delta_k$ , is a known hard problem [16][17]. The above analysis implies the defender cannot base the clustering on an assumed alert arrival ordering.

Under the idealized model, the only avenue open to the attacker is to identify and exploit the non-unique alerts within the critical alert sets. As discussed, these must exist or the defensive problem becomes trivial. Specifically, assume the attacker want to have  $\alpha_k$  be responded to sub-optimally. Then they need at least one  $a_k \in I_c(alerts(\alpha_k))$  to be absorbed into an already existing cluster, assuming the existing cluster's optimal response is not  $R^*(\alpha_k)$ . Hence, the attacker must cause this alternate cluster to be initiated prior to enacting the desired attack. This can be done in a controlled manner by "seeding" the required alert cluster through a *mimicry* attack.

For example, assume the deployed sensor suite includes a SNORT [18] sensor. SNORT signatures can be easily converted via NASL [19] to stateful NISSUS attack scripts, stateful scripts being required to avoid the pitfalls of SNOT [20] and Stick [21], as described in [22]. Executing the chosen script will initiate its associated alert cluster. Hence, the attacker can seed all of the required clusters prior to enacting  $\alpha_k$ . If  $I_c(alerts(\alpha_k))$  contains a suitably high percentage of non-unique alerts or if there is a reasonably high likelihood that the first arriving alert will be from this non-unique subset, then the attacker can be reasonably confident that the required mis-clustering will occur. Obviously, the attacker will only possess this capability for a subset of  $\alpha$ . But, it is distinctly to their advantage to exploit this subset as attacks structured in this manner will pass by undetected in the sense that the defender will not respond to the true attack. It has been recently shown that reverse engineering of COTs IDSEs' "secret" signatures is feasible [23]. Hence, such mimicry attacks must be assumed in the purview of rational and intelligent attackers for the a range of commercial INFOSEC sensors. Stating the above analysis more formally,

**Theorem:** For large-scale COTs-based real-world networks exposed to rational and intelligent attackers, if there ex-

ists a set of attacks for which (a) their critical information includes non-unique alerts and (b) their optimal responses are not identical then, under the standard alert correlation algorithm, the attacker can always develop a process, a subset of attacks, and an attack within this subset such that they can enact an attack to which the defender will react sub-optimally.

*Proof.* It is sufficient to show that one such attack set, attack process, and enacted attack exist under the idealized model and that the defender cannot address the issue by tuning the alert clustering algorithm's parameters. If all alerts are unique then such a set does not exist. But, defence is trivial under unique alerts and alert correlation would not be required. Therefore, the need for alert correlation implies that non-unique alerts exist; hence, the required subset of attacks can be found. Denote this subset by  $\{\alpha_k\}$ . By the arguments given earlier in Section IV, composite attacks need not be considered as they can be reduced analysing their atomic attacks. The attacker selects from  $\{\alpha_k\}$  the subset with the largest number of overlapping non-unique alerts. From this subset they select one attack  $\alpha_k$  to enact. Prior to enacting  $\alpha_k$ , the attacker seeds all of the other clusters to which the non-unique alerts of  $\alpha_k$  can be mapped. The attacker triggers these seed initiations through generating alerts not contained in  $\alpha_k$ . The ability of an intelligent attacker to do this is guaranteed through the existence of mimicry attacks. By definition, these seeded clusters cannot trigger  $R^*(\alpha_k)$ . The attacker then initiates  $\alpha_k$ . The only way the first alert  $a_1$  arriving from  $\alpha_k$  properly initiates its correct cluster is if  $a_1$  is unique. But, if  $a_1$  is unique then the defender need only focus on  $a_1$  to identify the attack and enact  $R^*(\alpha_k)$ . Hence, by definition,  $a_k$  could not have been part of  $\{\alpha_k\}$ . Therefore,  $a_1$  must be non-unique and it will be assigned to one of the seeded clusters and will not initiate  $\alpha_k$ 's cluster. As  $a_1$  is, by definition, part of  $I_c(alerts(\alpha_k))$  the defender will never enact  $R^*(\alpha_k)$ . The defender possesses no information by which mimicry attacks can be separated from their associated actual attacks at the time when the clustering decisions must be made. Hence, the defender cannot avoid this situation by modifying the alert clustering algorithm's parameters. ■

Given the above analysis, even in an defender idealized environment, the attacker can gain a significant advantage by enacting the postulated attack process. Hence, this provides a strong argument that sensor cross characterization is on the critical path to building systems that provide assured security. Such characterization are currently not performed.

## VI. CONCLUSIONS AND FUTURE WORK

In this work it has been formally proven that, under the standard low-level alert clustering algorithm of the alert correlation process, the attacker retains an ability to influence the contents of the clusters in a manner that is not correctable by the defender even when all other portions of the correlation system are assumed ideal. The work proceeds by formal analysis with models being developed of both the attackers' and defenders information and decision processes. Strictly speaking, the developed results only apply to large-scale networks using COTs-based defenses. Further research is required to

determine the degree to which the postulated existence of non-unique sensors alerts possess a risk within real-world environments, and the ease with which these risks can be exploited. Additionally, work is also required to identifying exactly to what extent clustering errors can be correct through the higher layer correlations stages and how the correlation process can be restructure to enable such corrections. This work lays the foundation for further the experimental and simulation work required to confirm and address the postulated issues.

## REFERENCES

- [1] S. Axselsson, "The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection", RAID 1999, West Lafayette, Indiana, Sept. 7-9, 1999.
- [2] F. Valeur, G. Vigna, C. Kruegel, and R.A. Kremmer, "A Comprehensive Approach to Intrusion Detection Alert Correlation", IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 3, pp. 146-169, 2004.
- [3] K. Julisch, "Clustering Intrusion Detection Alarms to Support Root Cause Analysis", ACM Transactions on Information and System Security, Vol. 6, No. 4, Nov. 2003, pp. 443-471.
- [4] H. Debar and A. Wespi, "Aggregation and Alert Correlation of Intrusion Detection Alerts", Conference on Recent Advances in Intrusion Detection (RAID 2001), pp.85-103, Oct., 2001.
- [5] F. Cuppens, "Managing Alerts in A Multi-Intrusion Detection Environment", 17th Annual Computer Security Applications Conference, Dec. 10-14, 2001, pp.22-31.
- [6] P. Ning, Y. Cui, D. Reeves and D. Xu, "Tools and Techniques for Analysing Intrusion Alerts", ACM Transactions on Information and System Security, Vol. 7, No. 2, pp. 273-318, May 2004.
- [7] H. Debar and A. Wespi, "The Intrusion-Detection Console Correlation Mechanism", RAID 2001, Davis, California, Oct. 2001.
- [8] B. Morin, L. Me, H. Debar, and M. Ducasse, "M2D2: A Formal Model for IDS Alert Correlation", RAID 2002, pp. 115-137, 2002.
- [9] D. Xu, and P. Ning, "Alert Correlation through Triggering Events and Common Resources", 20th Computer Security Applications Conference (ACSAC '04), 2004.
- [10] A. Valdes and K. Skinner, "Probabilistic Alert Correlation", RAID 2001, pp.54-68, Oct., 2001.
- [11] S. J. Templeton and K. Levitt, "A Requires/Provides Model for Computer Attacks", New Security Paradigms Workshop 2000, Cork Ireland, Sept. 19-21, 2000
- [12] P. Ning and D. Xu, "Hypothesizing and Reasoning about Attacks Missed by Intrusion Detection Systems", ACM Transactions on Information and System Security (TISSEC), Vol. 7, No. 4, pp. 591-627, November, 2004.
- [13] S. Noel, E. Robertson, and S. Jajodia, "Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distances", 20th Computer Security Applications Conference, pp. 350-359, Dec., 2004.
- [14] O. Sheyner, J. Haines, S., Jha, R. Lippmann, and J.M. Wing, "Automated Generation and Analysis of Attack Graphs", 2002 IEEE Symposium on Security and Privacy, pp. 273-284, 2002.
- [15] R. B. Myserson, Game Theory: Analysis of Conflict, Harvard University Press, 6th Ed., 2004.
- [16] S. Floyd, "Difficulties in Simulating the Internet", IEEE/ACM Transactions on Networking, Vol.9, No.4, August, 2001, pp. 392-403.
- [17] V. Paxson, "End-to-End Internet Packet Dynamics", IEEE/ACM Transactions on Networking, Vol.7, No.3, pp. 277-292, June, 1999.
- [18] Snort - The Open Source Network Intrusion Detection System, <http://www.snort.org>, 2005.
- [19] M. Arboi, The Nessus Attack Scripting Language Reference Guide, 2002, [http://www.nessus.org/doc/nasl2\\_reference.pdf](http://www.nessus.org/doc/nasl2_reference.pdf).
- [20] SNOT, Security Focus, <http://www.securityfocus.com/tools/1983>, 2005.
- [21] Stick, Security Focus, <http://www.securityfocus.com/tools/1974>, 2005.
- [22] SNORT faq, <http://www.snort.org/docs/faq/1Q05/node10.html>, 2005.
- [23] C. Kruegel, D. Mutz, W. Robertson, G. Vigna, and R. Kemmerer, "Reverse Engineering of Network Signatures", AusCERT 2005, Gold Coast, Australia, May 2005.

# Policy Segmentation for Intelligent Firewall Testing

Adel El-Atawy\*, Khaled Ibrahim, Hazem Hamed, and Ehab Al-Shaer  
School of Computer Science, Telecommunication, and Information Systems  
DePaul University, Chicago, Illinois, USA  
Email: {aelatawy,kibrahim,hhamed,ehab}@cs.depaul.edu

## Abstract

*Firewall development and implementation are constantly being improved to accommodate higher security and performance standards. Using reliable yet practical techniques for testing new packet filtering algorithms and firewall design implementations from a functionality point of view becomes necessary to assure the required security. In this paper, an efficient paradigm for automated testing of firewalls with respect to their internal implementation and security policies is proposed. We propose a novel firewall testing technique using policy-based segmentation of the traffic address space, which can intelligently adapt the test traffic generation to target potential erroneous regions in the firewall input space. We also show that our automated approach of test case generation, analyzing firewall logs and creating testing reports not only makes the problem solvable but also offers a significantly higher degree of confidence than random testing.*

## 1 Introduction

In enterprise networks, the security of the internal network against attacks, illegitimate traffic, and unauthorized access can be crucial to the success of the entire business operation. As firewalls are core elements in network security, deploying correctly functioning firewalls is mandatory to accomplish these security goals. Firewalls are filtering devices usually placed at the boundary of the network to block unwanted traffic from/to external networks, as well as to regulate traffic flow between domains inside the same network [6]. Firewalls usually undergo continuous modification to their internal design and implementation to improve performance or to add new features (*e.g.*, extra filtering fields, enhanced logging, etc.) Thus, the firewall matching functionality always needs to be tested and verified to have some assurance on the correctness of the firewall implementations.

---

\*On leave from Alexandria University, Egypt

The complete framework of firewall testing should contain two components: (1) generating test packets that test the firewall given a certain policy, and (2) generating various policies scenarios to test the firewall handling of different policy styles/configuration. Although both are needed to claim a complete testing environment for firewalls, our focus in this paper is on the first problem, *i.e.*, given a firewall/policy, how to ensure that the firewall implements this policy configuration correctly.

Testing the firewall by exhaustively injecting all possible packets into the firewall will be enough. However, this is infeasible due to the huge number of packets needed. Even if we try to restrict the test traffic to the range of relevant destination addresses only, it is still an impossible task to do. With destination address limitation and using many optimizations, the required testing time can be reduced from about  $4 \times 10^{13}$  years to 4.5 years at a test rate of 1G packets/second. Both traffic and time needed are still prohibitive. Random sampling/testing can also be used but its one-sided error probability (*i.e.*, probability of faulty firewall passes the test) is impractically high.

A smart method to select the packets used for testing is needed to save time and increase the test accuracy. Using a careful study of the policy, its rules, and the inter-rule interactions; the different decision paths of the firewall filtering algorithm can be tested individually. The whole range of different packet header values (*i.e.*, protocol, source address/port, destination address/port) is divided into different regions (segments) based on the policy. Each segment is tested according to some measures (weight) that reflect the importance of this segment in testing in order to achieve the best testing accuracy. The formalization, implementation and evaluation of this framework are discussed in this paper.

In the next section, a quick overview of related work is provided. Section 3 discusses the system framework and its different modules. The test data generator is discussed in Section 4. In Section 5 the system is evaluated. Finally, Section 6 presents the conclusion and plans for future work.

## 2 Related Work

Testing is considered the most important and time consuming phase in any production line. Testing techniques are categorized into: black-box testing and white-box testing [1, 4]. In black-box testing, the internal structure of the component under testing is unknown. In this approach, test case generation is performed either exhaustively or selectively based on statistical techniques. On the other hand, the white-box testing approach assumes the knowledge of the internal structure to generate the test cases. In the context of our work in this paper, the black-box testing approach is more appropriate for testing firewall as the filtering/matching implementation details may not be known.

We can generally categorize firewall testing into two groups: *offline* and *online*. In offline testing, simulation is used to mimic the network and firewall behavior to discover configuration errors. In [8], a CASE tool is used to model the firewall and the surrounding networks in order to generate firewall test cases against a list of simulated security vulnerabilities. While in [10] a firewall analyzer was presented to simulate firewall behavior given a certain policy configuration. In these approaches, the simulation is performed totally offline without injecting any packets into the network.

In online testing, real traffic is injected into the network to examine the corresponding firewall. A functional approach is presented in [9] where firewall vulnerabilities are classified based on the firewall functional units to run the corresponding penetration tests for different firewall products. In the CERT guide to system and network security [3], experts recommend a firewall testing strategy in which the boundaries of packet filter rules are identified and then test samples are generated from the regions immediately adjacent to each boundary. Although this looks to be the closest one to our approach, no formalization or specific technique was described.

In conclusion, most of the related work is focused on designing penetration testing using predefined test cases or based on network information and queries. We consider our work novel as it is the first to consider the policy structure and semantics for constructing the most critical test cases avoiding random and exhaustive testing.

## 3 Testing Framework Overview

The testing framework consists of the following components: segmentation module, segment weight analyzer, test packet generator, and test analyzer (see Fig. 1). The following is the description of each component.

- *Segmentation module*: This component converts the device-specific policies to a canonical policy format and segments them based on rule interactions [2] and network information.

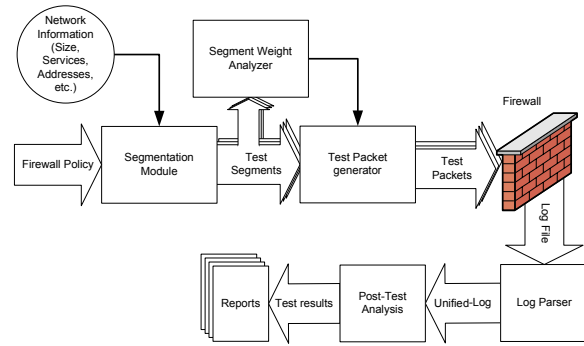


Figure 1. Firewall testing framework

- *Segment weight analyzer*: In this component, the segments are analyzed and each segment is assigned a weight based on a number of factors such as the number of rules intersecting in the segment and the size of the segment address space. The segment weight determines the testing intensity (number of generated test packets) for this segment.
- *Test packet generator*: This component generates test packets distributed in proportion to the segment weight. The evaluation of the segment weight is discussed in section 4.3. Each test packet carries test case information that includes a packet sequence number, the targeted segment id and the expected filtering action.
- *Test analyzer*: As a result of the firewall test cases, the firewall logs the matching results, which are then parsed and analyzed to validate the firewall operation. Using the test case information (embedded in each packet), the test analyzer matches the log results with the test case information to determine erroneous cases. In order to make this module device independent, the firewall log is converted to a unified format before the final analysis. The generated reports are stored in a repository for further inquiries and analysis.

## 4 Adaptive Smart Traffic Generation

There are three different approaches to generate test packets. (1) The *exhaustive* approach, in which we generate all possible packets, which requires a prohibitively large number of generated packets as well as long analysis time. (2) The *random* approach, in which, we select randomly a set of packets that will hopefully represent the different cases of the algorithm and hit an error if any exists. This mode generates a fewer number of packets but the probability of missing some of the decision paths of the filtering algorithm is quite high, as well as we do not differentiate between the importance of different regions of the filtering algorithm's input

space. (3) The *selective* mode, in which we consider the differences between different segments of the input space, and we can intelligently test the various decision paths of the algorithm.

In this section, we describe our selective testing approach in detail. Our main goal is to generate the least amount of traffic that is needed to test all possible decision paths for the given firewall policy. In order to have this span over the firewall/policy behavioral cases, we will have to define the space over which the entire policy could be tested.

**Definition 1** *The traffic address space is the space whose elements are all the different tuples that identify traffic flows in a network environment, especially from the firewall point of view. This is usually composed of the transport protocol, source address/port, and destination address/port. Other fields could also be used such as IP, TTL and flags.*

#### 4.1 Firewall rule representation

Rules and address spaces are represented as Boolean expressions where a packet satisfies this expression if and only if its header information is contained in this address space. We use Binary Decision Diagrams (BDD's) as a standard canonical method to represent Boolean expressions [5].

Each firewall rule is composed of filtering fields, namely: <protocol, source IP, source port, destin. IP, destin. port> The rule Boolean expression is created by concatenating the binary representation of all rule fields into one chunk of bits. Bits in this expression can be a value (0 or 1) or don't care. Each bit of this chunk is assigned a Boolean variable, and for each rule either the variable or its complement is taken into the expression, depending on the value of the corresponding bit in the rule. For example, consider this rule:

$$R = \langle \text{any}, 67.*.*.*, \text{any}, 121.*.*.*, \text{any} \rangle$$

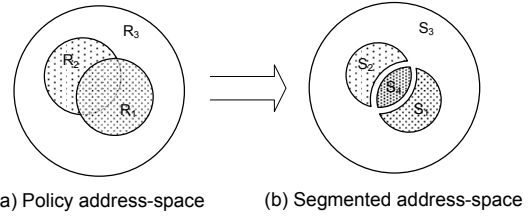
The corresponding Boolean expression ( $\phi$ ) includes only the non-don't care variables for the source IP (bits  $x_4$  to  $x_{35}$ ) and the destination IP (bits  $x_{53}$  to  $x_{83}$ ). Therefore:

$$\phi = (x'_4 \wedge x_5 \wedge x'_6 \wedge x'_7 \wedge x'_8 \wedge x'_9 \wedge x_{10} \wedge x_{11}) \wedge (x'_{53} \wedge x_{54} \wedge x_{55} \wedge x_{56} \wedge x_{57} \wedge x'_{58} \wedge x'_{59} \wedge x_{60})$$

As we see, 100 variables were reduced to 16 variables. This is a common case, where rules in policies are normally aggregates, and a small percentage of them use specific values for all its fields. In this paper we refer to the mapping from the firewall rule  $R_i$  into the Boolean expression  $\phi_i$  by the following function:

$$\phi_i = AS(R_i)$$

R1:	tcp	121.63.*.*: any	143.91.78.*: any	accept
R2:	tcp	121.63.71.*: any	143.91.*.*: any	accept
R3:	any	*.*.*.*: any	*.*.*.*: any	deny



Segment	AS	$R_{in}$	$R_{eff}$	OR	ACT
$S_1$	$\phi_1$	$\{R_1, R_3\}$	$\{R_1, R_2, R_3\}$	$R_1$	accept
$S_2$	$\phi_2$	$\{R_2, R_3\}$	$\{R_1, R_2, R_3\}$	$R_2$	accept
$S_3$	$\phi_3$	$\{R_3\}$	$\{R_1, R_2, R_3\}$	$R_3$	deny
$S_4$	$\phi_4$	$\{R_1, R_2, R_3\}$	$\{R_1, R_2, R_3\}$	$R_1$	accept

**Figure 2. Policy-based address space segmentation example.**

#### 4.2 Policy-based address space segmentation

Our goal here is to investigate the behavior as thoroughly as possible while keeping the testing traffic volume at a minimum. Thus, we need to determine all interactions exhibited in the firewall policy. In order to test every possible behavioral pattern (*i.e.*, decision path) of a firewall individually and efficiently, we selectively generate a small set of test packets tailored for each decision path. These decision paths can be determined by intersecting all the address spaces of the rules.

**Definition 2** *A segment is a subset of the total traffic address space that belongs to one or more rules, such that each of the member elements (*i.e.*, header tuples) conforms to exactly the same set of policy rules.*

In other words, packets that belong to the same segment are identical from the point of view of the policy. It is important to note that (1) all segments are pairwise disjoint, and (2) any segment can be uniquely identified via the rules that apply to it. The first property shows that there is no redundancy when segments are tested individually and the second property ensures that testing all segments implies testing all different decision paths with respect to the policy used.

To completely identify a segment, each one is associated with the following information:

- *Address space (AS)*: The Boolean expression representing the address space of the segment.
- *Included rules ( $R_{in}$ )*: Ordered list of rules (*i.e.*, as ordered in the policy) containing this segment.
- *Excluded rules ( $R_{out}$ )*: Ordered list of rules not covering this segment (*i.e.*, complement of  $R_{in}$ ;  $R - R_{in}$ ).

---

**Algorithm 1** DoSegmentation ( $R$ , defAct, InitDomain)

---

```
1:  $SEGLIST \leftarrow \Lambda$ 
2: AddSegment (InitDomain,  $\Lambda$ ,  $\Lambda$ , defAct)
3: for all rules:  $i = 1$  to  $n$  do
4:   for segments:  $j = SEGLIST.Count$  downto 1 do
5:      $S = SEGLIST[j]$ 
6:      $IncSeg \leftarrow S.AS \wedge AS(R_i)$  {Included part}
7:      $ExcSeg \leftarrow S.AS \wedge \neg AS(R_i)$  {Excluded part}
8:     if  $IncSeg \neq Seg.AS$  then {Segment not contained in
       the Rule's AS}
9:       if  $IncSeg \neq \Phi$  then
10:        AddSegment ( $IncSeg$ ,  $S.R_{in} \cup \{R_i\}$ ,  $S.R_{out}$ ,
           $S.R_{eff} \cup \{R_i\}$ )
11:        AddSegment ( $ExcSeg$ ,  $S.R_{in}$ ,  $S.R_{out} \cup \{R_i\}$ ,
           $S.R_{eff} \cup \{R_i\}$ )
12:       else {no intersection of rule and segment}
13:        AddSegment ( $ExcSeg$ ,  $S.R_{in}$ ,  $S.R_{out} \cup \{R_i\}$ ,
           $S.R_{eff} \cup \{R_i\}$ )
14:       end if
15:     else {Segment is inside the Rule's AS}
16:       AddSegment ( $IncSeg$ ,  $S.R_{in} \cup \{R_i\}$ ,  $S.R_{out}$ ,
           $S.R_{eff}$ )
17:     end if
18:      $SEGLIST.Delete$  (Segment  $j$ ) {delete original segment}
19:   end for
20: end for
21: return  $SEGLIST$ 
```

---

- *Effective rules* ( $R_{eff}$ ): Ordered list of rules that contribute to the formation of the segment.

- *Owner rule* ( $OR$ ): The first rule in  $R_{in}$  is taken as the owner of the segment.

- *Filtering action* ( $ACT$ ): Firewall action to be taken for this segment. This is taken as the action of the first rule in the  $R_{in}$  list (*i.e.*, the owner rule).

Fig. 2 shows a segmentation example of a simple firewall policy composed of three rules:  $R_1$ ,  $R_2$  and  $R_3$ . As a result of intersecting the three rules, four address segments are produced:  $S_1$ ,  $S_2$ ,  $S_3$  and  $S_4$ . The information associated with each segment is presented in the table in Fig. 2.

The segmentation algorithm (Algorithm 1) takes as input the policy rules ( $R$ ), the default action of the firewall (defAct), and the initial domain (InitDomain) which is the total traffic address space under consideration (*e.g.*, the internal network address space). As a result it outputs the list of segments induced by the given policy.

At the beginning, the initial segment is initialized, and added to SEGLIST (the list holding the whole list of segments created so far) at lines 1 and 2. Then we loop over the rules to impose their effect on all the existing segments. We use three variables just for the sake of readability of the algorithm;  $S$ ,  $IncSeg$ , and  $ExcSeg$  to refer to the currently processed segment, the overlapping portion of the segment and the rule's address space, and the non-overlapping portion

(*i.e.*, remaining address space of the segment), respectively.

We have three cases of interaction between the segment and the rule's address space; (1) either split the segment into included and excluded area, (2) leave the segment space intact as an excluded segment if the rule doesn't intersect with this rule, or (3) leave the segment space unmodified as an included space if the rule is a superset of the segment.

Restricting the segment adding (AddSegment) to non-empty ones (lines 8, and 9) is necessary; otherwise exponential running time in the number of policy rules is guaranteed as there will be a deterministic growth in the number of segments by doubling the number of segments after each rule, resulting in a list of  $2^n$  segments. It is also important to notice that the format constraints of firewall rules such as using contiguous address spaces and a limited range of field values significantly reduces the complexity of the segmentation algorithm, which, in the general case, could yield exponential running time.

AddSegment is used to add a new segment to the total segment list. If  $R_{in}$  is an empty list, the action ( $ACT$ ) of the segment is set to the default (defAct), otherwise it takes the action used in the first rule (assuming rule priorities are their order in the policy). Similarly, if the  $R_{in}$  is non-empty, the first rule is taken as the owner rule ( $OR$ ).

**Choosing the initial address space:** There are three ways to set the value of the initial address space (InitDomain). First, the initial domain can be the entire possible traffic address space (*i.e.*, corresponding to:  $\langle any, *.*.*, any, *.*.*, any \rangle$ ). Second, we can use the traffic address space of  $\langle any, *.*.*, any, \text{"network range"}, any \rangle$ . This can also be extended by ORing the  $AS$  of the network range with the  $AS$  of the desired additional address space such as multicast traffic (*i.e.*,  $\langle UDP, *.*.*, any, 224.*.*./4, any \rangle$ ). They differ in the address range used and the desire to restrict the testing scope. Third,  $InitDomain$  can be restricted to the address space used (*i.e.*, assigned to hosts and active services) in the network in order to further reduce the testing time. However, this limits the testing coverage and accuracy. To combine the advantages of these options, we consider each one with different testing intensity; the more general the initial domain the less dense the testing.

### 4.3 Measuring the segment weights

Having a measure for the importance (weight) of each segment is essential, as this leads us to decide how intense the testing should be within a certain segment. The weight of a segment is mainly determined by its inherent properties that might impact the filtering algorithm decision. The segment weights can also be adjusted by the administrator if necessary to reflect the business needs (*e.g.*, a segment containing critical domains or servers). From the testing point

of view, weight is a measure of the probability that an error can arise in this segment (*i.e.*, a packet is treated differently from what stated in the policy). We identify the following factors that affect the segment weight:

*Number of rules intersecting in the segment ( $|R_{in}|$ ):* As the number of rules increases within a segment, the filtering algorithm will face more decisions in order to find which rule should be applied, and as a result the potential of making errors increases. This factor is normalized by  $|R|$ ; the number of rules in the policy.

*Number of effective rules ( $|R_{eff}|$ ):* When a rule intersects non-trivially with a segment splitting it into two non-empty segments, this rule becomes a member of the set of effective rules of this segment. Similarly, as the number of these rules increases, the potential of making errors increases.

*Weight of the involved rules ( $wt(rule)$ ):* The involved rules in a segment ( $R_{in} \cup R_{eff}$ ) especially the owner rule (OR) are considered important because, effectively, packets match this segment only if they match the rules intersecting at this segment. Thus the weight of each of the rules involved in the segment affects the complexity of reaching the final decision. However, a greater emphasis is placed on the weight of the owner rule. In general, the rule weight ( $wt(r)$ ) is determined by (1) the complexity of the rule as presented in the rule format (*e.g.*, basic vs. extended), (2) the value of the fields (*i.e.*, specific values are more important than wildcards), and (3) the number of related rules preceding this rule.

*Segment address space size:* The main purpose of this metric is to allow generating more test traffic for segments that include more actually used addresses in the network because these segments are more critical for this particular network. This means that the testing intensity increases proportionally with the ratio of the used address space to the total allocated address space in the segment.

We formalize the testing intensity  $\rho_i$  of a segment  $S_i$  as a function that considers all of the factors discussed above:

$$\rho_i = w_1 \frac{|S.R_{in}| + |S.R_{eff}|}{|R|} + w_2 \cdot wt(S.OR) + w_3 \sum_{r \in R_{in} \cup R_{eff}} wt(r) + w_4 \frac{\|S.AS_{used}\|}{\|S.AS\|}$$

In this formula, each factor is given a weight ( $w_1 \dots w_4$ ) to enable tuning the segment-weight calculation function with the filtering algorithm under test. For example, when linear rule matching is used, more emphasis should be given to  $wt(S.OR)$ , so a possible weight assignment is ( $w_1=0.1$ ,  $w_2=0.7$ ,  $w_3=0.1$ ,  $w_4=0.1$ ). On the other hand, rule/field cross-producting algorithms [7] use overlapping sets of rules, and therefore more weight should be given to  $|R_{in}|$  and  $|R_{eff}|$ , for example ( $w_1=0.4$ ,  $w_2=0.1$ ,  $w_3=0.4$ ,  $w_4=0.1$ ). If the underlying filtering algorithm is unknown,

we recommend using equal weight values for all of these factors.

When test packets are generated, the packet header information is distributed on the policy address space in proportion to the calculated segment weights. During a test interval of  $T$  seconds and using a rate of  $R$  packets/second, the number of generated packets  $n_i$  for segment  $S_i$  is given by the formula:

$$n_i = \frac{T \cdot R \cdot \rho_i}{\sum_{S_j \in S} \rho_j}$$

#### 4.4 Post-test analysis

The post-test analysis is the process through which we can detect whether the firewall is functioning correctly or not through analyzing the testing results from the log. By processing the log, two possible types of errors can be detected. The first, is when the expected matching action is different than the observed matching action, while the second is when there is a malfunctioning device operation causing missing or duplicate log entries.

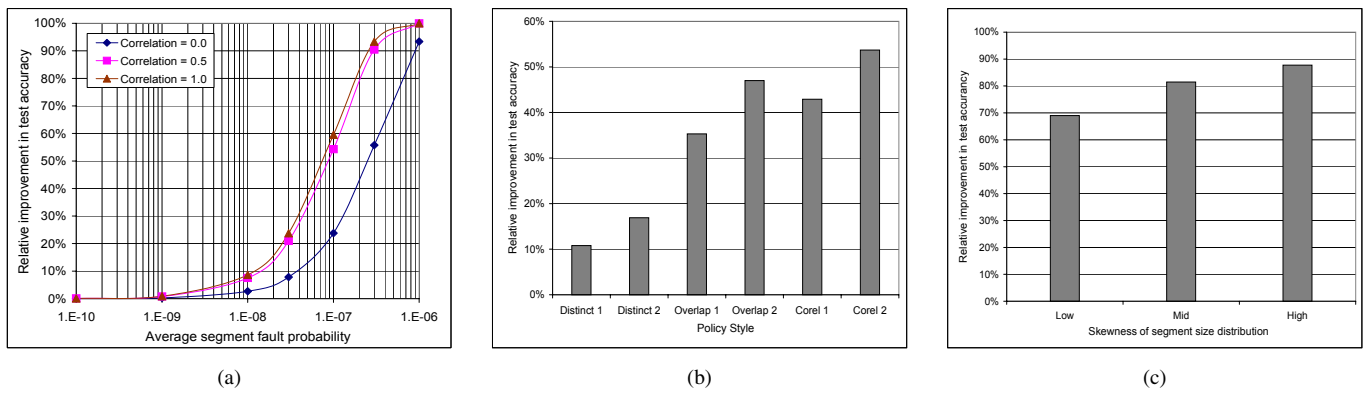
### 5 Evaluation and Results

As described in Section 4.3, the segment weight function was formalized in order to reflect a strong correlation with the probability of firewall implementation errors. In order to evaluate the efficiency of our testing framework, we investigate the effect of various parameters such as the degree of segment weight-fault correlation, policy style (*i.e.*, the interrelation between the rules within the policy [2]), and segment sizes, and then compare the accuracy of our scheme with the random test generator. The random tester generates packets uniformly over the whole policy address space within a given time period.

Fig. 3-(a) shows the effect of segment fault probability on the accuracy of our presented approach compared to the random approach using a full range of weight-error correlations. Our approach shows a significant improvement over the random tester even when zero correlation is used. This is attributed to the policy segmentation technique that ensures that each segment will be tested. This is unlike the random tester where entire segments might be skipped.

Fig. 3-(b) shows that our system gives significantly better results over the random sampling in all cases when the policy style is investigated. However, there is a general tendency that policies with high interaction and/or high portion of specific rules (*e.g.*, where all tuples are specified) would give better performance with our technique rather than with the random sampling approach.

Notice that in both Fig. 3-(a) and (b) we took a conservative approach by neglecting the tiny segments in order to



**Figure 3. Evaluation of different factors affecting the improvement in the firewall test accuracy relative to random-sampling test.**

remove their biasing effect. However, segments with very small address spaces are usually tested exhaustively, thus causing our technique to be superior over random testing but this might hide the effect of correlation and policy style.

Fig. 3-(c) shows the effect of the segment size distribution on the accuracy of our approach. It clearly indicates that the more the distribution is skewed the better the accuracy of the segmentation-based policy testing scheme. However, even with low skewness, the improvement is shown to be as high as 70% over random testing.

In conclusion, the segmentation based technique shows significant superiority over random sampling under various conditions. Moreover, when using more test options like exhaustive testing of small segments, and utilizing network information, the testing results become orders of magnitude more accurate within reasonable testing times.

## 6 Conclusion and Future Work

This paper presents a new efficient testing technique for the implementation of filtering-based security devices such as firewalls. The technique avoids both the exhaustive and pure random testing via segmenting and classifying the firewall policy address space based on critical testing factors such as the rule interaction, order, complexity, and network information. Using this approach, our evaluation study shows significantly better accuracy and performance than random testing. Our approach is also shown to be robust as it maintains better results than random sampling even when there is a small correlation between estimated segment weight and the probability of error/fault. When policies of various styles and different segment sizes are used in our evaluation study, our approach is shown to be superior in all cases. It is also shown that the policy segmentation approach has more advantage (as high as 70-90% over the random test-

ing) as rule interaction and/or the skewness of the segment size distribution increases.

Currently, our research is in-progress to bundle the system with a policy generator module in order to test a firewall under several scenarios of policy configurations. Also, studying the segmentation behavior for several policy styles needs further investigation. More refinement to the weight function to consider various filtering techniques used in IPSec and IDS devices is also under investigation.

## References

- [1] W. R. Adrion, M. A. Branstad, and J. C. Cherniavsky. Validation, Verification, and Testing of Computer Software. *ACM Computer Survey*, 14(2):159–192, 1982.
- [2] E. Al-Shaer and H. Hamed. Modeling and Management of Firewall Policies. *IEEE Transactions on Network and Service Management*, 1(1), 2004.
- [3] J. Allen. *The CERT Guide To System and Network Security Practices*. Addison-Wesley, 2001.
- [4] B. Beizer. *Black-Box Testing Techniques for Functional Testing of Software and Systems*. Wiley-VCH, 1995.
- [5] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.
- [6] D. Chapman and E. Zwicky. *Building Internet Firewalls, Second Edition*. Orielly & Associates Inc., 2000.
- [7] P. Gupta and N. McKeown. Algorithms for Packet Classification. *IEEE Network*, 15(2), 2001.
- [8] J. Jürjens and G. Wimmel. Specification-Based Testing of Firewalls. In *Proceedings of the 4th International Conference on Perspectives of System Informatics (PSI'02)*, pages 308–316, 2001.
- [9] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen. Analysis of Vulnerabilities in Internet Firewalls. *Computers and Security*, 22(3):214–232, April 2003.
- [10] A. Wool. Architecting the Lumeta Firewall Analyzer. In *Proceedings of the 10th USENIX Security Symposium*, August 2001.

# A Distributed Reputation Approach to Cooperative Internet Routing Protection

Harlan Yu  
Princeton University  
harlanyu@cs.princeton.edu

Jennifer Rexford  
Princeton University  
jrex@cs.princeton.edu

Edward W. Felten  
Princeton University  
felten@cs.princeton.edu

## Abstract

*The security of the Internet’s interdomain routing system hinges on whether autonomous systems (ASes) can trust the information they receive from each other via the Border Gateway Protocol (BGP). Frequently, this trust has been misguided, resulting in wide-spread outages and significant concerns about future attacks. Despite the seriousness of these problems, proposals for a more secure version of BGP have been stymied by serious impediments to practical deployment. Instead, we argue that the existing trust relationships between network operators (and the institutions they represent) are a powerful force for improving the security of BGP, without changing the underlying routing protocol. Our approach leverages ideas from online reputation systems to allow ASes to form a peer-to-peer overlay that integrates results from local network-management tools for detecting attacks and configuration errors. The proposed architecture is incrementally deployable, protects against shilling attacks, and deters malicious operator behavior.*

## 1. Introduction

The Border Gateway Protocol [21] is the Internet’s de facto interdomain routing protocol. The veracity of BGP routing information, passed as messages among ASes, is vital to the proper functioning of the Internet. BGP provides no intrinsic facility for an AS to validate the truthfulness of a received path update message— an AS must blindly trust the legitimacy of each route it receives. Illegitimate routing information can be propagated both by unintentional routing misconfigurations [17] and attacks by a malicious adversary [20]. Both types often cause traffic instability and reachability problems for at least a small subset of ASes on the Internet. BGP failures can even cause severe widespread outage as witnessed by the infamous AS7007 incident [10], when a flood of er-

roneous route advertisements caused global connectivity problems. BGP attacks can also force traffic redirection, interception, or modification in the case of a malevolent eavesdropper.

In order to defend against fallacious BGP update messages, the research community has focused on two broad classes of possible solutions. On one hand, researchers have proposed modifications to the actual BGP protocol and the addition of a centralized public key infrastructure (PKI) or a routing registry [14] [15] [19]. Adoption of these schemes have thus far failed for a variety of reasons: the potential for out-of-date central databases, the inability for incremental deployment or the prohibitive overhead of expensive cryptographic operations. On the other hand, researchers have suggested a multitude of analytic tools to mitigate the effects of erroneous updates locally at each AS [12] [22] [24]. While many have proven useful, many detect only specific types of misconfigurations and attacks. These locally-deployed tools also do not take advantage of AS coordination to resolve interdomain routing problems from multiple vantage points. Others have recognized the need for real-time inter-AS collaboration [11] [4] [18] but have either restricted adoption using a centralized “members only” paradigm or have focused only on tactics to mitigate a particular class of attacks.

In this paper, we contend that network operators can capitalize on existing trust relationships to better address any BGP routing fault by sharing existing results from local debugging tools. We propose a novel distributed reputation protocol that is particularly well-suited for inter-AS cooperation. The key idea is to mimic real-world trust relationships in a peer-to-peer (P2P) social network and weigh the information gathered from more trusted colleagues. Our approach requires no modifications to the BGP protocol and does not call for a centralized PKI or registry as in previous proposals. We demonstrate the system’s potential to complement existing network management techniques and amplify the detection of both misconfigurations and attacks.

## 2. Distributed Reputation

In this section, we first describe the challenges posed by online reputation systems in general and propose a novel solution of building real-world trust relationships into distributed online environments. We then discuss specifically the protocol's advantages and limitations in a BGP-specific context.

### 2.1. Essential Properties

When people look for information about unknown entities, they generally seek the opinions and recommendations of those who they regard to have good reputations. This works quite well in the real world— authentication is no harder than recognizing one's voice or face, and reputations are persistent and permanent. Major problems arise when we apply the concept of reputation to the online world. The Internet enables people to have ephemeral identities, pseudo-anonymous communications, and little accountability for their actions. Those who have built up bad reputations in an online community can simply shed this identity and return with a clean slate using a new pseudonym.

Trust in the interdomain routing context is based on the personal relationships between human network operators who manage ASes. Especially among backbone and other large transit ASes, many network operators have already developed trust relationships through personal and repeated contact at meetings of the North American Network Operators' Group (NANOG) [7] or similar groups in other geographic regions (*i.e.*, AfNOG [1] and SANOG [5]). Additionally, trust between ASes can be formulated through successful business relationships or known adherence to best common security practices [6]. An example is the Ivy Plus Administrative Computing group, a community of network operators from university campuses with an adopted set of group practices [2]. Network operator communities are suitable for online reputation systems since the barrier to entry is high— it is difficult for a notorious operator to shed a bad reputation and start afresh.

In general, any useful online reputation scheme requires at least the following four properties:

1. **A foundation of trust:** Any reliable reputation system must build upon repeated interactions between any two principals in the system. A principal must be able to assess its past interactions with other individuals in the system and be able to act accordingly to those judgments.
2. **Carrots and sticks:** To be useful, a system must provide strong disincentives to deter malicious be-

havior and levy effective punishments appropriately. Analogously, a system can also encourage and reward trustworthy behavior.

3. **Robustness against shilling:** The system's architecture must be robust against shilling attacks by dishonest principals. False ratings injected by an adversary must either have minimal affect on the system or be easily detectable. It must also be difficult for an existing principal to drop a bad reputation and start anew. This limits the number of repeated "hit and run" attacks by the same principal.
4. **An accurate scoring system:** Ratings distributed to and calculated by principals in the system must be both understandable and verifiable. Users must be able to accurately interpret the ratings in order to act knowledgeably on the recommendation. The ratings might also be weighted towards the most recent indicators of behavior.

### 2.2. A Trust-based Overlay Network

The essence of the distributed reputation architecture is to construct a P2P overlay topology that mirrors existing real-world trust relationships. This overlay network will then be used to implement distributed voting where peers vouch for the truthfulness of boolean propositions. We make the assumption that each AS is represented by a single node in the network.

A logical link in the overlay network is constructed between two nodes if one AS has an offline trust relationship with another AS. As such, nodes must swap authentication information through some form of out-of-band communication to establish the link. Authentication can come in many forms depending on the specific implementation of the protocol. For example, network operators might swap identifying information in person at NANOG key signing parties [3], over the phone or using e-mail. Note that the logical links are not limited to the set of immediate BGP neighbors, but can reach far across the physical network. The links are also unidirectional since the trust might not be reciprocated. The result will effectively be an online social network, where neighbors in the overlay are trusted acquaintances in real life: a node will trust its direct (one-hop) neighbors the most, the neighbors of those nodes (two hops away) less than its direct neighbors but more than others, and so forth.

Formalizing this idea, we define a variable weight factor  $\alpha$ , where  $0 < \alpha < 1$ , that geometrically decreases the level of trust as the relationship between two nodes grows distant. We will show in the next section how to

use this weight factor to share trustworthy information through distributed voting.

### 2.3. Distributed Voting

Having established the trust-based overlay, nodes will initiate queries in the form of a boolean bit-string, also called a proposition. For example, a node can broadcast the simple proposition “Is  $AS3 \rightarrow AS7$  a spoofed edge?” and solicit responses from other peers in the network. A response is simply a vote cast in ternary fashion  $(-1, 0, +1)$ :  $-1$  indicates that the node believes the proposition to be false;  $+1$  for true; and  $0$  means that the node is either neutral to the proposition or doesn’t have enough information to make a judgment.

As votes are cast, each node in the system will iteratively recompute its overall vote for each proposition based on both its own judgment and the votes of all other nodes in the system. To demonstrate this calculation, we consider a single node  $N$  in the network, with weight factor  $\alpha$ , for a proposition  $P$ . Node  $N$  will cast its own vote, a ternary choice  $V_N$ , based on its local analysis of edge  $AS3 \rightarrow AS7$ . Node  $N$  will also receive overall votes from its direct one-hop neighbors dynamically as they are cast and pushed out toward  $N$ . It will then compute the average of its neighbors’ scores to get  $V_{avg}$ . To mimic taking friends’ recommendations in real-life decision-making, node  $N$ ’s overall vote for  $P$  will be the weighted value  $\alpha V_N + (1 - \alpha)V_{avg}$ . By assuming that each of  $N$ ’s neighbors use the same weight value  $\alpha$ , the average of all votes from node  $N$ ’s  $L^{th}$ -degree friends contribute  $\alpha(1 - \alpha)^L$  to its overall score for  $P$ . Since  $0 < \alpha < 1$ ,  $N$ ’s overall score for  $P$  will converge despite the iterative recomputation and mutual dependency of weighted votes between itself and its neighbors.

At a particular AS, a separate server will run the distributed reputation service that verifies characteristics about underlying BGP paths with the opinions of trusted peers in the overlay. The server will monitor received paths through internal BGP sessions with the AS’s border routers. When a BGP route is flagged as suspicious (*i.e.*, a query returns a negative response, below a given threshold  $-T$ ), the server will automatically reconfigure the the router’s policies to filter out the suspect route. To implement a reputation node, a network operator need only modify the configuration of the router, not the underlying hardware or software of the existing routing mechanism.

### 2.4. Protocol Advantages

The distributed reputation protocol does not require a centralized PKI or registry. By constructing an overlay

according to the protocol, trust relationships are automatically built into the network. We show below that our protocol adheres to the three properties of a useful reputation scheme and assert three additional systemic advantages pertinent to BGP security.

1. **Strong deterrent for malicious behavior:** The primary architectural benefit is the protocol’s suppression and deterrence of misbehavior. Acting maliciously hurts an AS’s direct neighbors the most since the weight factor  $\alpha$  skews the impact of lies to hurt those who trust him. Repeated lies will not only jeopardize his online reputation, but may also lead to other real-world repercussions through the loss of real-world trust. Even if a node wrongly trusts a misbehaving or compromised AS, the node can simply disconnect it as a direct neighbor in the overlay network to prevent future harm.
2. **Difficult to shill the entire system:** The weight factor  $\alpha$  also forces a node’s vote to have geometrically smaller impact the further away it propagates. The effect of a single malicious vote, even coming from a highly regarded AS, will have only a small impact on direct peers and a negligible effect on ASes multiple hops away in the overlay. Even more importantly, colluding adversaries who form a clique in the overlay network can only do as much damage as a single isolated adversary. The best attack, where colluding adversaries attach to many locations in the network, is made difficult by the need of out-of-band authentication and real-life trust relationships.
3. **Intuitive scoring system:** With weight factor  $\alpha$ , an AS can look at its overall vote for a proposition and infer both its own and its neighbors’ beliefs on that query. A overall vote  $> \alpha$  or  $< -\alpha$  signifies a personal “true” or “false” belief, respectively. The vote’s variation from  $\pm\alpha$  signifies the confidence level of its peers. A vote between  $-\alpha$  and  $\alpha$  denotes a lack of information to make an informed decision. The AS either has a conflicting opinion from its neighbors, or has no opinion and is relying solely on his peers’ opinion of the proposition.
4. **Incrementally deployable:** Our proposed solution makes no changes to the de facto BGP protocol or the packet format of current BGP path announcements. The reputation system is completely disjoint from BGP’s control plane and can be adopted voluntarily at any time by network operators. The system would also function regardless of the choice of local detection tools run at each AS.

5. **Automated voting and decision-making:** Since the protocol depends on local analysis tools, it is possible to build in hooks that allow the existing mechanisms to automatically cast votes in the system. In addition, if an AS computes an overall score for a proposition that is above or below a set threshold  $T$ , it can automatically advise its routers to filter out suspicious paths and choose a more trustworthy one.
6. **Confidentiality of AS relationships:** ASes closely guard confidential information about their business relationships with other ASes. An AS can simply abstain from voting on a proposition that might divulge its business relationships with its neighbors. Overall votes are also passed only to one's direct neighbors since votes will be aggregated and averaged before being propagated further. Cryptographic mechanisms will encrypt and sign the votes cast in the overlay and provide repudiability such that the votes will be unverifiable to a third party.

## 2.5. Protocol Limitations

As a general framework, the reputation system is *not* inherently capable of detecting BGP misconfigurations and attacks itself. It is only useful when deployed in conjunction with the collection of available tools for debugging local networks. The positive tradeoff, though, is that the architecture itself is agnostic to a specific problem in question, allowing *any* boolean proposition to be raised and voted on. For a given proposition, the protocol works well if a minority of participants lie or believe a particular lie. Otherwise, the system could provide a slippery slope as principals unknowingly trust and propagate a faulty belief.

An honest AS might propagate false information in the reputation system if its local detection tools are not sufficient to uncover the fault. Given that *some* nodes in the overlay are able to diagnose the fault, the node would ideally discover the fault based on information collected from these trusted overlay peers. A trustworthy AS might also advertise bogus votes if it is compromised by a masquerading attacker. To reduce this threat, one basis of trust must be an understanding of a peer's security policies (*i.e.*, it follows best common security practices) before making a direct overlay connection. It matters not why an AS is receiving false information from another AS as long as it can manually drop the neighbor as a peer in the overlay when necessary.

As such, the reputation system is initially a reactive solution that may yield a small time lag between the injection of a new fault and its global detection. This time lag stems from the duration necessary for ASes to

run local detection mechanisms and for these results to propagate through the reputation network. Once enough ASes discover the fault locally, the scheme begins to behave proactively for ASes who have not yet detected this fault. For example, upon receiving a BGP update message, a node can immediately reject a route based on a precomputed negative rating  $< -T$  about this route gleaned from the trust system. An AS could alternatively accept a positively-rated route  $> T$  despite its appearance on bogon filter lists [8].

By building the overlay network on top of routes in the potentially faulty underlying network, an adversary could also attack the overlay itself. End-to-end encryption and signing of votes would protect the injection of bogus votes, but an adversary could still divert legitimate votes from its intended destination. A plausible yet expensive solution would be to create a separate out-of-band network that directly connects trusted peers.

## 3. Practical Applications

Currently in BGP, we can already verify *who* is speaking by using standard cryptographic techniques pairwise between communicating ASes. The preeminent problem in interdomain routing is verifying the truth of *what* ASes actually say. We hypothesize that our distributed reputation protocol can facilitate the detection of both misconfigurations and attacks. Our approach would make existing tools more effective by allowing trusted ASes to share data about widespread routing errors. Below, we speculate about the reputation system's BGP security potential using a preliminary examination of two types of common faults: prefix hijacks and invalid AS paths.

Our sketch relies on the common notion that it is impossible to detect all BGP faults affecting a single AS locally at that AS. While ASes are well-equipped to make truthful claims about routing connectivity within very close proximity (*i.e.*, prefixes originated by an immediate customer), they cannot be confident about connectivity claims many hops away. The vast majority of conflicts only affect a subset of ASes that are oblivious to the routing fault. Thus, it is imperative that ASes possess a means to verify with trusted peers the routes propagated from across the network.

### 3.1. Prefix Hijacks

One type of invalid route announcement is an IP prefix hijack. This occurs when an AS announces direct reachability to an IP prefix it does not actually own, either inadvertently or with ill-intent. This scenario often yields two announcements, a legitimate and a spoof,

from different ASes claiming to originate the same IP prefix. Zhao *et al.* studies extensively these Multiple Origin AS (MOAS) cases and suggests a method for detecting when these types of failures occur [24]. Their solution seems effective at raising alarms upon detection but offers no real capability to determine which announcement is actually legitimate. Their suggestion of a DNS database with valid origin information will inevitably become outdated and unreliable for practical use.

As an alternative, operators can amplify the above method by utilizing a distributed reputation system to identify the legitimate announcement upon a raised alarm. A hijacked prefix announcement only reaches a subset of the network since ASes closer to the real origin will continue to use the legitimate route. Assuming that an AS has overlay peers at diverse locations in the physical network, it can check upon detecting a new MOAS alarm whether the new route is a hijacked prefix. The conflicted AS could propose a boolean inquiry such as “Is AS88 entitled to originate prefix 128.112.0.0/16?” to assess the legitimacy of the route at hand. Unlike connections in the physical AS topology, trusted peers in the overlay would presumably have better information since well-configured ASes are more likely to filter bogus advertisements and less prone to adversarial reconfigurations. Another example is if  $AS_1$  in the United States trusts  $AS_2$  in Asia to originate prefix  $P$ .  $AS_1$  could then easily disregard a future prefix hijack of  $P$  even though it is many hops away in the physical AS connectivity. Even if the AS has no overlay peers outside the affected area, it may still be helpful to share local traceroute and data plane verification [22] information with other affected peers.

### 3.2. Invalid AS Paths

Another type of BGP fault is the announcement of an AS path that does not actually exist. Kruegel *et al.* have formulated a heuristic to detect invalid path anomalies based on insights about the Internet’s general topology [16]. They divide the topology into a group of core backbone ASes and clusters of periphery ASes. First, they claim that any valid AS path can never traverse the Internet backbone more than once and thus may contain only a single subsequence of core ASes. Their second constraint stipulates that all consecutive pairs of periphery ASes in the path must either be in the same cluster or two geographically proximate clusters. With the Internet’s topology constantly evolving, it can be practically difficult to obtain a precise and up-to-date topology using a service like RouteViews [9]. Moreover, an AS might not trust the BGP routing information provided by

these data publishing services.

To add precision to the above heuristic, an AS could issue propositions in the reputation system for each AS-AS edge in question to determine topological connectivity. The distributed reputation system can even be used to determine the cause of the invalid path. For example, a network operator concerned about a potential spoofing attack can poll her trusted peers with the proposition “Is the edge  $AS3 \rightarrow AS7$  a spoofed edge?”. Alternatively, she could also pose the proposition “Is the edge  $AS3 \rightarrow AS7$  currently down?” if she suspects a temporary failure. Augmenting the ideas proposed by Kruegel with reputation could yield a more accurate network model with built-in confidence. It will also allow ASes to probe globally for fine-grained information to pinpoint local conflicts.

### 3.3. Other Applications

Our solution is obviously not limited to these two above possibilities, but can be used generally any time information from collaborating vantage points might be useful. Another potential application would be to apply root-cause analysis in real time using the techniques described by Feldmann *et al.* [13]. Using RouteViews data from multiple vantage points, they demonstrate a method to locate a set of suspect edges that might have initiated an AS path change. A network operator might not trust information provided by the vantage points or might not have a complete view of other ASes. A distributed reputation system would introduce confidence in the accuracy of collected data and would integrate information from all border routers at an AS into individual votes.

Wu *et al.* developed a tool to pinpoint anomalies such as flapping prefixes and large traffic shifts near an AS by monitoring BGP updates at its border routers [23]. An AS implementing this tool could make assertions about traffic anomalies within close proximity and merge these results with analysis from other trusted vantage points using the reputation system. One might also utilize reputation systems to generate better AS topologies, to detect policy conflicts, to defend against denial of service attacks or to resolve other interdomain security issues.

## 4. Conclusion and Future Work

This paper demonstrates the promise of using our novel distributed reputation protocol to protect the Internet’s interdomain routing infrastructure. Our solution diverges from recent work by focusing on how ASes can collaborate in a trustworthy manner to improve resilience against attack and misconfiguration. We de-

scribe a lightweight overlay protocol that can be adopted incrementally by the AS network operator community. The reputation architecture significantly raises the difficulty of performing large shilling attacks and provides a strong deterrent against malicious behavior. Adopters will be able to receive and share valuable routing information with well-reputed peers at many vantage points and still keep sensitive data confidential.

As this paper hypothesizes about our reputation system's potential benefits for BGP security, future work will focus on implementation specifics and validation. We plan on devising efficient methods to dynamically recalculate and manage the large number of received votes. We also aim to quantify how resilient the system is against shilling attacks by isolated and colluding adversaries in a realistic Internet topology. We will also look to determine what specific queries work best to debug different types of BGP faults in practice and attempt to integrate existing local tools to optimize accurate and instantaneous voting.

## Acknowledgment

We thank Nick Feamster and the anonymous reviewers for their many detailed and insightful comments.

## References

- [1] Africa Network Operators' Group. <http://www.afnog.org>.
- [2] IvyPlus Admin Computing Practices. <http://ivyplus.stanford.edu/practices.html>.
- [3] NANOG PGP Key Signing. <http://www.nanog.org/pgp.abley.html>.
- [4] NSP-Security (NSP-SEC) Info Page. <https://puck.nether.net/mailman/listinfo/nsp-security>.
- [5] South Asian Network Operators Group. <http://www.sanog.org>.
- [6] The Network Reliability and Interoperability Council: Best Practices. <http://www.bell-labs.com/user/krauscher/nric>.
- [7] The North American Network Operators' Group. <http://www.nanog.org>.
- [8] The Team Cymru Bogon Reference Page. <http://www.cymru.com/Bogons>.
- [9] University of Oregon Route Views Project. <http://www.routeviews.org/>.
- [10] V. Bono. 7007 Explanation and Apology. <http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html>.
- [11] K. Cooper. An Information Sharing and Analysis Center for the Internet. <http://www.nanog.org/mtg-0105/cooper.html>, 2001.
- [12] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. USENIX NSDI*, 2005.
- [13] A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs. Locating Internet Routing Instabilities. In *Proc. ACM SIGCOMM*, 2004.
- [14] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing. In *Proc. ISOC Network and Distributed Systems Security*, 2003.
- [15] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, 2000.
- [16] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Topology-based Detection of Anomalous BGP Messages. In *Proc. RAID*, 2003.
- [17] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfiguration. In *Proc. ACM SIGCOMM*, 2002.
- [18] K. Moriarty. Distributed Denial of Service Incident Handling: Real-Time Inter-Network Defense. <http://ietfreport.isoc.org/all-ids/draft-moriarty-ddos-rid-06.txt>.
- [19] J. Ng. Extensions to BGP to Support Secure Origin BGP (soBGP). IETF Internet Draft [draft-ng-sobgp-bgp-extensions-02.txt](http://draft-ng-sobgp-bgp-extensions-02.txt), April 2004.
- [20] O. Nordstrom and C. Dovrolis. Beware of BGP Attacks. *ACM SIGCOMM Computer Communications Review*, 34:1–8, 2004.
- [21] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, March 1995.
- [22] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proc. USENIX NSDI*, 2004.
- [23] J. Wu, Z. Mao, J. Rexford, and J. Wang. Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. In *Proc. USENIX NSDI*, 2005.
- [24] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. Detection of Invalid Routing Announcement in the Internet. In *Proc. IEEE International Conference on Dependable Systems*, 2002.

## AUTHOR INDEX

---

Al-Shaer, Ehab .....	67
Bertino, Elisa .....	37
Chen, Ing-Ray .....	13
Cho, Jin-Hee .....	13
Choi, Jong Youl .....	25
El-Atawy, Adel .....	67
Felten, Edward .....	73
Ganesh, Lakshmi .....	49
Goodloe, Alwyn .....	1
Gorlatova, M. ....	55
Greenwald, Michael .....	7
Gunter, Carl .....	1, 7
Gupta, Minaxi .....	25
Hamed, Hazem .....	67
Heinbockel, William .....	43
Ibrahim, Khaled .....	67
Jacobs, Matthew .....	1
Keshav, Srinivasan .....	31
Khanna, Sanjeev .....	7
Kwon, Minseok .....	43
Lamont, L. ....	55
Lee, David .....	19
Li, Na .....	19
Lin, Weili .....	19
Liu, Zhijun .....	19
Mason, P. ....	55
Neville, Stephen .....	61
Rexford, Jennifer .....	73
Seth, Aaditeshwar .....	31
Shah, Gaurav .....	1
Sherr, Micah .....	7
Shin, Youngsang .....	25
Shue, Craig .....	25
Venkatesh, Santosh .....	7
Wang, M. ....	55
Wu, Xiaoxin .....	37
Yu, Harlan .....	73
Zhao, Ben Y. ....	49