An Adaptive Security Policy Design and Management for Distributed Systems

Veli Hakkoymaz, İsmail Alan

Abstract—**This paper presents an adaptive security policy management scheme for the threats in distributed systems. First, we investigate the security aspects of computer networks architecture and observe that security services are provided at application, transport and network layers of OSI model. Network layer security presents some important opportunities for offering controls over security threats to distributed information systems connected to the global Internet. Important details of Internet Protocol (IP) security services are studied and enhancements are proposed over handling both protected and unprotected packet traffics for a dynamic environment. Dynamism is in terms of severity of security threats. Our proposal includes a security policy manager (SPM), an adaptive security control mechanism that would automatically configure security levels for a host in response to the frequency of security attacks.**

Index Terms—**Adaptive security, computer network, IPSec, security model, security policy manager**

I. Introduction

Information systems are growing in large-scale distributed systems that operate in unbounded network environments such as Internet. In these systems, there is neither a centralized control nor a unified security policy to differentiate trustable sites from non-trustable sites [1, 10]. Security threats such as *interception* (i.e., illegal access to data or services), *interruption* (i.e., denial-of-services), *modification* (i.e., change of data or services from their original specification) or *fabrication* (i.e., appending additional data or activity) always exist in a distributed system [6, 7]. Many different conditions can influence these security threats to materialize. Origin of threats against the systems can be deliberate or accidental. For example, security flaws in design and organization invites attacks to take place to such a system. For that reason, an organization's system connected to an unbounded network would be vulnerable to attack from many potentially harmful sources on a public Internet.

Manuscript received October 17, 2007. This work was partially supported by the Fatih University research fund.

Veli Hakkoymaz is with Fatih University, Department of Computer Engineering, Büyükçekmece, İstanbul 34500 TURKEY (phone: 212-866-3300 x-5528; fax: 212-866-3412; e-mail: hakkoymaz@fatih.edu.tr).
İsmail Alan is with Fatih University, Department of Computer Engineering, Büyükçekmece, İstanbul 34500 TURKEY. (e-mail: ialan@fatih.edu.tr).

After these threats are materialized, the worst consequences are discontinuity of system services, and/or loss of system functionality and data. These in turn can have dramatic effects on life, health and environment for an organization. The fear of security breaches can lead organizations to decide to adopt a radical solution in such a way that "secure" protected private networks are physically separated from the public Internet. However, this would not be an ideal solution since the concept of a global Internet is compromised [8]. Instead, some kind of logical protection must be provided for information units during their transfer over the global network. In other words, an acceptable solution enables individual users or applications to communicate with each other over a global network in a secure manner. Security requirements of a system dictate a security policy to be employed in which for entities what actions to take are clearly defined. This research is about providing adaptive security policies against security threats [12]. We first investigate what is available and what improvements to make. We discuss security components of the network architecture with respect to Open System Interconnection reference model. Then, we look into management of security policy issues and clarify interactions between various security components. Finally, we devise and evaluate the configuration of an adaptive security policy.

Specifically, in the remainder of this first section, we discuss employing security solutions at different layers. Related work is given in section 2. Section 3 presents important details of Internet Protocol security services (IPSec) and how inbound and outbound packets get processed. Section 4 introduces our proposed adaptive security models for Security Policy Manager. Section 5 gives the conclusion and future work.

A. Security at What Level

Security in computer networks can be provided at application, transport or network layers of Open Systems Interconnection (OSI) reference model [7, 8]. At the application layer, an application that needs to transmit invaluable data over an insecure network can use internal mechanisms to achieve data integrity and confidentiality. In this case, the application programmer must implement and embed into his/her application all necessary security tools for transmitting data in a secure manner (i.e., secure channel). This extra effort involves expertise and adoption of tools in authentication, encryption and key exchange protocols on

behalf of the programmer. Other programmers working on other security-threat-prone applications would have to do the same thing for their applications from the scratch all over again. Usually, not all programmers are experts in implementing security protocols. Therefore, mistakes can be made in implementing security protocols for applications [3, 9]. At the transport layer, socket-based security implementations are common solutions to protect users' invaluable data from illegal accesses. Usually, applications are provided with enhanced sockets that encrypt data flowing through them. The main advantage of transport layer security over application layer security is that the applications are relieved from encrypting user data that need to be protected. For the application, the communication is secured using an enhanced socket and its interface [3]. The requirement of transport layer security mechanisms is that an application that needs data security must be modified to incorporate enhanced socket interface. At the network layer, Internet Protocol (IP) security provides the data confidentiality and integrity of its IP packets, regardless of how the application used the socket interface [2, 3, 5, 8, 9]. This means that any application can benefit from the underlying secure IP network, as long as it uses IP to send data. The main advantage is that applications are unaware of usage of IP security protocols. Thus, security services are provided to applications and users in a transparent fashion. Those real time and multicast applications that are based on connectionless user datagram protocol (UDP) can also benefit from IP security. Characteristic features that a security protocol provides at this layer can be listed as: (i) security is provided at the network layer and (ii) it is transparent to end users and applications. The disadvantage is that for high speed networking some performance loss can be observed due to packet processing overhead at IP layer [3].

II. Related Work

In recent years, number of protocols on information transfer has increased rapidly [2, 5, 6, 10, 11]. Speed and services of routers must be improved with new developments in protocols. Concept of "dynamically upgradeable router" is developed in [5]. Specifically, the design and implementation of modular and extended services router architecture are described for NetBSD OS kernel in software. *Routers* are the computer networking devices that buffers and forwards data packets toward their destinations through a packet scheduler. Routers perform packet processing operations based on the packet header information such as protocol type, destination address, source address and so on. This architecture in [5] allows dynamically addable plugins at run time. Plugins are code modules which implement a specific function of router. For example, security plugin is for providing security functions. Modular architecture also includes a packet classification algorithm. In addition to destination address based forwarding, state-of-the-art routers must provide new services such as integrated services, security algorithms, enhanced routing functionalities and selective packet droppings. The standard functions provided in this architecture are IP forwarding and routing, a packet scheduler, a packet classifier, security mechanisms, a firewall module and congestion control services. Another feature of this design is to classify packets into groups and apply different policies to different groups [5]. Difference of this with our work is that different application flows can be associated with different customized plugins in the former while the latter provides design of an adaptive security policy management for flows regardless of their types.

Another work is on preparation for a situation in which security threats get materialized. *Survivability* is defined as the ability of a system to continue to provide the adequate performance of its critical services and functions even after unforeseeable attacks have taken place and succeeded [1]. Providing survivability for a system can be difficult. Assuming that the security threat always exists for a system, security requirements must be defined for the system. These requirements must be classified into two categories, namely *essential services* and *non-essential services*. Essential services must be maintained even during successful attacks, non-essential services are to be recovered after intrusions have been dealt with [4]. This suggests developing systems that are autonomously reconfigurable, modular and adaptive. These features must be taken into account in existing systems for increased reliability in that as the security threats potential increases, system must increase its protection level and when threat potential decreases, system must decrease its protection level as well.

III. IP Level Security

To provide individual users and organizations with secure communications over global network, *Internet protocol* (IP), which can be viewed as common vehicle for various higher layer components, employs security features, called *IPsec*, as a service to all traffic using it [2, 3]. *IPsec services* are provided to protect (i) a path between a pair of end systems or hosts, (ii) a path between a pair of intermediate systems and (iii) a path between a host and intermediate system. *Intermediate system* can be defined as a system with a packet forwarding functionality (i.e., a router). IPsec has been developed for increasing network layer confidentiality and protection. Protection is of the form of data origin authentication, data integrity, replay detection, data confidentiality and access control [8]. These protections in IPsec are provided by means of following components:

a) two security protocols, namely *authentication header*(AH) and *encapsulating security payload*(ESP),
b) *security associations* (SA) on each IP path,
c) Encryption and authentication algorithms, discussions of which are beyond the scope of this paper.

Although these features are present at IPsec, policies governing the handling of inbound and outbound traffic to and from a host running the protocol are not well-defined and subject to improvement [2]. To illustrate, in packet filtering operation, a packet is checked first if it is protected or not. If it is not protected, what is the proper policy to handle such a packet? In addition, a packet under the scope of some security association (SA) may not be decapsulated because of incorrect keys. What is proper policy to handle such a packet? IPsec has

no clear answers to these questions. One way is to discard these packets, another to bypass the node to forward them to their final destination. We describe two different adaptive policies and give mechanisms for their implementations in section 4.

A. Packet Processing

IPsec is based on the concept of *datagram encapsulation* which means carrying IP packets across the network as cryptographically protected information units. This makes the encryption transparent to any intermediate nodes that must process packet headers for routing. In a secure network, there are certain operations on each network node for making sure that arriving packets and leaving packets are handled in such a way that packet protections are guaranteed end-to-end. For example, in IPsec transport mode, an authentication header (AH) consisting of security parameter index (SPI), sequence number (Seq No) and integrity check value (ICV) is appended to original IP datagram. This mode is intended for end-to-end protection implemented by source and destination hosts of original IP datagram. ICV is calculated by applying a secure hash function on IP datagram with following change: its mutable fields (i.e., hop-to-hop extensions) and authentication data field are set to zero before applying the hash function. Resulting value is recorded as ICV of IP datagram.

In tunnel mode, in addition to authentication header an encapsulating IP header which may have different source and destination addresses from those in original IP header are appended to original IP datagram. These new source and destination addresses refer to -*a secure path*- a fraction of end-to-end path between source and destination hosts of original IP header. *A secure path* between source and destination is defined with a structure called security association (SA). In other words, source and destination nodes (either end systems or intermediate systems) are connected through with a SA, providing a secure path between these two nodes. Important details of SA's are explained in next section.

B. Secure Connection

IPsec connection is described with a *Security Association (SA)*. An SA is a set of information consisting of security parameters that two network security endpoints agree upon in order to establish a secure communication. Major information contained in SA includes cryptographic keys, initialization vectors, digital certificates and so on for encryption, authentication and decryption algorithms in use for a secure channel at each endpoint. Each IP packet has a SA id. This id consists of packet destination address, protocol id and security parameters. It is used for accessing the related entry in the SA database at each host.

Fig. 1. A secure network where a box representing a security endpoint and a directed line representing an IPsec connection.

In Fig. 1, a secure path in a network is shown with a box representing a security endpoint and a line representing an IPsec connection. Basic packet processing involves checking first whether or not the packet is under the scope of some SA, which is used for performing network layer security check. How and when a new SA are created are explained in [12].

C. Inbound and Outbound Packet Processing

Policies related to handling of packet traffic entering and leaving a host which is running the IPsec protocol are described in this section. First, we give a flowchart that depicts how inbound and outbound packets are processed in Fig. 2.

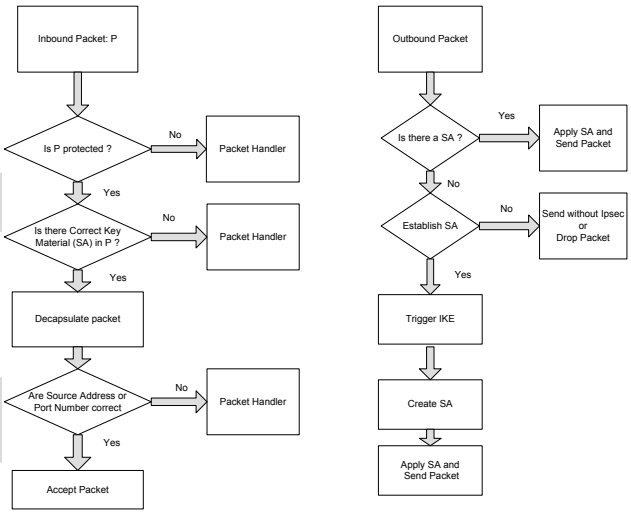

Fig. 2. Flowcharts for processing inbound and outbound packets in IPsec.

For an inbound packet arriving at a host from the network, the host goes through following steps [2]:

Step 1: *packet filtering*: check if the packet is protected or not. If it is protected, go to Step 2. If it is not protected, should it be accepted? This decision is left to network firewall.

Step 2: *protected packet handling*: for protected packets, either one of the following two cases are true

Case1: there is a correct key in the SA to decapsulate the packet. If so, decapsulate it. This does not mean that packet is acceptable. It may contain invalid source address or unauthorized destination port number. If that is the case, what should be done with it?

Case2: there is no correct key in the SA, thus unable to decapsulate the packet. If this is the case, what should be done with this packet?

For an outbound packet, similar decisions are made. These decisions are made per packet basis.

Step 1: *check SA*: check if there is an SA applicable to this packet? If there is only one SA applicable, apply it to the packet and send. If multiple applicable SAs exist, which one gets selected?

Step 2: *handling packet with no SA:* If no SA is applicable to the packet, what are the options available to handle such a packet? Some options are *forward* to some network interface, *drop, or queue* until an SA is made available.

Various options available and their ramifications are considered and explained in detail in terms of how and when they are employed in [11, 12].

These operations may decrease network data rates when applied per packet basis. To avoid such a slow-down, special care must be taken to avoid degradation into intolerably low level (i.e., it is important to setup threshold parameters t_1 and t_2 with proper values in the proposed models).

IV. Security Policy Manager

We design an adaptive security policy manager (SPM) in that as the frequency of attacks increase for a host in its responsibility area, security barrier (i.e., protection) for that host will be made higher; as more time passes without any attacks to the host, the security barrier for that host will be lowered. As shown in Fig. 3, SPM has a responsibility area consisting of N hosts. Let G denote the set of these hosts in the responsibility area of SPM (i.e., $|G| = N$, host h_2 in G, and host h_1 not in G).

A. Security Model 1: A Statefull SPM

In this particular configuration, in case any unprotected packet arrives with a destination not in G, that packet will be delivered to packet scheduler for routing to its destination bypassing SPM. Otherwise, if an unprotected packet arrives at SPM for a host in G, there could be further processing at SPM with related to this packet based on feedback from destination host in G. In this model, a host provides a feedback to SPM

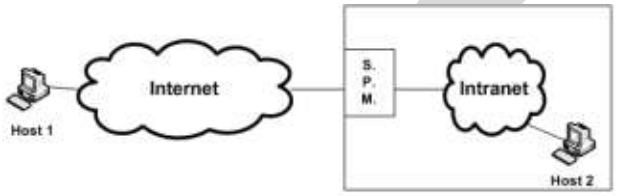

Fig. 3. SPM environment for both security model 1 and security model 2. Host 1 is source, Host 2 is destination.

when a faulty packet arrives

(i.e., we assume that each host has a means to determine a faulty packet from the rest when they are not protected) and SPM will make a note of the source of the faulty packet in the list maintained for this particular host. Also bad-packet-count for this particular host will be incremented. SPM maintains two pieces of information for each host h_i in G: (a) bad-packet-count to keep track of the severity of the attacks to host h_i, (b) a list of host addresses of suspected sources (i.e., we call it suspect hosts). Consider a host h_i in responsibility area of SPM. **Suspect hosts** for h_i (i.e., denoted by SH_i) would be classified as own suspects (i.e., OS_i) and imported suspects (i.e., IS_i). OS_i are those source hosts that send bad packets to destination h_i. IS_i are those suspects that have sent bad packets to those hosts h_j in G other than host h_i (i.e., h_j in G, $1 \le j \le N$ and $i \ne j$). Note that IS_i = union of OS_j's where $1 \le j \le N$ and $i \ne j$. Suspect hosts list SH_i grows as the new attacks take place for host h_i. It would include source addresses of those bad packets addressed to h_i. Although potentially any host on Internet can send bad packets to a host (thus, their source addresses would be added to the list), not all of them will appear in the suspect hosts list. In fact, list size due to own suspects are bounded by threshold t_1 (i.e., $|OS_i| \le t_1$). In some extreme cases, the list size can grow to a maximum of $N*t_1$ since there can be a maximum of t_1 many number of h_i's own suspects, and $(N-1)*t_1$ many numbers of imported suspects. However, since multiple suspect lists may contain same entries (i.e., SH_i and OS_i are not disjoint sets), we anticipate that the size of the list SH_i for each host h_i to be much smaller. If enough time passes without a security incident (call this interval **recovery period**), then security barrier will be lowered to reflect this change in the environment. More specifically, as the time passes without any bad packet addressing a particular host h_i (after a recovery period), first, those safe sources (i.e., hosts not in SH_i) are allowed to communicate with h_i. As more time passes without a security incident (after a second recovery period) for host h_i, then those sources that are suspects to other hosts (i.e., imported suspects or hosts in IS_i) would be allowed to communicate with h_i. Another two recovery periods without a security incident would clear the entire list of suspect sources and therefore SH_i becomes empty for host h_i. Fig. 4 shows pseudo code of this security model.

Security Model 1: SPM runs the following steps on arrival of a faulty packet for host h_i. Assume that packet is not protected (no SA, no encryption); t_1, t_2 are thresholds ($t_2 > t_1$)

```
Increment bad_packet_count
Tag source as blocked (add to OSi)
If bad_packet_count > t1
   Tag all others' suspect sources as
   blocked (ISi=union OSj's, 1≤j≤N, i≠j)
else if bad_packet_count > t2
   Tag all sources as blocked (deny all)
   Start a recovery time interval
If no bad packet arrives for a time
interval ≥ 4 * recovery period
   Remove tags from own suspects(in OSi)
   Drop bad_packet_count to 0.
else if no bad packet arrives for a time
interval ≥ 2*recovery period
   Remove tags from those hosts in ISi
   Drop bad_packet_count to t1/2
else if no bad packet arrives for a time
interval ≥ recovery period
   Remove tags from those safe sources
   (hosts in SHi are still denied)
   Drop bad_packet_count to t1+(t2-t1)/2
```

Fig. 4. Pseudocode of adaptive security model 1.

B. Observations on Security Model 1

One can make following observation after investigating this algorithm for Security Model 1. As bad packets arrive, suspect list size will grow; however, number of suspects in the suspect list is bounded by $N*t_1$ for each host. Fig. 5 illustrates algorithm's behavior with respect to bad-packet-count versus list size. Following observation can also be made after

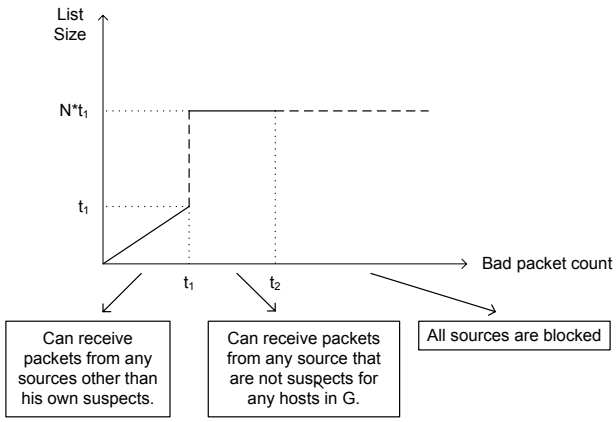

Fig. 5. Growth of host h_i's list of suspect sources as security attacks taking place for h_i.

investigating this algorithm. As the time passes without bad packets for host h_i security barrier will be lowered and system will accept packets from ever growing set of sources gradually. Fig. 6 illustrates algorithm's behavior with respect to bad-packet-count versus time passing without security incidents.

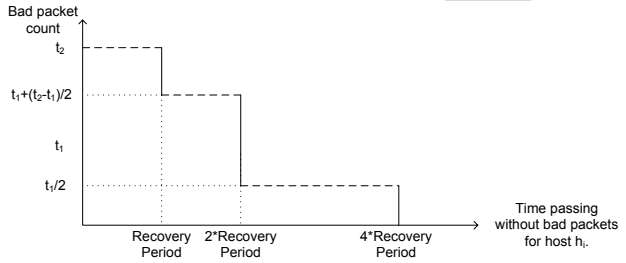

Fig. 6. Resetting bad packet count back to 0 as time passes without a security attacks.

C. Security Model 2: A Stateless SPM

As a second scenario, a host on the Internet may make an SA with the host through SPM. For this scenario we are proposing another algorithm that is slightly different from the first one. Different processing is explained in Fig. 7. Difference between security model 1 and security model 2 is that the former deals with the unprotected packets while the latter deals with the protected packets (i.e., exchanged under the scope of an SA). For the protected packets, SPM does not need a feedback from the destination host in its G for determining if the packet security is breached since enough information is already available in SA to check for it. Furthermore, during the interval between bad-packet-count of 0 and threshold t_1, SPM will only tag source for host h_i as blocked, without removing the SA. This is so since the SA creation is costly procedure, and if threats disappear in the near future, the source and destination hosts can communicate under the scope of this SA. However, if security threats persist (i.e., number of faulty packets exceeds threshold t_1), from then on, all sources in the bad packets not only get blocked, but also their SA's with SPM are removed. If threats continue to persist even more (i.e., number of faulty packets exceeds threshold t_2), those SA's unremoved during the initial stage (i.e., time interval of bad-packet-count of 0 and threshold t_1) will be removed as well. The way for the security model 2 to get out and to recover from a security attack is same as that of security model 1.

Security Model 2: SPM runs the following steps on arrival of a faulty protected packet (under scope of SA) for host h_i. Assume that packet is encapsulated under an SA (SA exists, but, its keys are incorrect); t_1 and t_2 are thresholds, $t_1<t_2$.

```
Increment bad_packet_count
Tag source as blocked (add to OSi)
If bad_packet_count > t1
   Drop SA between this source & dest hi
   Tag others' suspect sources as
   blocked(ISi=union of OSj's, 1≤j≤N,i≠j)
else if bad_packet_count > t2
   Drop SAs between hosts in OSi & dest
   Tag all the sources as blocked
   Start a recovery time interval
If no bad packet arrives for a time
interval ≥ 4*recovery period
   Remove tags from hosts in OSi
   Drop bad_packet_count to 0
else if no bad packet arrives for a time
interval ≥ 2*recovery period
   Remove tags from hosts in ISi
   Drop bad_packet_count to t1/2
else if no bad packet arrives for a time
interval ≥ recovery period
   Remove tags from hosts not in SHi
   Drop bad-packet-count to t1+(t2-t1)/2
```

Fig. 7. Pseudocode of adaptive security model 2.

D. Observations on Security Model 2

In this model, just like the previous one, the list size for a host h_i grows linear as faulty packets arrive at the host until threshold t_1. The behavior of model 2 suggests that after threshold t_1 the list size becomes $N*t_1$ again and remains the same from then on until the first recovery period. Fig. 8 shows how the security model 2 behaves with respect to the SA removal. As depicted, between bad_packet_count of 0 and t_1 no SA's are removed. Between t_1 and t_2 total number of SA's dropped grows linear with the number of bad-packets arrived (i.e., at t_2, number of SA's dropped is $t_2–t_1$). After t_2, number of SA's dropped gets to a maximum of t_2 as the algorithm has determined that it is under attack.

V. Conclusion and Future Work

In this paper, we have investigated the security aspects of computer networks architecture and observed that security services are provided at application, transport and network layers of OSI model. Network layer security presents some important opportunities for offering controls over security threats to distributed information systems connected to the global Internet. We have looked at the important details of Internet Protocol (IP) security services and proposed

enhancements over handling both protected and unprotected packet traffics for a dynamic environment. Dynamism is in terms of severity of security threats. Our proposal includes the security policy manager (SPM), an adaptive security control mechanism that would automatically configure security levels in response to the frequency of security attacks. In particular, when security attacks become more frequent, SPM increases the security level by blocking interaction with the packet traffics suspected to contain security risks. When the time shows that the attacks are no longer occurring, SPM decreases the security level by gradually allowing interaction with the packet traffics starting with the least-likely towards the most-likely to contain security threats. As for future works, although empirical evaluation seems promising and related research papers [5, 11] suggests a notable performance increases with a fast hash table implementation, how much processing overhead this adaptive policy and its implementation mechanisms bring about must be investigated further.

We are currently conducting simulations to observe the interaction of the various components of SPM, to develop interfaces for each of these components and to monitor the packet processing for both inbound and outbound packets. We hope to find out more about ideal threshold values for each of the algorithms presented.

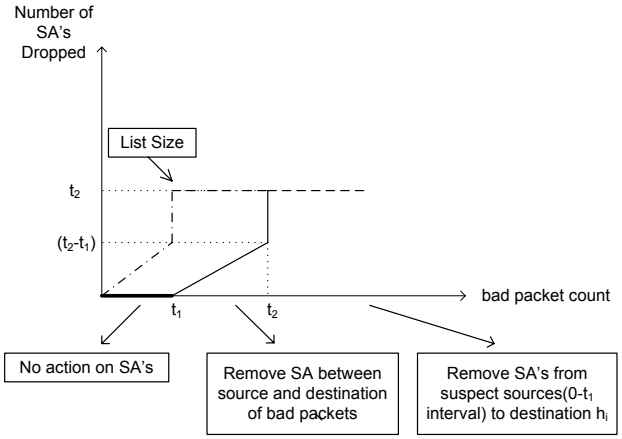

Fig. 8. Growth of host h_i's list of suspect sources and removal of SA's as security attacks taking place for h_i.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their invaluable comments which caused us to express our ideas in more precise manners and made this paper more readable.

REFERENCES

[1] K. Kyamakya, K. Jobmann, and M. Meincke, "Security and Survivability of Distributed Systems", IEEE, 2000.
[2] M. Blaze, J. Ioannidis and A. Keromytis, "Trust Management for IPsec", ACM Transactions on Information and System Security v. 5, no.2, May 2002.
[3] G. Insolvibile, "The IP Security Protocol", LINUX Journal, September 2002.
[4] A. Avizienis, "Toward Systematic Design of Fault-Tolerant Systems", IEEE, 1997.
[5] D. Descaper, Z. Dittia, G. Parulkar and B. Plattner, "Router Plugins: a Software Architecture for Next-Generation Routers", IEEE, 2000.
[6] W. Stallings, *Cryptography and Network Security, 4th ed.,* Prentice Hall, 2006.
[7] A. S. Tanenbaum, *Computer Networks, 4th ed.,* Prentice Hall, 2002.
[8] R. Molva, "Internet Security Architecture", Computer Networks & ISDN Systems Journal, v.31, no.8, April 1999.
[9] R. Oppliger, "Security at Internet Layer", Computer, Vol.31, No.9, pp.43-47, September 1998.
[10] A.S. Tanenbaum, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2002.
[11] R. Morris, "The click modular router", 17th ACM Sym. on Operating Sys. Principles, Dec 1999.
[12] Ismail Alan, "Trust management policies for distributed computing systems," M.S. thesis, Dept. of Computer Engineering, Fatih University, Istanbul, Turkey, July 2007.