ABSTRACT

HE HUANG. On the Protection of Link State Routing and Discovery of PKI Certificate Chains in MANET (Under the direction of Shyhtsun Felix Wu)

The growing awareness of the network vulnerability draws much attention to security from both the academic community and industrial society. Security is no longer a luxury but an independent and indispensable service to the current Internet. While various security mechanisms such as cryptographic and intrusion detection techniques have been proposed, designed, and even deployed in the field, the newly exposed network vulnerabilities and the emerging network technologies create new security challenges which make the existing security solutions either inefficient or insufficient. My Ph D research focuses on the efficient protection on link state routing and the self-organizing and self-dependent hierarchical public key certificate management in the emerging mobile ad hoc networks. The contributions of this thesis include two parts. In the first part, a cost reduced secure link state routing protocol with the capability of detecting disruptive links is proposed to efficiently protect the routing control messages (e.g., LSA) and trace the faulty intermediate routers; then a confidence extended routing mechanism enhanced with secure virtual links is designed to increase network reachability through selectively including the uncertain routers in packet relaying and further continuously monitoring the behaviors of those selected uncertain routers. A theoretical security analysis and an experimental evaluation are conducted to prove the feasibility and advantages of this new design under various rates of false alarms. In the second part, an approach is presented to discover the optimal PKI certificate path even without help from centralized certificate entities in the non-centralized and infrastructureless mobile ad hoc

network and a secured and distributed certificate-chain searching protocol is developed to collect the needed certificates on the fly in the mobile ad hoc network.

On the Protection of Link State Routing and Discovery of PKI Certificate Chain in MANET

By

He Huang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree Of
Doctor of Philosophy

Computer Science

Raleigh
2005

APPROVED BY

Dr. Peng Ning	Dr. Rudra Dutta
Dr Arne A. Nilsson Chair of Advisory Committee	Dr. Shyhtsun Felix Wu Co-Chair of Advisory Committee

To my parents: Zaikang Huang and Shuying Xu

my wife: Ying Yu

and our children: Liz and Eddie

Biography

He Huang was born in Fujian province, China. He received his B. E. degree in the Computer Science and Technology Department at Xi'an Jiaotong University in Xi'an, Shannxi, China, in July, 1991. He was a graduate student in the Computer Science Department at Fudan University in Shanghai, China before he entered the graduate school in North Carolina State University. He received his M.S degree from the Computer Science Department at North Carolina State University in 1999.

Later, he enrolled into the Ph D program in the Computer Science Department at North Carolina State University as a part-time student while he worked as a full-time employee in Nortel in RTP, NC. His current research interests are in network security, key management in wireless mobile network, and VoIP security.

ACKNOWLEDGEMENTS

A Ph.D. study is a long journey mixed with frustration and joy. I would have not made it without the support and help from these people.

I'm grateful to Dr Felix Wu for guiding me through the entire journey. He taught me the knowledge of the network security, directed me in the essential research principles, advised me toward the right research directions, and encouraged me during my research downside. The research results in this thesis are the fruits born from numerous email discussions and long-distant phone calls with him.

I would like to sincerely thank Dr. Arne Nilsson, Dr. Peng Ning, and Dr. Rudra Dutta as my thesis committee members for their continuous support and constructive comments on my research work. Their constructive criticism and insightful comments in my preliminary exam helped quietly in my late research.

Many thanks to Graduate Director, Dr. Edward Davis, Dr Theutente, and staff member Ms. Bishop in the ECE department and others in the computer science department for their help in my course registration, course grading and student status maintenance during these years. I also want to thanks Dr. Ming Kan in the Math Department for the help in Statistics.

I want to give special thanks to my managers Kevin Klinge and Robert Scheible for their consistent support in my Ph. D. study even during the difficult time and my employer Nortel for its financial aid in my tuition and travel fees for paper presentations.

I also want to thank my colleagues from SHANG lab, Xiaoliang Zhao, Zhi Fu, Cheunglong Wu, Hoyen Chang, Jim Yuill, and many others. I enjoyed the time when we worked together in the lab. Even if they all graduated from school, their constant encouragement and sincere help meant a lot during that time.

I am indebted to my family for their love and support. My parents always taught me the importance of getting educated. Their unlimited love, lifelong instructing, and constant encouragement always remind me of not-giving up in the middle. I thank my beloved wife, Ying Yu, for your understanding, support, and patience during the difficult time and my two lovely children, Liz and Eddie for your understanding of my excuses without playtime and story time. I dedicate this dissertation to them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

During recent years, we have witnessed the tremendous development in computer networking technology and the explosive growth of the Internet. The network is becoming indispensable to many aspects of our daily life. It dramatically helps us in information sharing and improves our life quality. However, with the growth of the Internet, the rising number of security incidents and their explicit and potential impacts are gradually becoming of much concern. The Morris Internet worm [1] in 1988, the Mellisa Macro Virus in 1997 [2], and the Smurf denial of service attack [3] in 1997 are several of the early examples best demonstrating how vulnerable the networks are and how important security is for the networks. The attacks to the networks may lead to unauthorized access, information leakage, illegal resource occupation, and service unavailability. The awareness of the vulnerability of the networks has already gained much attention from the Internet society, and variant security mechanisms have been proposed, designed, and even deployed in the field to protect the network against the possible attacks since then. For example, cryptographic mechanisms such as public key scheme or symmetric key are used to ensure the confidentiality, integrity, and authentication of the data in the transmit. The public key infrastructure (PKI) helps establish the trust relation between the unknown communication parties in the network; the intrusion detection forms a second line of defense against the intruders sneaking into the network. However, the present security solutions are not a panacea to either prevent

the emerging network attacks or protect the newly introduced applications such as wireless network applications. Recent security reports [69] [73] [74] show that more and more sophisticated forms of network attacks have been created to exploit new network vulnerabilities, system misconfiguration, and network misoperation. For instance, in June 2002, Code Red virus [69], exploiting the buffer-overflow vulnerability in HTTP requests, infected 359,000 hosts in a short time and led to the service outage in a large number of servers running Microsoft NT/2000. Besides, the continuously increasing computational power in individual personal computers and the increasingly improving capability of collaborating the computing efforts from the hundreds and even thousands of individual computers also grant the adversaries more power than ever to generate attacks. For example, with such high-speed computer the brute-force break of the weak password becomes possible, and it is not very difficult now to generate high volumes of unfaltering or untraceable junk traffic for denial of service. Further, new merging wireless network models also bring new security challenges. This thesis concentrates on the security challenges posed by link state routing and the wireless mobile ad hoc network.

Link state routing protocol (e.g., OSPF and IS-IS) has been widely deployed in the network for path determination within packet relaying. The importance of the routers in the network and the vulnerability in the nature of the link state routing protocol highlight the necessity of effective routing protection against any possible attacks. One of the influential attacks come from the faulty intermediate routers (FIR) [70] which intentionally compromise the LSA information passing by and pollute the routing tables in its downstream routers. Since all LSAs have to pass more than one intermediate router

in order to reach other routers in the network, such attacks not only cause more impacts on the network but are also not easily detected. While presently some security mechanisms have been proposed and developed to protect the link state routing protocol, they expose some limitations in terms of high expense and explicit vulnerability so that they are unacceptable to prevent the network from the FIR attacks. Thus, how to efficiently secure the link state routing against the FIRs becomes the challenge to network security. In this thesis, we propose a novel enhanced link state routing mechanism combining security and intrusion detection to address the network attacks from the FIRs.

With the complicated network circumstance and developing intrusion detection technologies, some routers may escape conviction detection as false routers or be cleared as innocent immediately. How to treat those uncertain routers, called routers under monitor (RUM), becomes one challenging issue [71]. On the one hand, if we treat the RUM the same as the trusted routers to relay the packets without extra caution, it may increase the risk of service disruptions such as packet dropping or packet corruption. On the other hand, totally excluding the RUMs from the network also leads to negative effects, especially in the sparsely connected and resource scarce network systems. For example, the blocking of traffic over those falsely accused routers may somehow result in the network partition and disrupt the network reachability. To address the above issue, we propose to determine the path with the consideration of the routers' confidence and build a secure virtual link between two trusted routers across those selected RUMs. By doing that, we achieve our goal of increasing the network reachability without sacrificing network security.

While the newly emerging network threats require new security solutions, the lately introduced network technologies also impose new security challenges. Mobile Ad Hoc Networks (MANET) are those networks which consist of a collection of wireless mobile hosts and are formed on the fly without the aid of any pre-established infrastructure and centralized administration [4] [5] [6]. MANET brings new, challenging issues as well. One of the challenges is how to conduct public key certificate validation in MANET. Public key certificates prove validity and authenticity of their ownership and possibly other properties. Certificate path discovery is the critical process for public key verification in hierarchical public key infrastructure (PKI) diagrams. This process is conventionally done in centralized public key management system such as a central CA or directory. However, in an infrastructure-less environment, such as a mobile ad hoc network, no such central service is present due to network partition. That brings the challenges for public key verification. Some researchers have been exploring public key management in ad hoc networks, but none of their studies are based on the hierarchical public key trust model. In this thesis, a scheme [72] is presented to represent each CA certificate with a coded certificate path label and to design an algorithm to speed up the process of certificate path discovery without the presence of a central PKI service. Furthermore, an on-demand protocol is proposed to collect the certificates in networks on the fly.

The rest of this thesis is organized as follows. In chapter 2, we present the related work. In chapter 3, we develop an efficient authentication approach to protecting the link state routing and detecting the disruptive links. In chapter 4, we propose a new routing mechanism with the combination of confidence routing and secure virtual links to

securely and reliably leverage the resources from the RUMs and increase the network reachability. In chapter 5, we propose an efficient certificate chain discovery approach in the mobile ad hoc environment and design a self-organizing protocol to collect the needed certificates on the fly in non-centralized and infrastructureless mobile ad hoc networks. In chapter 6, we conclude the thesis and suggest some future work.

Chapter 2

Related Work

In this section, we will review the academic or industrial work related with my research focuses.

2.1 Secure Link State Routing

Recently much research has been done on securing link state routing against malicious operations and discovering the faulty origin after the violation of the routing and forwarding services.

Because compromising the routing protocol could lead to catastrophic consequences in the nework, such as network performance downgrade, or even global service disabled, in the network, more research attention has been paid on securing routing packets against the compromise in the middle and discovering faulty origin after the security violation on the link state routing.

The earliest security work on link state routing protocol can be traced to Permlan's seminal work [48]. In her PhD thesis [48], Permlan proposed a robust flooding routing mechanism under the Bayzantine environment. Each packet is signed with the originator's private key. To guarantee such robustness, some cost must be paid, mainly due to the expenses from the public key security, as described in [53].

Murphy et al [49] design and implement the digital signature based protection for OSPF. The identification and the sensitive fields in routing packets are authenticated with the source router's private key prior to being sent out. However, recognizing the high

expense with the digital signature in each routing packet, they suggested either adding the extra hardware or delaying the verification of the signature in LSA messages in order to reduce the cost.

Addressing the cost of public key security in link state routing, Hauser, et al[50] present an efficient authentication mechanism with a one-way hash chain function which was originally developed by Merkle and Lamport [56]. By hashing the time and a secret number, each router generates two distinct hash chains with n distinct values for two link states, UP and DOWN, separately. The first LSA is the last hash value in the chains signed with the originator's private key. Without knowing the secret number, another party cannot derive the rest of the hash values in the chains. However, the subsequent hash values representing the latest link state from the originator can be verified with the previously received hash value and the synchronized time interval. This mechanism is able to reduce the computation cost in verification by an order of magnitude cheaper, but as mentioned by the author, some limitations such as authentication for multiple-values link remain.

To solve the limitations mentioned in [50], Cheung [47] proposed an improved hash-chain authentication approach to efficiently protect the link state routing messages. After detecting the compromise, a bad routing update advertisement (BURA) including one bogus LSA with the smallest sequence number or largest checksum (s) is generated and signed with its private key, then flooded into the entire network. Each router will form a bad LSA propagation graph with the collected BURAs and run a DFS algorithm to search the faulty routers.

On the other hand, instead of attacking the routing protocol, the faulty routers, especially located in the critical path of the network topology, could misoperate the data packets passing by, e.g., dropping the packets. Some researchers proposed probing approaches which intend to identify the faulty origin. Traceroute is a popular network tool to check the reachability between the source and the destination, but it is vulnerable to malicious attacks, such as selectively drop. Padmanabhan, et al [67] presents a secure traceroute to achieve the same goal as Traceroute without being compromised by the selectively drop attack. However, the common limitations on the probing mechanisms are its impractibility due to the existence of multiple routes between the source and the destination and the difficulty in detecting occasional violation of forwarding policy. The other approach is to monitor and analyze the traffic over the network. Bradley, et al [51] propose a distributed monitoring mechanism, WATCHER, where flow counters in each router are created to record the traffic from each router through each of its neighboring routers. Within the validation phase, each router can detect the packet dropping or misroute through comparing with the counters from its neighboring routers. If the discrepancy is detected with the router's neighboring router(s), a diagnostic process further is initiated to discover the faulty origin. However, WATCHER suffers variant limitation as detailed in [53]. More recently, Mizrak, et al [54] formalize the specification of the traffic analysis mechanism and propose two detection protocols with the difference in accuracy, completeness, and overhead. Though showing some advantages over WATCHER, such as cost reduction of monitoring storage in each router, it is unclear how the detection protocols countermeasure the compromised routers in the network.

2.2 Overlay Networking with Virtual Link

The idea to achieve reliable and assured service from the network by using an overlay network model is not a new. Concerned about inefficient performance in the Internet, recently quite a few researchers have proposed the more end user–controllable approaches in which the sources deliver the traffic over the predetermined path instead of relaying on the slow-recovery and unpredictable Internet routing. Ericksson in [60] introduced a MBONE which transfers the multicast traffic via the virtual tunnels between different network daemons in the separate subnets. MBONE facilitates the multicast service transfer in the large Internet network but does not provide reliability for the virtual link and security against the attacks in the middle. Addressing the inefficient routing and transportation protocols in the modern Internet, Savage, et al introduced Detour architecture [61] to tunnel encapsulated IP packets in the routers. Detour is able to support the exchange of real link metrics of the tunnels among the detour routers and provide multi-path routing in place of the traditional inefficient routing strategies; besides, the detour routers are suggested to share more network capacity information with the hosts to avoid slow start or traffic burst from TCP application in those hosts. More recently, D. Andersen, et al proposed a resilient overlay network [59] to discover the good path between any peer of network routers and quickly detect the faulty path by the aggressive probing mechanism and path performance monitoring in the application layer between the communication peers. All approaches assume a fully trusted network environment where all routers honestly provide all the link metric information and behave as expected. Thus, if some compromised routers get involved, they may

downgrade the performance of such overlay networks, even make such service unavailable.

Considering the security issue of transferring data over an untrusted public network, the virtual private network (VPN) solution applies the cryptography in end users to provide data confidence, authentication, and integrity. While the VPN solution is able to provide the end to end security of data transportation, it is vulnerable to the security attacks in the middle, for example, packet dropping and artificial delay, which is a typical behavior of the compromised faulty intermediate router (FIR). VPN relies one the existing routing infrastructure to relay the cryptographic data; thus, it may fail if the network does not prevent the compromised FIR being selected in the path.

Keromytis, et al [62] proposed secure overlay services (SOS) architecture to prevent the denial of service attacks. In SOS architecture, a secure overlay access point (SOAP) must authenticate the legitimate user and transfer the request into a beacon which acts a relaying router and is periodically informed of the location of the secure serverlet. Only the secure serverlet is allowed to transfer the packet into the target region. Before the request reaches the target server, a filter mechanism between them is installed to drop the illegitimate packet and prevent overwhelming the routing and resource at or near the server. One major feature in SOS is the redundancy; thus if one SOS service component gets attacked, the other service components will take over providing the service. SOS also embraces the randomness and anonymity in the architecture to increase the difficulty of targeting the server and the user traffic in the attackers. The author notices it is future work to select a path between the SOAP and the target server in the untrusted overlay network. No reliability about the data transfer inside the overlay network is discussed.

2.3 Decentralize Global Trust in PKI Infrastructure

In conventional centralized PKI, the problem of certificate path discovery can be easily solved by searching the complete CA hierarchy in central devoted entities such as a directory with popular graph searching algorithms such as DFS. However, MANET gives rise to a dynamic network diagram which requires the need of validating certificates in the absence of centralized public key infrastructure.

There are some works in the decentralization of global trust. Blaze and Feigenbaum in [8] proposed a keynote framework which reduces the dependence on a centralized trust management system by binding both public key and policy together in certificates. Each relying node is equipped with a policyMaker engine which is able to determine the visitor's capability such as access permission locally based on its certificate. However, their proposal still relied on centralized public key infrastructure for public-key verification. Gunter and Jim [36] introduced QCM to retrieve the relevant certificates based on the policy language. If the verifier's local certificate database does not contain the right certificate, QCM will drive a remote query from the policy in the certificate and send it to the issuer for membership checking. Their proposal required that the certificate issuer centrally store the certificates and handle the membership query. As we discussed in section 1, the central model is not feasible in an ad hoc environment. Li and Winsborough [37] proposed a role based trust management language to represent the credential and further introduced algorithms to discover the credential chain even in the distributed credential storages. However, those algorithms require the complete

information of the intermediate nodes between the issuer and subjects, That is impractical in ad hoc networks as well.

Zhou and Hass in [9] adopted the threshold cryptograph for distributed public key management in ad hoc networks. The essence of their solution is to divide a key pair (K/k)of security service into n shares, (K_1/k_1), (K_2/k_2), …, (K_n/k_n), one share per server. These n security servers collaborate for security services such as certificate signature and key refreshness. By doing that, it prevents such security vulnerability as a single failure CA point which may generate erroneous certificates. But, neither did this solution provide the mechanisms for certificate validation without relying on a centralized certificate directory.

In [12], Capkun and Buttyan proposed a fully self-organized public key management system in the ad hoc situation. Their essential idea is that each individual node acts as CA to issue certificates to other trusted nodes and stores the issued or issuing certificates locally; certificate path is discovered by combining the local stored certificate information. Their public key management relies on the local certificate repository for key authentication. However, it is unclear that how to cooperate with neighboring nodes for a complete certificate chain if two communication parties cannot build a certificate path with their locally caching certificates.

Chapter 3

Cost Reduced Secure Link State Routing

3.1 Introduction

Link state routing protocol (e.g., OSPF and IS-IS) has been widely deployed in networks for path determination within packet relaying. Generally, the link state advertisement (LSA) message is broadcast by each router to inform other routers in the network of its neighboring routers and the connectivity situation such as connection status, link delay, et al. A network topology is formed based on the collected LSAs, and the optimal paths to the reachable destinations are computed based on this network topology. The importance of the routers in the network and the vulnerability in the nature of the link state routing protocol highlight the necessity of the effective routing protection against any possible attacks. Mainly there are two types of network attacks, external attack and insider attack [52]. The external attack from the non-legitimate routers can be easily prevented by the cryptographic mechanism and thus will not be discussed further in this chapter. The inside attack comes from a legitimate but misbehaving network participant/router, called a faulty router. A router could become faulty due to the software defection [41], or be subverted because of the weak passwords and the latent software vulnerability [42][43]. The faulty routers can attack the network in various aspects such as packet dropping, delay, reorder, alteration, denial of service, et al. to degrade the network performance, even make network service unavailable. Here we focus on the attack from the faulty routers toward the link state routing protocol such as OSPF.

Categorized with the origination of the LSAs in the network, mainly there are two types of faulty routers. The first one is the faulty source router which generates the LSAs with fake link information such as an incorrectly shorter link delay to attract more traffic and then conducts the blackhole attack by dropping all packets passing by. The second one is the faulty intermediate router (FIR) which intentionally compromises the LSA messages passing by and pollutes the routing tables in its downstream routers. Since all LSAs have to pass more than one intermediate router in order to reach other routers in the network, such attacks not only cause more impact on the network but are also not easily detected. In this chapter, we focus on the solution to detect and trace the FIR attacks.

While a few security mechanisms have been proposed and developed to protect the link state routing protocol, they expose some limitations. Though the digital signature solution , in which the originating router authenticates the LSAs with its private key and the others verify the LSAs with the originating router's public key, is able to prevent such attacks, the expensive computation [52] prevents its wide adoption. The IETF recommends shared Key-MD5 in OSPF v4 [44] and IPsec protection between the neighbors in OSPF v6 [45]. The cost in authentication and verification on the LSAs is reduced, but those mechanisms are unable to prevent FIRs from attacking the routing packets passing through. Thus, how to efficiently secure the link state routing against the FIRs becomes the challenge to the network security.

In this chapter, we propose a novel efficient secure link state routing mechanism combining security and intrusion detection to address the network attacks from the FIRs. The significant property of our approach is the **detectability** of the abnormal behavior toward the LSA and the **traceability** of the FIR generating those bogus LSA.

Specifically, we first introduce an efficient authentication mechanism – credit based authentication, by which the originating router authenticates its $\boldsymbol{M}$ subsequent LSAs with a session key without prior key exchange; if bogus LSAs are detected with the later disclosed session key, we propose two FIR tracing schemes to trace back the malicious FIRs through the analysis of historical LSA logs from the affected routers which detect the receiving of the bogus LSAs. Our proposal is capable of detecting the multiple FIRs which may conduct the manipulation of the LSAs from multiple originators in different time, narrowing the location of the faulty intermediate routers, isolating and eventually removing the faulty intermediate routers in order to achieve the purpose of the routing security.

Our contributions in this chapter are as follows. First, a novel authentication mechanism is introduced to efficiently protect the LSAs and ensure that the malicious manipulation of the LSAs from the FIRs is detectable and the faulty origin is traceable. Second, we develop two practical tracing schemes with the difference in system environments, completeness, and communication cost to trace back the FIRs based on the log information. Moreover, the confidence values are computed from the schemes as the measurement of link disruptiveness in order to reduce the false alarms.

The rest of the chapter is organized as follows. In section 3.2, we present the system models. In section 3.3, we introduce the mechanism of credit based authentication, then propose two tracing schemes to search the FIRs. In section 3.4, we conduct the analysis of the memory requirement and the communication cost. Finally, we conclude this chapter in section 3.5.

3.2 System Models

Network Model This work considers networks consisting of routers running link state routing protocols, such as OSPF. A network is described with an undirected graph G(V, E), where V is the set of the routers in the network and the E is the set of links in the network. The following assumptions are made: (1) Each router holds a pair of keys, one public key and one private key. A public key distribution mechanism has been deployed so that each legitimate router's public key is known to other routers in the network. (2) To avoid the outsider attack which is not authorized for routing operation, we assume that neighboring routers use IPsec with AUTH option to prevent receiving routing packets from the unauthorized / rogue routers. (3) Each router has the capability of generating a random number. The sequence of the generated random numbers from one router must be statically independent and unbiased such that the disclosed one can not predict the later one. The random number is used as the session key to authenticate the router's own LSA. We will discuss its usage in the next section. (4) Assume there exists a network timer with which each router can synchronize its local clock and adjust its skew in time.

Threat Models In our discussion, we focus on the threats from the FIRs against the routing protocol. FIR can attack the routing packets passing through at will. It may drop, delay, reorder, alter the routing packets sent by other legitimate routers, or masquerade another router to insert fake routing packets, or flood the network with excessive routing packets. More than one FIR may exist in the network simultaneously, and those FIRs could behave independently or collusively. We assume that the location of FIR does not lead to network partition. Otherwise, if the FIRs gain all control for the routing between the separate network partitions, the schemes proposed in the chapter will not work. We

assume at least a good path with no FIR existing between any pair of routers in the network.

Notation Ks is denoted as a session key. To simplify our discussion, link state advertisement (LSA) is used to represent all types of routing packets in this chapter. One LSA can be represented concisely with such important components, (*Id, Sq, T, Pl, MAC*), where *Id* is the identity of the originating router; *Sq* is the monotonically increased sequence number; *T* is the timestamp for this routing packet; *Pl* is the packet content; and *MAC* is the message authentication code of the routing packets. Besides, MAC_k is denoted as the authentication with the session key, and Sg_{pri} is denoted as the signature with the private key. We use H(s) to denote a message authentication code (e.g., MD5) of message s.

Definition 1: We call a link a ***disruptive link*** if it is connected with a FIR; otherwise, we call it a ***normal link***.

A FIR always manipulates the routing packets passing through; thus, a disruptive link indicates where the LSA manipulate happened. Through identifying the disruptive link, we can narrow the location of the FIRs.

Design Goal Our goal is to develop an efficient authentication mechanism for link state routing protocol and trace the FIRs by identifying the disruptive links. Though the FIRs may not be exactly marked, we argue that the removal of the disruptive links will reduce the impact from the FIR and eventually isolate the FIRs from the network.

3.3 Efficient Authentication and Fault Tracing

3.3.1 Efficient Credit Based Authentication

Due to the high expense in using the public key scheme, the symmetric key scheme is used to efficiently authenticate the LSAs in our proposal. The prior key exchange usually being required in the symmetric key scheme shows some security vulnerability and inefficiency in link state routing. If a session key is known within the communication between the originator and all other routers, the FIR can impersonate the originator without being detected. If the session key is shared between the originator and each individual router, it will require a large cost in both key storage and communication. For example, the originator has to make a copy of the LSAs with a separate signature for every router in the network. In contrast, there is no prior key exchange in our credit-based authentication scheme. Each time a router generates a session key K_s and signs its own LSAs with symmetric security scheme, for example, by using Keyed MD5 algorithm. The session key will be disclosed until M subsequent LSAs are signed with this session key K_s from this originating router. Note that the sequence number is used as the separation for both the LSA and the session. If the sequence number in the new LSA is M larger than the starting sequence number for the current session key, the router will regenerate another session key for subsequent new LSAs. In the meantime, the previous session key K_s is included in a session key disclosure message (SKDM). The SDKM is signed with the originator's private key and broadcast to the entire network. The SKDM is concisely represented as $(Id, T, K_s, \sum_{i=1}^{M} (Sq_i, MAC_i), Sg_{pri} (K_s, H(\sum_{i=1}^{M} (Sq_i, MAC_i)))$, where Sg_{pri} is the signature with the originator's private key.

Similar to the concept of the credit in our society, we adopt the term "credit" to represent the reputation of the communication reliability between the originator and the

receiver. Initially, each router grants M credits for each router in the network except itself. Each receiver will immediately trust and log the LSAs without verifying the attached authentication code if the credit is available for this originator. After accepting one LSA, the receiver will decrease the credit associated with the LSA originator by one, until 0. The information associated with the LSA will be logged for later verification. To reduce the logging size, we store the hash of the LSA instead of the entire LSA packet. Besides, the MAC, the sequence number, the propagation time $\boldsymbol{\delta t}$ derived from the difference between the timestamp in the LSA and the current time in the receiver, and the IP address / identity of the previous router sending the LSA will be recorded in the local table. Below is the description of the LSA log table.

LSA log table $\boldsymbol{LLT}$ ($\boldsymbol{Id}$, $\boldsymbol{Sq}$, $\boldsymbol{\delta t}$, $\boldsymbol{MAC_K}$, $\boldsymbol{H}$, $\boldsymbol{Ng}$), where $\boldsymbol{H}$ is the hash value of this LSA, and $\boldsymbol{Ng}$ is the neighboring previous router sending this LSA.

After M logs have been stored for one originator, the receiver will neither trust nor propagate the following LSAs claimed to be from the same originator, but the LSA will be logged locally.

Once receiving a SKDM, the receiver will verify it with the originator's public key. If successful, the receiver will use the session key K_s included in the SKDM to check LSA logs for this originator within that session specified by the starting sequence number in the SKDM. Specifically, the receiver key-hashes the H value in the tuple with the session key Ks, then it compares the result with the value from the hash of the $\boldsymbol{MAC_K}$ value in the same tuple. If equal, it means no change was made during transmit; then, one credit is added back and the tuple will be removed from the log table; otherwise, the bogus tuple with verification failure will be stored in a bogus log table. A bogus LSA log

table ***BLLT*** has the same fields as the regular log table but contains those tuples from failed verification. The number of the available credits at the end of verification for one particular receiver indicates if any intrusion against the LSA occurred and how many of the originator's LSAs had been compromised between the originator and the receiver. The lower the available credit in the end of the verification, the poorer the security of the communication channel between the LSA originator and the receiver. The credit may not be the same between two neighboring routers because one router may have received more bogus LSAs from more than one router than its neighboring routers. Figure 1 describes an example of session key usage and message processing between the originating router and the receiver.

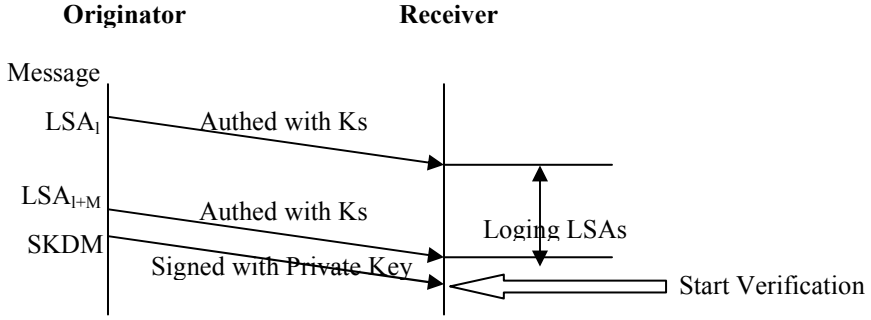


Figure 1 An example of Session key usage and Message Processing

As we see, the value of M directly affects the effectiveness of this authentication mechanism. The smaller value could cause more frequent regeneration of the new session keys and consume the computing power of the router; the larger value may cause the larger log storage in each receiver and the bigger delay to detect the security violation of the routing packet if it exists. Generally speaking, a more trusted network is usually configured with a larger M. Besides, the network stability also affects the frequency of the generation of the session key. The more dynamic the link status change, the shorter

time the associated routers take to reach the M. Thus, the M value can be configured longer in a dynamic network environment than in a stable network environment.

The low credit and the bogus LSAs will invoke the FIR tracing action. In the next section, we will present two practical tracing schemes to locate the FIRs.

3.3.2 Tracing the FIR

From the above, once the routing packets are manipulated by the FIRs, the bogus LSAs will be detected by the downstream non-faulty routers, called **affected routers**, upon the receipt of the disclosed session key. Since the bogus LSAs logs record the neighboring **previous routers** who sent the bogus LSAs, we are able to trace back the faulty origin based on the historical log information. The traceback of the FIRs is accomplished by comparing the bogus LSA log tables from two neighboring affected routers. The discrepancy from the comparison indicates the occurrence of the LSA manipulation. Because only the FIR manipulates the routing packets passing by, the discrepancy also indicates the location of a disruptive link. Based on such an essential idea, we propose two tracing schemes, centralized and decentralized, to fit to different system environments.

3.3.2.1 Centralized Tracing Scheme

We assume there is a centralized management system (CMS) e.g., central intrusion detection system, to conduct the tracing activity in the network. Once detecting the bogus LSAs, the router will report the bogus log table to the CMS. To reduce

communication cost, only five fields, namely ***Id, Sq***, ***δt***, ***H***, ***Ng***, are included in each bogus log table. The bogus log table is transferred via a secure channel between the sending router and the CMS. To simplify our discussion, we assume that the LSAs from only one originator are reported.

With the knowledge of the entire network topology and the routers submitting the bogus log tables, the CMS is able to draw an affected network topology for one specific originator. We define the **affected network topology** G1 consisting of the bogus-reporting routers and represent it with G1(V1, E1), where G1 Є G. With such collected information, the CMS can start the traceback of the FIRs. Figure 2 shows a generic procedure and the centralized tracing algorithm.

The function checkPreviousRouter() defined in Figure 2 is used to check if one bogus LSA reported from one router is able to find a matched one from its previous router. Line (1.5) further checks that the matched one is in order and no loop exists between these two routers because a FIR may want to hide from being detected by modifying the receiving timestamp or misleading the propagation direction of the bogus LSA.

Ctrace() is the centralized algorithm to trace the FIRs. The CMS conducts the neighboring comparison of the bogus log tables by starting any affected router and will go through all the affected routers. In the tracing algorithm, we define the disruptive value associated with each link to locate the disruptive link and measure the severity against the reliability of the LSA relaying between two associated neighboring routers. The value is initialized to 0. As we can see, the disruptive value in the algorithm reflects how much difference exists between two neighboring routers. The non-FIR routers

always forward the bogus routing packets unchanged; thus there should be no discrepancy between the two bogus log tables from the two associated non-FIRs, and the result of the disruptive value over that link will be 0. Only the disruptive link connecting to the FIR(s) shows the inconsistent LSA info or untrue log information. Therefore, the disruptive value will be increased by 1 if there exists one different LSA between two associated neighboring routers. Thus, the link with the disruptive value larger than 1 is the disruptive link. In the section 3.3, we will discuss how to measure the link severity with the disruptive values. The collusive adjacent FIRs, which hide the discrepancy between them, can be treated as a single FIR. Its eventual connection with a non-FIR will disclose the discrepancy and the discovery of the disruptive link certainly isolates the adjacent FIRs.

While mostly the FIRs will compromise the LSAs from more than one originator, it is not necessary to submit all the logs for those affected originators because much overlap will exist in the tracing results - the location of the disruptive links. Therefore, the log information associated with the small number of affected originators is enough to locate the disruptive links and isolate the FIRs. We assume central IDS will determine which originator's bogus LSAs are submitted.

Theorem 2 (Complexity) In the worst case, the runtime of the verification in the centralized tracing algorithm is $\Theta(|E|+|V1|*M*\log M)$ if the binary search algorithm is used, where E is the links of the entire network topology and the |V1| is the number of the router reporting the bogus LSAs.

Proof:

The runtime in checkPreviousRouter() is dependent on the size of the log table in the previous router. In the worst case, the previous router reports up to M bogus LSAs. With the binary search technique, the runtime for checkPreviousRouter() is Θ(logM). From the algorithm in Figure 1, the runtime of the Ctrace()procedure is dependent on the size of the affected network topology and the size of the reported bogus LSA log table. Assuming in the worst case, the size for every bogus LSA log table is M. The runtime for the initialization of the disruptive value associated with each link from (2.1) to (2.3) is |E|. The runtime to conduct the comparison between neighboring log tables is Θ(|V1|*M*logM). Thus, in the worst case, the total runtime of the centralized tracing algorithm is Θ(|E|+|V1|*M*logM).

□

As an example in Figure 3, router 4 is a FIR and it compromises two LSAs from router 1. Each tuple of the compromised LSA in the squares consists of ***Sq***, ***δt***, ***H***, ***Ng.*** After the session key is disclosed by router 1, routers 5, 6, 7 detect the bogus LSA log and report to the CMS. In the meantime, assume FIR router 4 also reports a bogus LSA logs in order to shift the blame away. If the CMS starts from the bogus log table from router 7, the matched bogus LSAs are found in the routers 5 and 6. Upon continuous tracing from the routers 5 and 6, the disruptive values over those links are kept as 0 because FIR router 4 reports the matched bogus LSA info. However, FIR 4 cannot hide the difference of the log tables between itself and its previous router 2 because router 2 does not report any bogus LSA. Eventually, the tracing algorithm returns finding one disruptive link between router 2 and 4, and the disruptive value is 2.

1. Generic Verification Procedure

Assumption:
Tp(Id, s, **δ**t, H, Ng): a bad LSA tuple to be verified,
pTab(1..k2): the bogus log table from the previous router
identified by *Tp*.Ng, 1<= k2 <= M

//The function checks if a matched bad LSA is found in the
//previous router and both tuples are in order
Function CheckPreviousRoute (*Tp, pTab*)

(1.1) **Search** *pTab* through the combined index (*Tp*.id+Tp.s)
(1.2) **IF** *Tp* is in *pTab*
(1.3) THEN
(1.4) **Let** (*pTp* represents the matched tuple in *pTab*)
(1.5) **IF** *Tp*. **δ**t is larger than *pTp*. **δ**t **AND** *Tp*.H is equal to *pTp*.H
(1.6) **AND** *Tp*.Ng <> *pTp*.Ng //check the matched LSA with no loop and in order
(1.7) **THEN** return 0 //A matched is found in the previous router
(1.8) **RETURN** 1 //No matched is found in the previous router

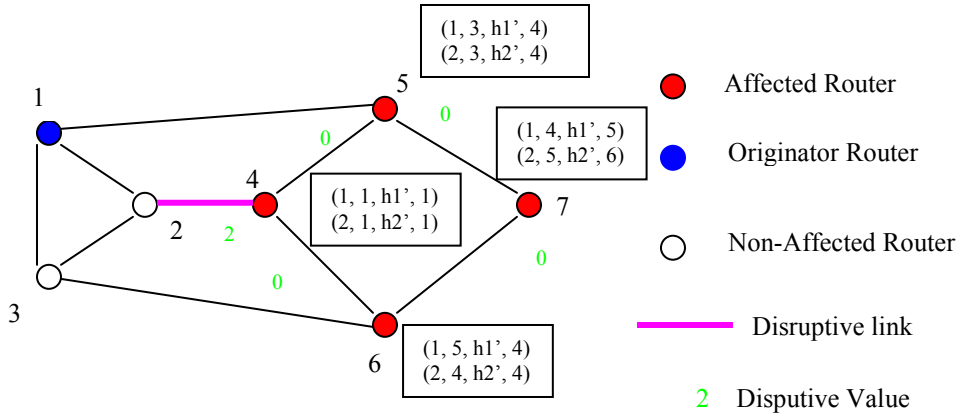
2. Centralized Tracing Algorithm

Assumption:
G1(V1, E1): the affected router topology
e(i,j).disruptive : defined to measure the disruptiveness of the link between router i and j
BLLT(1..|V1|); the reported bogus LSA log tables

//The procedure is defined to trace the disruptive links connected with the FIRs
Procedure Ctrace

(2.1) **FOR** any link e(i,j) in E
(2.2) e(i,j).disruptive = 0;
(2.3) **ENDFOR**
(2.4) **FOR** any affected router i Є V1
(2.5) **FOR** (any tuple *tp* from BLLT(i))
(2.6) **IF** (*tp*.Ng Є V1)
(2.7) **THEN**
(2.8) e(i, *tp*.Ng).disruptive = + CheckPreviousRouter(*tp*, BLLT(*tp*.Ng);
(2.9) **ELSE** e(i, *tp*.Ng).disruptive ++
(2.10) **ENDIF**
(2.11) **ENDFOR**
(2.12)**ENDFOR**

Figure 2 Algorithm for Generic Verification Procedure and Central Tracing



h1' and h2' is the hash of the compromised LSA messages

Figure 3 An example of affected network topology

3.3.2.2 Decentralized Tracing Scheme

In some situations where the CMS is not available such as network congestion or disaster, we propose a decentralized scheme to trace back the FIRs. Different from the centralized scheme in which every affected router reports its bogus LSA log to a CMS, in the decentralized scheme, every affected router asks its previous routers to send their bogus LSA log information in order to check if its links with its neighboring routers are disruptive or not. However, the FIRs may send back the faulty information in order to hide its malicious behaviors. Thus, to detect the false evidence from its faulty previous router, the affected router also queries the evidence from the previous routers of its previous router to prove that the information from its previous routers is not false. In this section, we first present a decentralized solution under the assumption of no collusive neighboring FIRs, then discuss the extension of such a generic idea into the scenario with collusive FIRs.

We called an affected router a **requestor** which requests the innocent evidence from its previous; the **responder** is a router which receives a request of innocent

evidence from its downstream affected routers. While tracing the FIR, a router could be either a **requestor** or a **responder**. The evidence information has the same fields of the bogus LAS table and contains the tuples with the same sequence numbers as those in the request. To simplify our discussion, we assume only one originator's LSAs are compromised though the mechanism can be easily extended to the scenario with multiple originators. We assume *vTab* is the table of the bogus LSAs from the requestor and the *rTab* is the table of the bogus LSAs from the responder corresponding to the request. Below shows the detailed query process,

Step 1: The requester sends a request (*Id*,2) to its previous router where the bogus LSAs in *vTab* are sent from and sets the expiration time.

Step 2: If the responder has all evidence approval associated with this *Id* from its previous routers, it goes to step 4; otherwise, it generates a new request with (*Id*, 1) to its previous routers for the required info and sets the expiration time.

Step 3: If the router is the last router asked for the approval information, it sends back its signed evidence information (rTab) associated with this id. The evident packet is described as below,

$$(\text{rTab}, Sg_{pri}(\text{MD5}(\text{rTab}))$$

Step 4: If the responder has all evidence information, it will append its bogus LSA table associated with the query to the list along with the signature and send back to the requestor.

Step 5: The final requester verifies the status of the link by running the verification algorithm detailed in Figure 3. The result of the disruptive value from the algorithm indicates if this link is disruptive or not.

One requester might receive the bogus LSAs from its multiple previous routers. It will split its *vTab* based on the field **Ng** and send the split *vTabs* to their corresponding previous routers in Step 1. Any received response will be verified by examining the signatures. If verification fails, the response will be discarded and a disruptive link will be announced. The disruptive value associated with this link is increased with the number of bogus packets through this link. If the waiting time expires, the intermediate responder sets the reason for this link where no response is received or the response fails in verification and wraps the received evidence information and sends back to its requester in the same format as in Step 3.

If the responder is a non-affected router, then it will send back its response with an empty log table signed with its private key and announces the disruptive link; if the responder is an affected router too and it does not has such innocence approval information from its UR yet, it will hold the request until it receives the response from its previous routers.

Figure 4 shows the algorithm that the requester runs to determine if the link with the previous router is disruptive and what the disruptive value is based on the collected information from the responders and their previous routers. If the requester finds the discrepancy in the bogus LSA logs with the responder's, then definitely, something disruptive happened in the responder if the requester is a good router. However, the

```
Assume:
vTab:  the bogus log table from the requestor i
eTab(1..k1): the evidence bogus log tables from previous routers of the responder j
rTab: the bogus log table from the responder j

 Function DTrace ()

(4.1)      Initial:  e(i,j).disruptive = 0;

(4.2)      FOR (any tuple tp from vTab)
(4.3)           IF CheckPreviousRouter(tp, rTab) == 1
(4.4)           THEN
(4.5)              e(i,j).disruptive ++;
(4.6)           ELSE
(4.7)              LET tp1 is the matched tuple in rTab
(4.8)              LET z = tp1.ng
(4.9)              IF (eTab(z) exists)
(4.10)             THEN
(4.11)                  LET aTab(z) is the log table for innocence approval from router z
(4.12)                  e(i,j).disruptive = disruptive + CheckPreviousRouter(tp1, eTab(z))
(4.13)             ELSE
(4.14)                  e(i,j).disruptive ++
(4.15)             ENDIF
(4.16)          ENDIF
(4.17)     ENDFOR
```

Figure 4 Distributed Tracing Algorithm

discrepancy may be found between its responder and the responder's previous routers; then apparently either the responder or those previous routers are faulty. The request will first suspect the responder and watch the further reaction from the responder. If the responder has or will soon announce the disruptive link with the specific disruptive values, the requester will go ahead to remove the same disruptive value from its result. The disruptive values from the algorithm will be used to measure the confidence for the link as described in the next section.

Theorem 2 (Complexity) In the worst case, the runtime of the decentralized verification algorithm in the requester is between Θ(M*logM) and Θ ($Mlog^3M$) if the binary search algorithm is used.

Proof:

The runtime in checkPreviousRouter(0 is dependent on the size of the log table in the previous router. In the worst case, the requester received up to M bogus LSAs. From theorem 1, with the binary search technique, the runtime for checkPreviousRouter() is Θ(logM). On the one hand, if no match is found between the requester's bogus LSA log table vTab and the responder's bogus LSA log table rTab, then the lower bound of the runtime of the verification algorithm is Θ (M*logM). On the other hand, if al are found to match, then the algorithm will continue to check if the tuple from rTab is proved from the responder's previous router. In the worst case, the checking of the existence of the bogus log table of the responder's previous router in line (4.9) and the running of the verification of the previous router's tuple against the evidence approval in (4.12) are all Θ (logM). Thus, the upper bound of the decentralized verification algorithm is Θ ($Mlog^3M$).

□

As an example, consider figure 3 with the same assumption as that in section 3.2,1. Among them, router 4 is a faulty router. It manipulates two LSAs from router 1 and pollutes the routing tables in routers 5, 6, 7. The squares attached in each affected router show the bogus log table associated with the originating router 1. Assume router 5 sends a request to router 4. If Router 4 pretends to be innocent, it has to announce the disruptive link between itself and router 2 in order to clear its innocence to router 5. Once the link

e(2, 4) is known to the entire network, router 2 will not send the LSAs to router 4 via this link. As a result, the following LSA propagation originated from router 1 would be immune from the manipulation from the FIR 4.

In the above, we assume that there are no collusive FIRs and each FIR behaves independently. When two or more collusive FIRs are present in the network, they are able to collaborate, and the faulty responder can present its bogus LSA logs with the false evident information from its collusive FIR partners to the requester, thus hide itself or the associated disruptive links from being discovered. To thwart the collusive FIR attack, we still can leverage the generic idea of locating the disruptive link through the comparison of the bogus LSA logs between neighboring affected routers. Instead of collecting the evidence information from the routers one-hop away from the responder, we can extend the query process to collect the evidence information from those routers k-hop away from the responder, where k is the upper-bound number of collusive FIRs in the network, and check if that evident information can support the innocence of the responder through the same method described above. This would, however, substantially increase the cost in the communication and verification. Some optimization on our original approach can be taken to reduce such cost. One of the ideas is to first collect the bogus LSA propagation tree k-hop away from the responder, then query the routers in the tree with the binary searching method instead of collecting the evident approval from all routers in the tree; another one is that we may propagate the query to the routes k-hop away from the responder and ask them to send back the comparison results instead of the entire bogus LSA table.

3.3.3 The Severity Evaluation of the Disruptive Links

A routing packet could be changed within the transmit due to various factors. Some are malicious and intentional, but others are inadvertently and temporary. In real networks, the occasional bad network situation such as network congestion or link noise could lead to a rare and very mild packet discrepancy. A straight removal of such disruptive links may lead to a false alarm. Thus, it may not be proper to remove the link immediately without further investigation, especially in a loosely connected network. Instead, it would be better to acquire the severity of the disruptive link and determine if a link removal is necessary.

The disruptive value enables measuring the severity of the disruptive link and the confidence of the reliability of LSA relaying over this link. We use the confidence value to describe the severity of the disruptive link and it can be represented as follows,

$$\text{Confidence value} = 100\% - \frac{\text{Disruptive value}}{\#\text{LSA received from this disruptive link}}$$

Assume a large portion of the LSA traffic is through one disruptive link. On one hand, if the confidence value is very close to 100%, then it is reasonable to monitor the link further prior to making any decision. On the other hand, if the confidence value is equal to 0, it means every LSA passing through this link is manipulated. It may be reasonable to remove this link to avoid more negative impact over the network routing.

How to handle the disruptive link with a high confidence value is illustrated in next chapter.

Though link state routing protocol is well-known to its self stabilization, the FIRs are able to modify the LSAs at its will and thus seriously impact the network stability, even if the source router notices the change of the LSAs and fights back with updated LSAs. The confidence values help the routers to measure the severity of the disruptive link and determine if a disruptive link shall be kept or removed. By doing that, we achieve the goal of removing the disruptive link as well as avoiding the possible false alarms. The removal of the disruptive link requires one-time table change in each routing table, but it reduces the possibility for the FIRs to pollute the rest of the network further and thus achieve more stable network environment.

3.4 Analysis

3.4.1 Analysis of Correctness and Completeness

Theorem 3 (Correctness) We say both centralized and decentralized tracing schemes are correct if one link is reported as a disruptive link; it is actually a disruptive link connecting with at least one FIR.

Proof:

As we see, both schemes call Procedure checkPreviousRouter() which returns the discrepancy in the bogus LSA log tables between two neighboring routers. The non-FIR always honestly forwards the LSAs with no change to the next routers; thus, if there exists a discrepancy within the comparison, then either the previous router lies or the

current router lies or both of two routers lies about in the information associated with the bogus LSA. In either case, the link is connecting to a FIR, thus is a disruptive link. This procedure deduces the disruptive link from the reasons below,

a) **Exist mismatched LSA**. If the procedure cannot find a matched tuple with the same sequence number from the previous router, it implies either the router itself lies about the bogus LSA or the previous router sent the bogus LSA but denies its misbehavior. Either case is associated with at least one FIR;

b) **LSA matched but disorder**. If a matched LSA is found in the bogus log table from the previous router, with the assumption of a network clock synchronization, the receiving timer in one affected router should be larger than that in its previous router(s). However, if a disorder in the matched LSA is found, it could be that the present router lies about the timer in order to shift the blame away or the previous router intends to deny the sending of the bogus LSA at the earlier time. Either case is associated with at least one FIR;

c) **Detect the multiple-time LSA manipulation**. Since one LSA may pass through more than one FIR, multiple-manipulation could happen within the LSA transmit. One middle FIR may manipulate the LSA again but pretend itself a victim as well. By comparing the neighbor's hash of LSA with the FIR's, the FIR's pretense will be disclosed.

d) **Detect the looparound in a disruptive link**. The looparound could happen if the FIR reports the same bogus information but pretends the bogus LSAs originated from its next direct victim; thus a loop is formed. Criteria a), b),

and c) above will not detect such a security violation. Only the comparison of the previous router identification is able to disclose such a lie.

□

Theorem 4 (Completeness) We say both centralized and decentralized tracing schemes are complete if the bogus LSAs are detected; at least one disruptive link connected with one FIR is discovered with the schemes.

Proof:

We prove this statement in centralized and decentralized schemes separately.

(1) In a centralized scheme, one FIR has three choices for the reporting activity and we discuss them below,

(a) First, the FIR acts as a non-affected router and does not report any bogus LSA. If the neighboring router of this FIR is an affected router and reports the bogus log information to the CMS, the neighboring comparison of the bogus log table will indicate that the FIR was the fault originator. If the neighboring router is FIR as well and intends to hide the misbehavior, the assistance will eventually be disclosed by the comparison with the bogus log table from the non-FIR and thus lead to discovering a disruptive link.

(b) Second, the FIR reports the false bogus LSA log information. In this case, the discrepancy between the FIR and its previous router(s) and the discrepancy between the FIR and its polluted victims will be found after the CMS conducts the neighboring comparison of the bogus LSA log info. In other words, at this case, at

least two disruptive links connected with the FIR will be found through the tracing algorithm.

(c) In the third, the FIR reports the correct bogus LSA log information. Then a disruptive link will certainly be disclosed once the comparison between the FIR and its previous router indicated in the bogus LSA log.

(2) In the decentralized scheme, as we assume there are no collusive FIRs. After receiving a query for the evidence information, one FIR also has the following three choices,

(a) send back its log table and the innocence approval information from its URs honestly to the requester; certainly the requester will find out the descrpency in the bogus LSA tuples between itself and the FIRs as the responder and mark the link as a disruptive link;

(b) ignore the request. If the FIR ignore the request, the affected router timeouts the request and will go ahead announce the disruptive link;

(c) or send back the response with false information. If the FIR decides to send a response with the false log information, it can only pretend it received the same bogus LSAs but will not be able to modify the evidence log tables from its previous routers because the evidence information is signed with the private key of each sender. The comparison between the log table from the FIRs and the evidence log tables as shown from step (4.8) to step (4.15) in Figure 4 will eventually disclose the disruptive link between the request and the FIR responder. Of course, the FIR may want to shift the fault origination

by pretending not to receive from its UR, but it is required to announce the disruptive link between the FIR and its previous router. That also achieves the purpose of removing the disruptive link connecting with this FIR.

Overall, no matter what choice the FIR selects, one disruptive link associated with this FIR will be reported. Thus the statement is proved.

□

3.4.2 Cost Analysis

- **Memory Requirements**

It is required to allocate the memory in each router in the network to store the LSA logs for the delayed verification until the LSA originators reveal their session keys. There are totally 6 fields in the log table as described in section 3.1. The lengths of MAC from either MD5 or KMD5 are 128 bits. It is reasonable to say the range of the timestamp is less 30 minutes / 1800 second; then the size of the propagation timestamp is 11 bits; the originator's and neighbor's address can be represented with the IP address in 4 bytes / 32 bits, and the sequence number is 4 bytes/32 bits in [44]. Thus, each tuple of the log table requires 363 bits, around 46 bytes. Each originator will send up to M LSAs before it reveals its session key which is expected to take a very small time to propagate into the entire network. Note that, the bogus LSA log can be stored in the hard disk in each router so that no memory is required for the bogus LSA log. Overall, at least (M) tuples have to be stored for each LSA originator. Let N represent the number of the routers in the network. Then, in each router the total memory requirement in our proposal is O(N*M*46). As an example, in a large and stable network with 500 routers, assume M is 50, the total memory requirement in each router in such scenario is around 1.15 Mbytes

which is small given the fact that the 512 Mbytes and even Gigabytes memory are often used in network routers presently.

- **Communication Cost**

In the **centralized scheme**, the communication cost depends on the traffic from each affected router to the CMS. As described in section 3.2.1, all fields except the MAC of the LSA are needed to be sent to the CMS; Then the size of each tuple is 30 bytes. In the worst case when one router receives all bogus LSAs, then up to M tuples associated with each individual originator will be submitted to the central CMS. As discussed in section 3.2.1, it may not be necessary to submit the bogus LSA for all originators in order to locate the FIRs in the network because the results – the location of the disruptive links could be largely duplicated. We assume central IDS will choose the log information associated with up to k originators to be submitted. Then, the total size of submission of the log table from each router to the central IDS is O(k*M*30) bytes. In the worst case where the existence of the FIRs compromise all LSAs, each router in the network will submit their logs table for k originators. The network will see a total traffic with O(N*k*M*30) bytes though in reality, the distributed submission will require much less network bandwidth. As an example, let the values of M and N be as the same values as the previous example and pick k as 10, the total traffic to trace the FIRs is 7.5 Mbytes in the worst case. Such traffic is trivial given the fact that 100BaseT and Gigabyte Ethernet are widely deployed in the current network.

In the **decentralized scheme** under the assumption of no collusive FIRs, one of the proprieties in our tracing mechanism is localized, which means the bogus LSA logs transferred from the responders to the requesters will not flood into the entire network

where some areas may not see the bogus LSAs. To conduct the tracing, only two types of messages are sent over the link, one is the request and the other is the response. The worst case is that each affected router received the bogus LSAs from M previous routers separately. One query from the requester contains the Id and the hash value; thus, the size of one query is 20 bytes. Each previous router of the responder sends back up to M tuples, and the each responder sends back up to M^2+M tuples to the requester. With 30 as the size of each tuple, each one of the entire query-response process transfers 20*M+30*M(M^2+M) bytes data and involves (M^2+M) different links. In the worst case, each link involves up to N query-response processes. Therefore, in the worst case, the average traffic over each link is N*[20*M+30*M*(M^2+M)]/ (M^2+M) bytes. As an example, let's pick the same value of N and M as those in the above example, the total traffic to trace the FIRs per link is 750 kbytes, which is much smaller compared with the link capability as described above.

3.5 Chapter Summary

In this chapter, we have proposed a novel mechanism combining security technique and intrusion detection to address the challenges from the FIRs. In our solution, we introduce an efficient authentication mechanism for link state routing protocol to protect the LSAs and ensure the LSA manipulation caused by the FIR is detectable and traceable. Then two practical tracing schemes are developed to trace the location of FIRs with the help from the historical LSA log information. Our analysis shows that the memory requirement in our proposed mechanism is small, and the communication cost is acceptable.

Chapter 4

Improve Network Reachability with Confidence Routing and Secure Virtual Links

4. 1 Introduction

The network routers play important roles in both packet relaying and path determination. However, the importance of the routers in the network and the emerging network attacks highlight the necessity of routing security and protection. It's well known that the cryptographic mechanisms are able to prevent the outsider attack but fail in detecting the insider attacks and the faulty routers. Recently different network intrusion detection techniques have been introduced to monitor the router behavior and alert the defender of the network abnormality. Many network intrusion detection systems (NIDS) either use the known attack patterns (e.g., signature based detection) or discover the abnormality from long-term statistical learning models (e.g., anomaly-based detection). Therefore, when the unknown-pattern traffic or new-model attacks cause the affected routers to behave oddly, sometimes it is difficult for NIDS to accurately and timely identify the reasons and determine the real trustworthiness of those routers. More investigation is needed on those uncertain routers, called routers under monitor (RUM). Recently a numerical indicator represented with probabilistic [56][57] is introduced to more precisely describe the trustworthiness of one monitored router or link. The indication will become more accurate after more time has been spent and more traffic

samples have been collected to learn this traffic pattern or attack model. We call the numeric indicator as one router's confidence value.

With the complicated network circumstances and developing intrusion detection technologies, we can imagine that more and more RUMs may exist simultaneously in the network as time passes. While it may take not a short time and even need human's physical intervention to either clear one RUM as innocent or convict one RUM of an actually faulty router, how to treat those RUMs in the network becomes one challenging issue. On one hand, without distinguishing the RUM from trusted routers in path determination, each router has more chances to pick up the uncertain RUMs possibly with the false link metrics such as bandwidth, link delay, etc in path selection and that results in the significant deterioration in the quality and reliability of packet delivery. We called it INCLUDE approach. On the other hand, if we totally exclude the RUMs from the network, same as the common policy based routing mechanisms which prune suspected routers from the network topology before running the routing algorithm, that could lead to other negative impacts as well, especially in the sparsely connected and resource scarce network system. For example, the traffic blocking over those falsely accused routers may somehow result in the network partition and disrupt the network reachability. We called it EXCLUDE approach. Thus, the conventional approaches of either rashly excluding the RUMs from the network or including the RUMS in the network without extra cautions are inadoptable due to the resource need and the security concern.

To address this challenge, we propose to include RUMs in the network but select the RUMs with caution by taking the routers' trustworthiness and the overall path confidence into account within path selection, and in the meantime continue monitoring the behavior and performance of selected RUMs to ensure the reliability of packet relaying. Especially, in this chapter, we first extend the link state routing mechanism by considering the confidence of each router in the network as one primary parameter in the path determination. As result, an optimal path consisting of routers with the highest confidence values is always selected for packet relaying. Second, to support the reliability and predictability of packet relaying directly over the RUMs, we propose to build a virtual link between two trusted routers across the selected RUMs. A virtual link is a secure tunnel across one or more RUMs. Since the packets are passed through the uncertain RUMs, extra caution must be imposed for data relaying over those RUMs. One important component in virtual link is a simple and efficient monitoring mechanism which tracks the multi-hops connectivity of the virtual link, provides the reliable measurement of link performance, and timely detects the disruptive events over the virtual link. The probabilistic analysis with a simple faulty model demonstrates the advantages of our design over the other two conventional solutions above in respect of reducing the risk of including a faulty router within path determination. The evaluation also shows the convergence of the confidence routing and the efficiency and practicability of the secure virtual links.

The rest of the chapter is organized as follows. In section 4.2, we discuss system models and design goal. In section 4.3 we describe the link state routing with confidence extension. In section 4.4 we present the construction of a secure virtual link, then

illustrate a reliable flow monitoring and analysis scheme for the virtual link. In section 4.5, we conduct the theoretical analysis of security performance compared with the conventional approaches of handling RUMs. In section 4.6, we evaluate the performance of our design and conduct a case study. finally we conclude this chapter in section 4.7.

4.2. System Models and Design Goal

Network Model This work considers the network consisting of routers running link state routing protocols, such as OSPF. A network is described with an undirected graph G(V, E), where V is the set of the routers in the network and the E is the set of links in the network. We assume that a centralized NIDS broadcasts the updated confidence values of the RUMs into the network. The confidence value of each RUM is represented with the η, where $0< \eta <1$. The confidence value for a trusted router is 1 and for an untrusted router the value is 0. An untrusted router is excluded from the network. A confidence value for a link is represented as the minimal of the confidence values of two end routers. The change of a router's confidence value among 1, η, and 0 reflects the corresponding move of the three trustworthy states of this router among, trusted, uncertain, and untrusted. This centralized administration is also responsible for instructing the trusted end routers (TERs) to establish a virtual link via a set of selected RUMs. We also assume that each router holds a pair of keys, one public key and one private key. A public key distribution mechanism has been deployed so that each legitimate router's public key is known to other routers in the network.

Considered Threat Model In this chapter, we focus on the threats from the RUMs, especially the selected RUMs in the virtual link because without being selected for packet

relaying, the RUMs would pose much less threat to the network. If the adversary RUMs reside in the virtual link, they could intercept the packets, fabricate the packets, inject fake packets, artificially delay the packets, misroute the packets, and drop the packets passing through. The attack could be persistent or a selective one which drops the data packets but passes the control packets. The subverted routers could behave either independently or collusively. The colluded adversaries are able to conduct the attacks without being detected easily. For example, they may claim the existence of a faster link between these two nodes so that these RUMs can have more chances to be picked up in the construction of the virtual link. Afterward, they can collaborate to carry out the attacks such as blackhole without being explicitly spotted.

Design Goal Our goal is to develop an efficient and secure solution on the network environment of including the RUMs to increase the network reachability and ensure the security and reliability of packet relaying over those selected RUMs.

4.3 Link State Routing with Confidence Extension

The introduction of the RUMs differentiates the trustworthiness of the routers in the network; thus to accomplish our goal, first we have to extend the link state routing with the consideration of confidence values for each routers in the network. Unlike other link metrics, such as link cost, which are broadcast by the routers themselves within LSA messages, the confidence values are included in confidence messages to reflect the change of the trustworthy state of the associated routers. The confidence messages are generated and broadcast by the centralized NIDS. In default, the confidence value for a trusted router is 1.00. If a router reboots or newly joins the network, it will send out a

query to the NIDS to acquire the confidence information about all presently uncertain RUMs in the network. Upon the receipt of a confidence message, each router will check the authenticity and freshness of the confidence messages first. If successful, the receiving router will update the confidence values of the corresponding routers in its local routing database. In the meantime, the new confidence messages will also be forwarded to its neighbors. After updating the local routing database, the next important step is to update the path selection within the network topology through recomputing the optimal paths to all reachable destination routers. Different from the conventional shortest path algorithm, the new optimal paths consider the path confidence to be the preemptive criteria within path determination. Let **C** and **W** represent the overall weights and confidence values, respectively. Note that, the overall weight of a path is the addition of all weights of the selected links and the overall path confidence is the multiplication of each link confidence associated with those selected links. The optimal path for path selection through confidence routing can be defined as follows,

Definition 1 : An optimal path P(C, W) from the source S to the destination D is defined as, for any other path P' (C', W') from S to D, either C > C', or (C=C') && (W < W')

The computing of the optimal path can easily be accomplished through extending the existing shortest path algorithms. We extend the Dijkstra algorithm [68] with the confidence value as the primary parameter in Figure 5. Note that, confidence routing does not exclude the other parameters or link metrics from the path selection. Instead, it intends to highlight the importance of the routers' trustworthiness in which somewhat reflects the faith toward the commitment for the promised network resource from the

selected routers. Due to the introduction of the uncertain RUMs, selecting RUMs must be done with extra caution. The higher confidence value in one path means that more possibly the capability or resource availability such as QoS is guaranteed and thus the traffic delivery would be more reliable and predictable. To simplify the discussion, we assume link cost is the other link metric considered in path determination. The extended

```
ASSUME:
    s: is the router which wants to establish most confident paths to all other routers.
    S: is the set of vertex which has the shortest distance to s
    C[u,v]: is the confidence value for link (u,v)
    D[u,v]: is the link cost for link (u,v)
    C[u]: is the total confidence value for the path from s to u
    D[u]: is the total link cost for the path from s to u
    G: is the network topology where router s depends on for the construction of the virtual
          link

INITIALIZE_SINGLE_SOURCE (G, s)
    1   for each vertex v belongto V[G]
    2       do c[v] ← unacceptable maximum value; //e.g., ∞ -> weight cost
    3          d[v] ← unacceptable maximum value; //e.g., ∞ -> weight cost
    4           previous[v] ← 0
    5   c[s], d[s] ← init_value  // initialize the starting point

RELAX(u,v)
        Δ make sure we have the larger value in link properties
    1   if (c[v] < c[u]*C(u,v)
    2   then c[v]=c[u]*C(u,v);
    3        d[v]=d[u]+D(u,v);
    4        previous[v] ← u
    5   else if (c[v] = c[u]*C(u,v)) && (d[v] > d[u]+D(u,v))
    6        then d[v]=d[u]+D(u,v)
    7             previous[v] ← u

Dijkstra_Extension (G,s)
    1   INITIALIZE_SINGLE_SOURCE (G, s)
    2   S ← 0
    3   Q ← V[G]
    4   While Q ≠ 0
    5       Do
    6         Δ search the vertex with the highest confidence values in link properties in Q
    7           u ← EXTRAC_MIN (Q)
    8           S ← S union {u}
    9           for each vertex v belongeto Adj[u]
    10              RELAX (u,v)
```

Figure 5 Extended Dijkstra Algorithm to Support Confidence Routing

algorithm helps each router select a path with an overall confidence value as high as possible. If there exist two paths with the same confidence value, the shorter one is the preferred choice for packet delivery.

Theorem 1: *The correctness of this confidence algorithm*

We say that the Dijkstra extension algorithm is correct if this algorithm is able to find the most confident path satisfying definition 1.

Proof:

For this proof, we use the contradiction approach similar to that in [68].

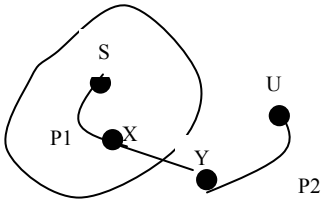


Figure 6 An graph for searching an optimal path with extended Dijkstra Algorithm

In Figure 6, assume node u is the first node selected from this algorithm but does not hold the statement, D(u) <> MINIMAL (u). In other word, we can find a shortest path from s to u but that goes through some vertex not in S. Assume y is the first node in the path but not in S, and x is the processor to y in this path but x is in V. There exists a link between x and y.

On one hand, because y occurs before u in the path from s to u, then

$D(Y) = D(p1)+C(X,Y) < D(p1)+C(X,Y)+D(p2) < D(u)$

On the other hand, when the algorithm runs, both u and y are not in V, but the algorithm in line 7 picks up U, then D(U) < D(Y)

Thus, D(u) = D(y); in other words, u and y should be the same vertex in the graph, that contracts the assumption in the beginning.

□

The network convergence means that after receiving the confidence message, the network migrates to a new steady state. The steady state in this point is that in each legitimate router, D(u) = minimal (W(p)). It is well known that the network running the conventional Dijkstra with the link cost is a convergent network. Here we prove that with the confidence extended Dijkstra algorithm, this convergence property keeps.

Lemma 1 *Once the NIDS sends out a confidence message, every node will receive the message in a finite time.*

With the flooding property and the assumption that no network partition occurs, each router in the network will receive the confidence message in a finite time.

Lemma 2: *Upon the receipt of the confidence message, the computation in each router will result in Minimal (W(p))*

As only the authenticated confidence message is accepted by each router, the same confidence values are held in the network. Besides, within the time of computing the path, the network is not changed; in other words, the link state information in each router is the same. Therefore, the result from the Dijkstra alogirthm in each router is the same, the minimal (W(p)).

Theorem 2: *With the confidence extension, the network is still convergent.*

Prove:

With the lemma 1 and Lemma 2, the theorem 2 is proved.

□

Below we present an example to show how the change of one router's confidence value affects other routers' routing tables. Figure 7-1 shows the original network connectivity and the link metrics in each link before the confidence value of router B is lowered. Two parameters are used for a link metric in this example. The first parameter is the confidence value associated with the link and the second one is the link cost. Table 7-(a) shows router A's original routing table including next hop and path metrics from router A under the original network topology. Figure 7-2 shows the updated link metrics after NIDS lowers the confidence of the router B from 1.00 to 0.80. Table 7-(b) shows the new routing table of router A after recomputing the routing with the updated confidence information. As a result, to deliver the packets from the router A to router C in more confidence, the packets will be relayed via router D instead of the router B.

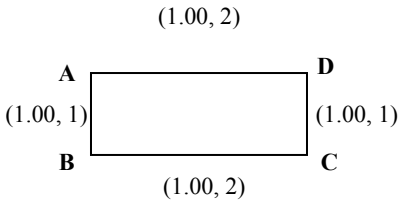


Figure 7-1

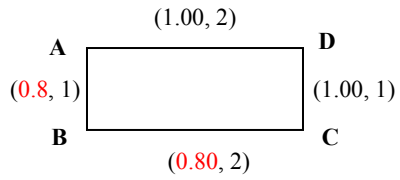


Figure 7-2

Table 7-(a)

Destination	Next Hop	Path Cost
B	B	(1.00,1)
C	B	(1.00, 3)
D	D	(1.00, 2)

Table 7-(b)

Destination	Next Hop	Path Cost
B	B	(0.80,1)
C	D	(1.00, 3)
D	D	(1.00, 2)

Figure 7 An example of Confidence Routing

Due to the existence of the RUM routers in the network, the confidence routing mechanism helps find the most confident route among those RUMs and reduces the risk of selecting an adversary router in a path. However, the uncertainty of the RUMs still brings concerns on the reliability and service guarantee while the packets are actually transferred through those RUM routers. Without further close monitoring, the selected adversary RUMs are still able to disrupt the routing and forwarding services without being detected. To address the concern, we propose to build a virtual link between two trusted routers across the RUMs as described in the next section.

4.4 Secure Virtual links

The virtual link has some unique properties: (1) it passes through one or more RUMs as intermediate hops, but two end routers, TERs, of the virtual link must be trusted routers. In other words, the confidence values for TERs must be 1.0; (2) The packets are forwarded exactly along with this virtual link because we are mainly interested in the connectivity and performance of those involved RUMs. With the virtual link, all traffic can be relayed over the predictably reliable link virtually. In this section, we first present an approach to establish a virtual link and facilitate the tracking of the connectivity and

the measuring of the performance of the virtual link, then develop an efficient monitoring mechanism which is capable of providing the reliable performance measurement of the virtual link and detecting the disruptive event timely over the virtual link.

4.4.1 Link Construction and Packet Forwarding

One TER assumptively launches a construction session to instruct each selected RUM to set policy in packet relaying and protection for this new virtual link. We utilize inter domain security management agent coordination protocol (ISCP) [63] for virtual link construction. ISCP is a generic protocol originally designed to transport the security requirements among security enabled entities, negotiate security agreement, and reserve the security capability from the communication participants. It consists of two phases: discovery phase and reservation phase. The discovery phase is to disperse the security requirement and collect the local keys and security capability from the intermediate routers. The reservation phase is used to transport the local keys to the source router and confirm the path establishment. To better serve the needs in the construction of virtual link, in the discovery phase, the orderly selected RUMs are included in the request signaling so that each intermediate RUM knows where to forward the packets labeled for this virtual link. The figure 8 illustrates an example of the message transmission for the link construction, where each router in the virtual link attaches its security capability (e.g., A_info) and its local key (A_localkey) for this virtual link in the discovery phase, and the termination TER F sends back the integrated security assignment and localkeys to source TER A in the reservation phase.

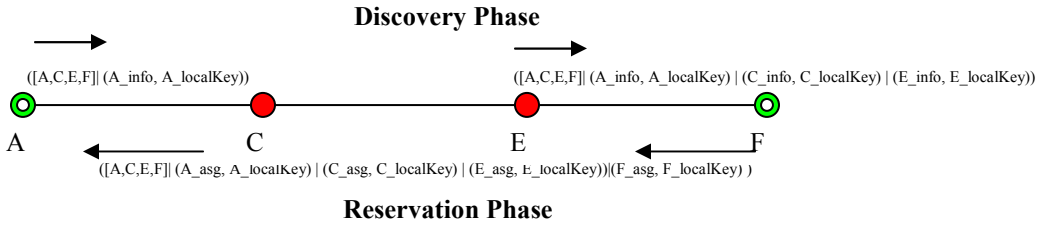


Figure 8 An example of negotiation flow in ISCP

As presented in [63], the security mechanism built into the two phases in ISCP is able to prevent the attacks either from the insider (e.g., compromised RUMs) or the outsider. The shared keys are used to authenticate the packets really for the virtual link and easily provision those verified packets. Besides, the authentication mechanism with the shared keys will reduce the intensive computation overhead in verification compared to the public key crypto graphics. Without being selected in the virtual link, the attacker won't gain the significant advantage in disrupting the packet routing and forwarding service.

Prior to sending a packet destined to the other end router of the virtual link, a monotonic sequence identification (SID) is added in front of each outgoing packet, and each packet must be timestamped and authenticated with the shared keys. The SID is used for synchronization purposes during the link performance measurement. We discuss its usage in detail in the next section. Below is the format of IP packets specifically transferred over the virtual link,

$\mathbf{IP}_{Pkt} = IP_{header} + IP_{payload}$

$\mathbf{IP}_{header} = IPAddr_{TER.SRC} + IPAddr_{TER\_DST}$

$\mathbf{IP}_{payload} = SID + Original\ IP_{Pkt} + TimeStamp + \textit{MAC-List}$

$\textbf{\textit{MAC-List}} = \Sigma LocalKey_i\ (HMAC(SID + Original\ IP_{Pkt} + TimeStamp))$,

where i is the i-th RUM from the sending TER

Once receiving a packet, an intermediate RUM will verify the MAC with its shared key. If verification is successful, it will record the packet in local counters for the data flow from this direction. We will explain the counter in detail in the next section. Then, the packet will be forwarded to the next router in the virtual link. A corrupt or misrouted packet will be dropped and not be counted.

Like the physical link, when either a new virtual link is established or an existing virtual link is broken, an LSA with a virtual link flag and the overall link metrics will be disseminated for the update of network topology. A virtual link is accepted only when the LSAs from two TERs associated with the virtual link are received. The link performance afterward will be updated by the monitoring mechanism described in the next section. Like the change of QoS parameters, to avoid the frequent broadcasting of the LSA due to the link performance swift, the LSA will be generated only when the swift is larger than one predefined range. If the link performance, such as the successful packet delivery rate, is larger than the predefined link-broken threshold, it will be claimed as broken.

The introduction of the virtual link does not require the change for the existing network operation in link state routing. The virtual link can be treated as the new link between two trusted ending routers except that the confidence value of the link is lower that 1.00. While the virtual link is propagated into the network, the link information from the RUMs is still be acceptable, but it is only used for its reachability. For example, for diagnostic purposes, the system may want to send out the inquiry messages to one RUM. If the legitimate routers want to reach the destination which is not RUM, it will ignore the

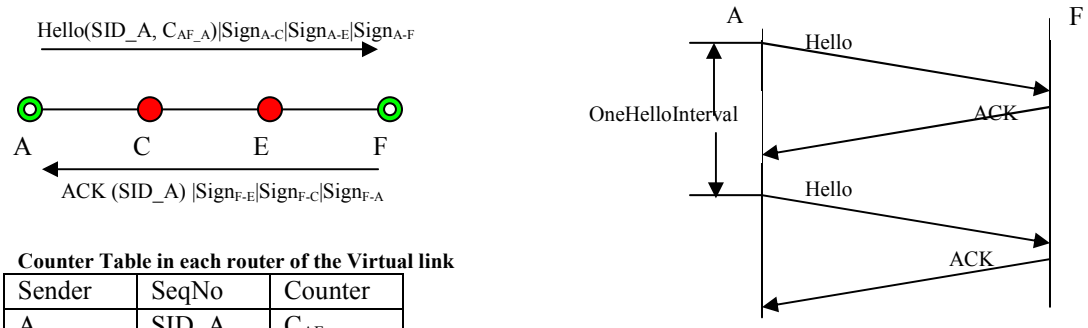
existence of RUM. Instead, the virtual link present in the network topology will be considered within the end to end path selection.

4.4.2 Monitoring the Virtual Links

After the successful construction of a virtual link, a process will be launched immediately to monitor the continuous reachability and the real-time performance of the virtual link. Even the careful selection may not prevent the selected router from being compromised later on or the compromised routers being selected, especially among those uncertain RUMs. Thus, a reliable and efficient built-in monitoring mechanism is necessary to measure the performance of virtual link and report the updates if necessary. Due to the multi-hop property and the uncertainty of the RUMs, the performance measurement on the virtual link is not straight and trivial. The probing mechanism [64] and the sampling approach [59] only use for checking the link reachability within the probing period. They can neither provide the clear and continuous description of the link performance nor detect the inconsistent attack such as a hit-and-run attack and selectively dropping attack from the intelligent adversary. Here we develop a simple and efficient flow analysis approach to monitor the link performance persistently and timely detect the disruptive events along the virtual link. We describe the mechanism below in detail.

Two counters, representing the number of passing packets sent from either direction, are defined in each router along with the virtual link. As an example in Figure 9, counter C_{AF} represents the number of packets from router A to router F, and C_{FA} is defined in the same way. Each of the two counters is associated with a SID from the corresponding originating router in order to indicate the number of packets received

within that time period. As described in section 4.4.1, this SID is indicated in the packets in either direction. If the received packet is corrupt, it will not be counted into the corresponding counter. The MACs computed with the shared keys between the packet originator and the intermediate RUMs prevent the malicious RUMs from messing up the counters, e.g., by injecting the fake packets into the virtual link.



Counter Table in each router of the Virtual link

Sender	SeqNo	Counter
A	SID_A	C_{AF}
F	SID_F	C_{FA}

Figure 9 Hello-ACK mechanism in Detection Phase

Within the OneHelloInterval period, a Hello message is generated by one end router in the virtual link and sent to the other end router along with the virtual link. The Hello message carries the counter for the number of packets sent from this router during this synchronization period and the current SID. The same as the other packets for the virtual link, the Hello message is authenticated with the shared keys. The Hello message is unicast toward the other end TER along the virtual link. A Hello message means the beginning of the detection phase. Note that, after sending out the Hello message, the new packets sent along with the virtual link from this router will be attached with a new SID. Once receiving the Hello message, each intermediate RUM will verify the validity of the Hello message as well as the virtual link indicated by these two IP addresses in the Hello

message. If it is valid, then the status of the virtual link is updated and the Hello message is propagated to the next RUM or the other end router. Upon receipt of the Hello message, the other end router will measure the performance of the virtual link within that oneHelloInterval time. In the figure 9, the updated link performance measurement in packet drop rate can be computed as the below,

$$L_{A\text{-}B} = (C_{AF\_A} - C_{AF\_B}) / \text{OneHelloInterval}$$

If no packet is dropped or the packet drop rate is acceptable, it will send out an ACK message with the same sequence number as that in the corresponding Hello message; otherwise, it will broadcast a link broken message for this virtual link along with the dropping rate to prevent more traffic from being sent over this virtual link. An ACK is authenticated with the shared keys mentioned above. With the information in the Hello message, the receiving end router can compute the packet loss within the Hello interval period. The Hello packet is sent out every OneHelloInterval period. If the Hello originating router does not receive the ACK within a RTT time, it will continue to send another Hello packet with the same sequence number but different timestamp into the virtual link until either timed out or an ACK is received to associate with that Hello message. The intermediate RUM forwards the latest Hello packet to the next RUM toward the other TER.

Like the existing Hello message, the Hello/ACK mechanism is able to inform the other ending router of the continuous link connectivity as well as help measure the link performance. As Hello messages could be dropped in the middle, the ACK message provides the reliability of the delivery of the Hello message and it also shortens the

detection of dropping Hello messages from the attacks and reduces further packet loss along the virtual link across at least one adversary. Without receiving the Hello message or ACK message from the other ending router after a pre-setup time RTT, the virtual link will be claimed broken. The Hello-ACK mechanism is able to detect packet dropping including the Hello message within a reasonably short time. As we see, the Hello/ACK monitoring mechanism is capable of detecting the attack toward not only the data packets over the virtual link but also the monitoring mechanism itself.

The Hello-ACK mechanism is a supplement for the existing one-hop broadcast Hello mechanism, and it is designed specifically for the multihop virtual link scenario. It is able to provide the reachability info and the link performance of the virtual link across multiple RUMs between two end routers and update the status of the virtual link in each intermediate RUM.

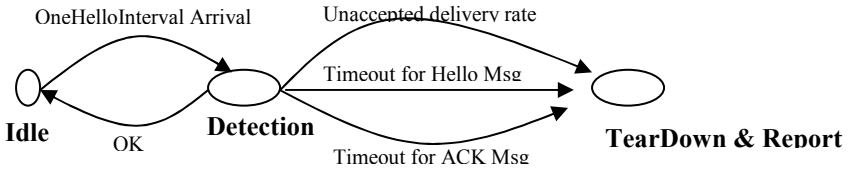


Figure 10. The State Machine in Virtual Link Maintenance

Figure 10 shows the state changes within monitoring a virtual link. If the result in detection phase falls into the above three cases, the virtual link will be torn down and the TER will notify other routers of the broken link with LSA and report the problem to the central management system. The central management system polls the counter table from each router in this virtual ink, including two counters, the corresponding SIDs, and the

latest Hello message. With the comparison of the packet counters, we should be able to recognize the bad adversary or at least the disruptive link with the significant packet drop and Hello packet. The removal of the bad RUM or the disruptive link will reduce the chance to include a adversary RUM in next path determination and improve the network connectivity and packet relaying greatly.

4.5 Theoretical Analysis of Security Performance

Given the fact that the emerging variant attack patterns and the unavoidable false alarms from the developing network intrusion technologies, the adversary routers may not be detected and removed from the network in time even if they behave maliciously. If any one adversary router is selected in a route, then the adversary can easily disrupt the routing or forwarding service over the path either persistently or selectively. We call it a bad selection. Without being chosen, it is more difficult for a adversary RUM to launch an attack. It is interesting to study the probability of including one adversary router with our confidence routing mechanism. Thus, the probability ρ of a bad selection is equal to the probability of at least one selected router being an adversary router. In this section, we present a preliminary probabilistic analysis to study the likelihood of a bad selection in the confidence routing mechanism under the variant false alarming circumstance and also conduct the comparison with both INCLUDE and EXCLUDE approaches in this aspect. .

Here, we adopt a simple adversary model in our analysis. In this model assume in one network with a total of N routers, K routers are actually controlled by adversaries. The NIDS marked K routers as RUMs. Let m represent the number of routers to be selected

for a path, $0 < m+K < N$. To associate the real network scenario with false alarm, we further assume the false positive rate is δ, where $0< \delta < 1$. It means among k suspected RUMs actually $(K* \delta)$ routers are innocent and $(1- \delta)*K$ are adversary routers. Then the false negatives which are missing from the detection are $\delta *K$. Let $g= (1- \delta)*K$ We represent the confidence values of the K RUMs with $\eta_1, \ldots, \eta_k$.

Table 1 Notation for Theoretical Security Analysis

Symbol	**Meaning**
δ	The false alarm ratio from the detection system
N	The total number of routers in the network
M	The number of selected routers in a route
K	The number of the adversaries in the RUM zone
A	Represent the statement "selected routers are non-adversary "
B(K)	Represent the statement "there exist K adversary routers in the network."
$I=\{ \eta_1 \ldots \eta_k \}$	The set of K confidence values corresponding to each RUM in the network
$\sigma (r)$	A complete set of *r* confidence values picked up from the *K* RUMs under A
$\sigma (g, r)$	A complete set of any combined g confidence values picked up from the $I- \sigma (r)$

To facilitate our discussion, we categorize route selection into two cases and study their probabilities separately.

Case one: If all selected routers are from the set of trusted routers, then the probability is represented as below,

$$\rho 1(m) = 1 - \frac{C(m, N-K-(\delta * K))}{C(m, N-K)} \quad (1)$$

Case two: If we cannot find a path consisting of all trusted routers and have to select r routers from the RUMs set, where r < K; then the probability can be derived as below,

$$\rho 2(m) = 1 - (P\{A|B(g)\}) * (1 - \rho 1(m-r)) = 1 - \left(\frac{P\{A, B(g)\}}{P\{B(g)\}} * (1 - \rho 1(m-r))\right)$$

$$= 1 - \left(\frac{\prod_{\eta_z in\ \sigma(r)} \eta_z * \sum \left[\prod_{I-\sigma(r))} (1-\eta_x) * \prod_{\eta_x in\ \sigma(g,r)} \eta_y \right]}{\sum \left[\prod_{I} (1-\eta_x) * \prod_{\eta_x in\ \sigma(g)} \eta_y \right]} * (1 - \rho 1(m-r)) \right) \quad (2)$$

$\prod_{\eta_z} \eta_z$ indicates the probability of including non-adversary routers if r RUMs are selected. $\sum \left[\prod (1-\eta_x) * \prod \eta_y \right]$ means among one specific set of RUMs, the probability of g adversary RUMs and the rest are innocent routers. Our main interest is to analyze the effect due to the false alarms and the dependence on the confidence values of the selected RUMs.

Figure 11-14 shows the likelihood of including an adversary within path selection activity by adopting a confidence routing mechanism. We hold σ(r) and σ(g,r) fixed at 0.7 and 0.3, N in 300 and m at 4 but vary the false positive rate in the axis X. Figure 11 is the likelihood in case 1 where all selected routers are trusted. It shows that the high false

positive rate means more innocent routers are falsely accused and more adversary routers belong to the set of the trusted routers, thus, leading to the increment of the possibility of a bad selection in case 1. Oppositely, the more hidden adversary routers means fewer RUMs actually subverted; thus, the probability of the bad selection for case 2 declines along with the increment of the false alarm rate from Figure 12. It suggests that in an inaccurate and over-sensitive network intrusion detection system with high false alarm rate, including the RUMs in a path does not necessarily mean the increment of the probability in the risk of selecting a bad router. Figure 13 presents the relation between the likelihood of a bad selection and the confidence of the selected routers in a path. We hold $\sigma(g,r)$ fixed at 0.3, N in 300, m at 4 and the false positive rate δ in 0.375, but vary the selected confidence values $\sigma(r)$ in the axis X. The probability of bad selection keeps

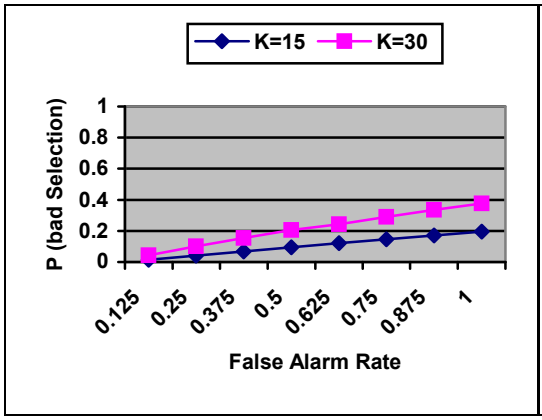


Figure 11 Probability of Bad Selection in Case 1

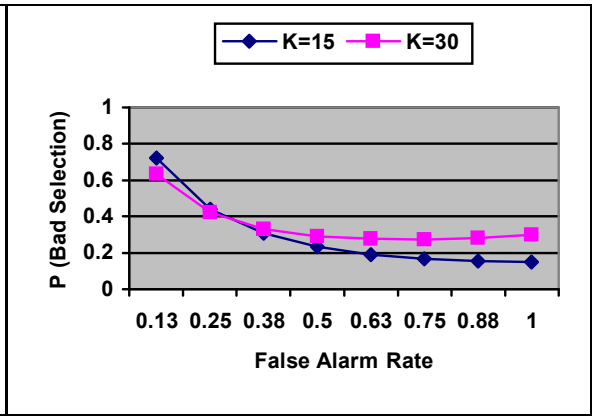
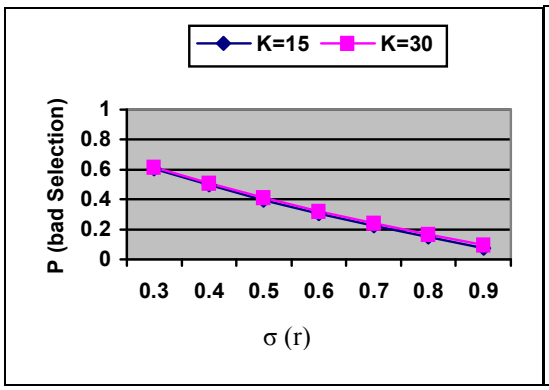


Figure 12 Probability of Bad Selection in Case 2 with r = 1



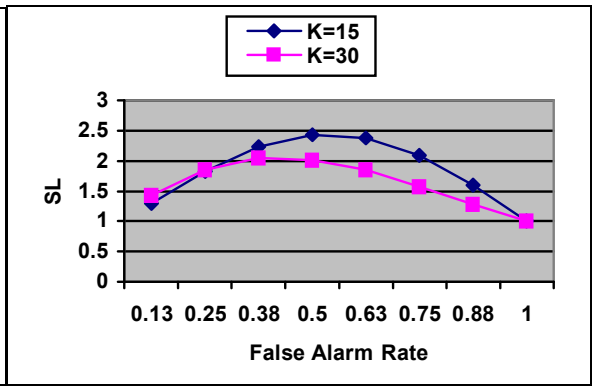


Figure 13 Probability of Bad Selection in Case 2 with r = 1

Figure 14 Safety Loss in Case 2 with r = 1

declining while the selected confidence value becomes larger than the rest of the unselected RUMs. That again illustrates the importance of selecting the RUMs with as a high confidence value as possible to reduce the possibility of a bad selection, which exactly reflects the essential idea of our confident routing mechanism.

The probability for EXCLUDE approach is equal to formula 1 because this approach only chooses the routers from the set of trusted ones. If there exists one route consisting of all trusted routers, our confidence approach will select the same path as well. In this point, the EXCLUDE approach is no different from our Confidence approach. The advantage of the confidence routing approach over the EXCLUDE approach is that once no route with all trusted routers exists, the EXCLUDE approach just reports a broken path and shows network outage. In contrast, our approach is able to select a best candidate from the RUM with lower risk of leading to a bad selection. As an example, in Figure 11 we have 80% possibility to select an adversary-free route with our confidence routing mechanism. This advantage becomes larger in the case that the false positives ratio is very high.

In case two where we have to include one or more RUM in our route establishment, the confidence routing mechanism is able to select the routers with as a high confidence value as possible. It is different from the INCLUDE approach which treats all RUM as indifferent from trusted routers. Under case two scenario, the probability for INCLUDE approach is

$$\rho_3(m) = 1 - \left(\frac{C(r, (\delta * K))}{C(r, K)} * (1 - \rho_1(m-r)) \right)$$

To easily see the advantage of the confidence routing mechanism over the INCLUDE selection approach, we define safety loss (SG) as the comparison between $\rho 3(m)$ and $\rho 2(m)$.

$$\mathbf{SL}(m) = \frac{P3(m)}{P2(m)}$$

SL shows how well the confidence routing mechanism reduces the risk in selecting the non-adversaries in comparison with INCLUDE approach. If the SG is larger than 1, it means that the INCLUDE approach is more likely to establish a path with adversary RUMs. To shorten our discussion, we only show the comparison under the variation of the false alarm rates. In Figure 14, we hold $\sigma(r)$ and $\sigma(g,r)$ fixed at 0.7 and 0.3, N in 300 and m at 4 but vary the false positive rate δ in the axis X. Under the same false alarm rate, the INCLUDE approach always has higher probability of selecting an adversary router in a path determination than the confidence routing mechanism. As observed, the safety loss becomes more than two folds within the false alarm rate equal to 50%.

In summary, from this section, we know that the probability of a bad selection from our mechanism is affected by multiple parameters. The intent of our mechanism is to reduce the risk of selecting a bad RUM router as well as leverage the resource from the falsely accused RUM as much as possible. The comparison between the confidence routing mechanism and other present RUM-handling mechanisms confirm the more advantages gained from our approach in this aspect.

4.6 Performance Evaluation

With the introduction of the path confidence into network routing, we are able to leverage the network resource as much as possible in such a network with high false alarms while reducing the risk of selecting the adversary routers for packet delivery. In the meantime, the tradeoff is to add more network operations and increase the overhead to the network. In this section, we first want to study the effects toward network stability and network performance after the introduction of confidence routing. Second, we analyze the cost with the presence of the secure virtual link and demonstrate the practicability of secure virtual link through a case study. In confidence routing, we would measure the effect based on the metrics below,

- ***Convergence times*:** we are particularly interested in the time when all routers in the network receive one confidence packet and recomputed their routing table.
- ***Message overhead***: the number of the confidence messages packets is received in the network after the NIDS sends out one confidence message

- ***Average route flap***: the average number of entry changes in the routing table for each router in the network upon the receipt of the confidence message.

- ***Average path length increase***: As our proposed confidence routing mechanism selects the routes based on the confidence value first and the link cost in the second, we will see the increase of the path length. We want to compare with the traditional shortest path routing protocol which only considers the link cost in path selection. We use the formula below to study such effect.

$$\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \left(d(i, j) - d_s(i,j)\right)}{N*N}$$

where, N is the total number of RUMs and trusted routers in the network at that time, i and j represent two reachable routers in the network. The d(i,j) is the path length when the confidence routing is adopted and $d_s(i,j)$ is the path length computed from the shortest path algorithm. Note that, the untrusted routers are removed from the network and not included in N.

4.6.1 Simulation Environment

Our design is implemented with network simulator ns-2 running on Linux 2.4.18 in a 1.2 GHz PC. Each router in our simulation runs the link state protocol with our new Dijkstra algorithm to compute the most confident routing path. We use Openssl [65] as a cryptographic toolkit to provide security capability required in processing of the confidence packets and virtual link.

We created the network topology with BRITE [66] according to the RouterWaxman model. To observe the different performance effect along with different network topology, we conduct the simulation in the networks consisting of 100 and 200 routers respectively. The maximum network diameters in the generated topologies are 6 in the topology with 100 routers and 7 in the topology with 200 routers in the beginning. The entire network is designed as one OSPF area. To focus on the study of the effect due to confidence parameter, we assume all routers with the same data rate, 10 Mb/s and the same link delay, 2ms. A NIDS connects to one network router from which the updated confidence packets are propagated into the entire network.

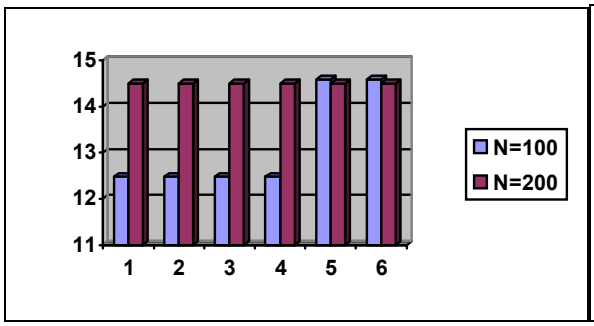


Figure 15 Propagation Time

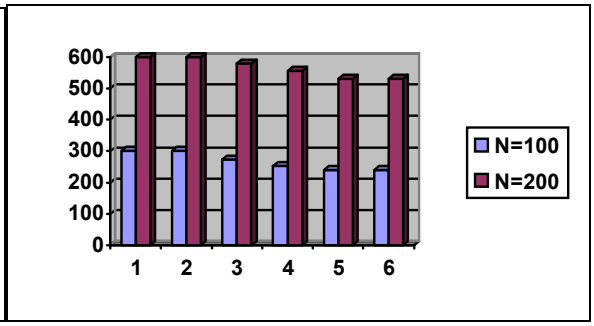


Figure 16 Message Overhead

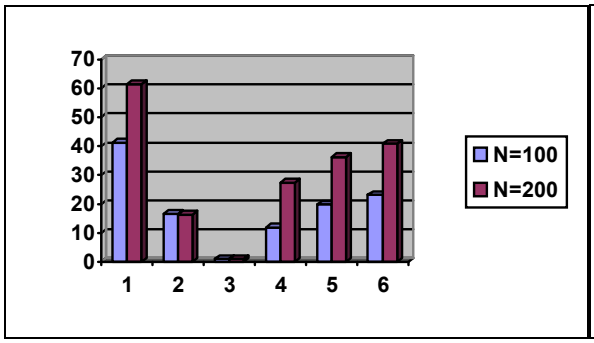


Figure 17 Average Route Flap

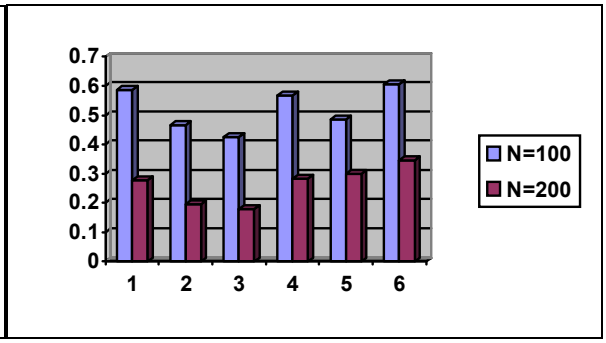


Figure 18 Average Path Length Increase

The duration per experience is 1800 seconds. As we discussed in section 4.3.2, different state changes lead to different network operations in each router. In order to observe and compare the different effects along with the different combination of the state changes, we generate a set of confidence messages with different combinations of state changes (see table 2) over the routers with high degree. Those packets will be generated and sent out every 300 seconds during the experience period. Note that, state changes between TRUST and UNTRUST are not included in our experience because such state changes are similar to the link failure and link recovery which have been well studied in recent years. Note that T, U, and R represent three different trustworthy states, trust, untrusted, and RUM(uncertain) state. The second row in table 2 represents the type of events and the number of events included in one confidence message. For example, the first message contains 4(T->R) which means the trustworthy state of 4 routers is changed from Trust to Uncertain.

Table 2 Event Messages Used in Simulation

Message #	1	2	3	4	5	6
Event	4(T->R)	1(R->T)	1(R->U)	1(T->R) 1(R->U)	1(T->R) 1(R->T) 1(R->U)	2(T->R) 1(R->T)

4.6.2 Experience Results

In Figure 15, axis Y represents the convergence time in ms. Despite of the number of the routers in a network, it shows in Figure 15 that the propagation time of one confidence packet is roughly proportions to the maximum traveling distance between the

NIDS and any other routers in the network. Note that, after message 4, the link broken from the vent (R->U) leads to the network diameter increase by 1 in the scenario with 100 routers. Thus, the later propagation time in this scenario increase to the same as that in the network with 200 routers. The result also has been confirmed by many other research papers. Thus, to reduce the propagation time, it could be better broadcast the packets from the center of the network.

Figure 16 presents the control message overhead in our experience. Axis Y reflects the number of the confidence messages receiving by all routers after one confidence message is sent out. The message overhead is determined by the number of the up links in the network. In our design, a router floods the fresh confidence message to its neighbors but discards the packets received before. Also, a confidence message won't be propagated into the original sender. Thus, the overall message overhead should be equal to $\sum Di - N$, where $\sum Di$ is the total degrees of the routers and N is the number of the routers in the network. The larger network with more link degrees generated more control messages. The magnitude of message overhead shows a little change after receiving different confidence under the same network topology. In Figure 16, we observed that the message overhead in processing packet 3, packet 4, and packet 5 declines a bit. The reason is because either the message itself or its previous message contains the state change from RUM to totally untrusted and no more messages are generated or sent to the untrusted routers. Note that, in the relatively calm network, the message overhead is trivial in comparison with the regular routing traffic which is generated by all the routers either periodically or bustly.

The route flop is one of the important measurements for network stability. The more route flop, the more unstable the network. As we change the states of the routers with a higher degree, we observe the relatively big route flop after processing the confidence message. In figure 17, as we include four state changes for four routers in packet 1 and the routers with affected routes recompute the new path which may has different path cost and next hop, thus, we observe a significant route flop in our experience. The route flop happening in packet 2 is because adding a trusted router back to the network may shorten some routes in the network. The reason why the route flop is smallest for packet 3 is because many routes did not include this previous RUM, and thus the removal of the RUM from the routing table of each router did not cause much churn in the routing table except marking this destination as unreachable / untrusted. Besides, the reason that the larger network has more route flop is because it has more routes passes through those routers in our example before the state change of those routers occurs.

The conventional approach with the shortest path algorithm is equal to the INCLUDE approach which treats all RUMs the same as the trusted routers. Figure 18 illustrates the average path length increase in Confidence approach. The increase shows that to acquire the most confident path may require rerouting the packets in a longer path but the gain from the tradeoff is to reduce the possibility of selecting one more compromised router(s) into the route as discussed in section 4.5.

As illustrated in section 4.4, extra cost must be considered in order to enhance the security and protection in the link performance measurement in the virtual link. The first cost is the overhead added in each packet transferred over the virtual link in order to support the authentication and integrity against the compromise of the packets in the

middle. As described in section 4.4, the new components, namely SID (2 bytes), timestamp (4 bytes), and MAC-List (16 bytes for KMD5), are appended along with each data packet. It adds (16*k + 14) bytes overhead to each packet, where k is the number of routers in the virtual link. Note that, given the fact that the number of RUMs is small and the path determination algorithm e.g., confidence routing, also considers the path length, thus, the overhead introduced by the extra authentication is acceptable. The new Hello message includes SID, the counters (two bytes per counter), the timestamp, and the signature list, and the new ACK message includes SID and the signature list. The total length for one Hello message is (18+16*k) bytes, and the ACK message is (10+16*k) bytes. Note that, the Hello overhead is only applied to the virtual link not globally. Thus, the overhead from a small set of virtual links is quite trivial and can be neglected.

Another cost is the latency from the authentication of each packet toward the other end router of the virtual link and the verification of each packet in RUMs. In our evaluation, we establish a set of virtual links across 3, 5, 8 routers, respectively in our experience. Figure 19 shows the delay comparison between the virtual link mechanism and non-authentication relaying. As we can see, the computation from the authentication and verification with the shared keys costs quite a little compared with the overall packet delay in the virtual links.

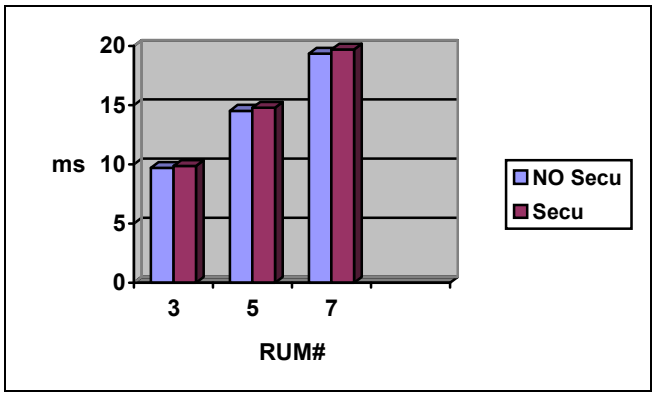


Figure 19 Computation Overhead for the virtual link

Hello message is critical to detecting the link reachability and measuring the link performance. The interval of the oneHelloInterval determines the sensitivity of detecting the link outage. The longer interval may cause the slower detection of link outage; the greater frequency of Hello messages means more overhead toward the virtual link. In our experience, we use 10 seconds for oneHelloInterval and 3 seconds for RTT time used to reply with ACK. To evaluate the link performance monitoring mechanism, we simulate a scenario that one compromised RUM among 5 selected RUMs arbitrarily drops the packets passing though. Assume that during the experience, a CBR traffic in 1.52MB/s rate is passing the virtual link. The alerting threshold for the dropping rate is 0.3%, around 4.6Kb/s. For convenience, we will not tear down the link even if the TER detects the disruptive event. Instead, the monitoring mechanism continues to monitor the performance of the virtual link and report the disruptive events. The simulation result is shown in Figure 20.

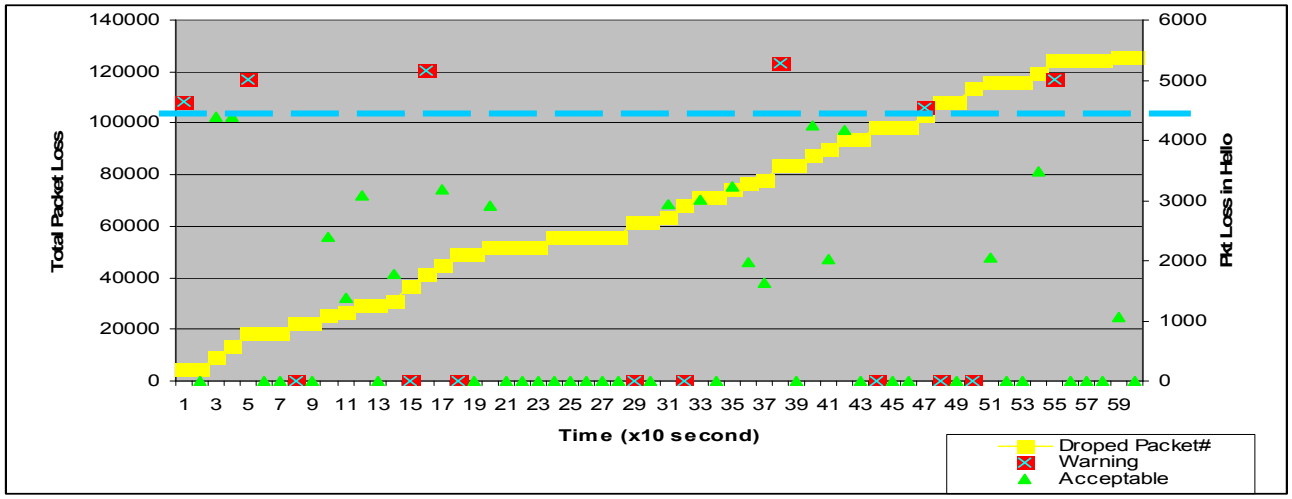


Figure 20 10-minute Monitoring Report of one Virtual link

The yellow curve shows the total number of packets dropped at a certain time. The most aggressive dropping happened within the period from the 150th second to the 180th second. The right axis Y shows the number of dropped or corrupt packets within oneHelloInterval. The blue dotted line in the diagram shows the alerting threshold. The green dots show that the Hello messages are received in the other side, and the dropping rate is under the threshold. If the green dot crosses the axis X, it means all packets within that period are transported correctly. The Red dot gives the warning if either the dropping rate is larger than the threshold or the Hello messages in the corresponding period are continuously missing. If the red dot crosses the axis X, it means the Hello messages within that period are lost and thus possibly the adversary RUMs attacked the monitoring mechanism. As we can see, the Hello/ACK mechanism is able to accurately and timely reflect the performance fluctuation of the monitored virtual link.

4.7. Chapter Summary

The complicated network circumstances and the developing network intrusion detection technologies lead to the emergence of the RUMs, uncertain routers behaving unexpectedly, in the network. Current approaches either rashly exclude RUMs from the network or reuse them without extra cautions. Neither approach is acceptable due to the resources needed and the security concerns. This chapter presents a new solution – selecting a path with the consideration of routers' confidence values and establishing a reliable and secure virtual link over the selected RUMs to increase the network reachability and ensure the reliability of packet relaying over the RUMs. With the consideration of the router's confidence values, an optimal path consisting of the routers with highest confidence values is selected for path determination. To continue monitoring the link reachability and the link performance, a flow analysis mechanism must be deployed over the virtual link to collect the packet delivery information and compute the link metrics in real time. Mainly three benefits are gained from our solution including the confidence routing and secure virtual link, (1) maximize the network routability and reachability, especially within the false positives; (2) quickly detect the fault and the outage in the selected virtual link between two participating routers; (3) seamlessly integrate into the link state protocol with little impact on the existing routing infrastructure. Few changes have to be made in order to use the virtual link idea in the existing routing protocol, such as OSPF. The preliminary theoretical analysis illustrates the advantages of our design in reducing the risk of including an adversary in path selection. The cost and overhead from the confidence routing and the introduction of the virtual link are trivial compared to the overall network traffic.

Chapter 5

An Approach to PKI Certificate Chain Discovery in MANET

5.1 Introduction

Mobile ad hoc networks (MANET) are those networks which consist of a collection of wireless mobile hosts and are formed on the fly without the aid of any pre-established infrastructure and centralized administration [4] [5] [6]. MANET is self-deployable and self-organized. Each node acts as an endhost, sending/receiving data and, maybe at the same time, a router, relaying data for other nodes. To support node mobility, MANET is intentionally designed to quickly adapt to network changes and operate without human interference, thus, increasing the survivability of the network in unfriendly and even hostile environments, such as in disaster or battlefields. Because of these promising conveniences and benefits, we are envisioning an increasing number of MANET applications in different areas, such as military fields, rescuing operations, mobile conferencing, home networking, etc. While many new opportunities are coming out along with this new technology, MANET brings new, challenging issues as well. In this chapter, we focus on the concern about public key certificate validation in MANET.

Public key cryptography is the prerequisite of network security. Two parties, who have never met before but know the other party's public key, can securely communicate

through public key based protocols, such as IP-Sec [23], S/MIME [24], TLS [25], etc. Before used, a public key has to be certified by the certificate authority (CA), a trusted entity, and timestamped with a proper expiration time. Public key certificate, used as digital credential, proves validity and authenticity of its ownership and other properties. In security-sensitive systems, certificate validation which includes public key verification and trust verification, must be done between communication parties in order to protect resource access and communication against security attacks such as spoofing and masquerades. Certificate validation is done by checking the signature of the certificate with the authentic public key of the CA of the verified party. However, in public key infrastructure (PKI), the certificates of communication parties may be signed by two different CAs; thus a CA certificate path between these two CAs must be established to retrieve the authentic public key of the CA of the verified party. Conventional PKI in wired networks provides a centralized entity such as a public key directory to be queried for a certificate path from a global public key certificate hierarchy. The directory, normally a database running on one machine, supports maintenance of the public key certificates and verification requests from legitimate subscribers.

The significant characteristics in MANET are infrastructure-less; thus, no central entity is expected to be present in such networks. Even if we have a global CA around the neighborhood of a particular MANET, due to the wireless network dynamics, it might not be available to all the nodes at all time.

To support the concept of infrastructure-less in MANET, we explore a new public key management approach without relying on centralized entities, such as on-line CAs or a static, centralized directory. As observed, public key certificates can be

categorized into two types, CA certificates and normal subscribers' certificates. It is the relatively small-size CA certificates that are used to verify the validation of one host's certificate. In our approach, we assume each mobile host keeps a set of valid CA certificates. We argue that saving a complete CA certificate hierarchy in any mobile host is infeasible but it is acceptable to locally cache a set of CA certificates recently used. Local caching of CA certificates is able to reduce both dependency on one or more central certificate administration entities and delay in certificate verification, but it may

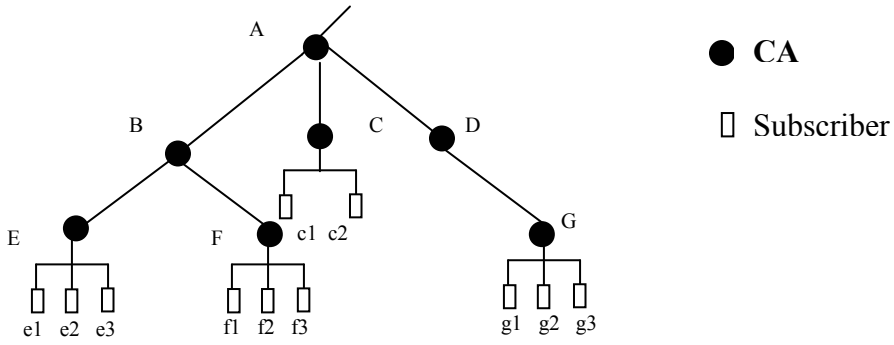


Figure 21 An Example of Hierarchical PKI model

not be able to verify all unknown hosts' certificates because of its insufficient CA certificate information. It is critical that a dynamic certificate path discovery activity can be initialized on demand and help collect needed CA certificate information, if at all possible, for a complete and authentic certificate path. In this chapter, we will discuss the difficulty and complexity of certificate path discovery in the absence of static centralized entities in MANET, present a codeword-based certificate path discovery (CCPD) algorithm to speed up certificate path discovery, and eventually propose an on-demand certificate collection protocol (OCCP) to accomplish certificate path discovery on the fly. Note that, we assume that each host gains its certificate and initial CA information on line

prior to its mobility. In this chapter, we focus our discussion on hierarchical PKI trust model (see Figure 21) [33] [33] [34]. The results however can be easily extended to other PKI trust models such as the bridge PKI model.

The rest of the chapter are organized as follows. We discuss the challenges of certificate path discovery in MANET in section 5.2. We represent each CA node in hierarchical PKI with unique codeword and design a CCPD algorithm to speed up the certificate path discovery in section 5.3. In section 5.4, by adopting the CCPD algorithm we propose OCCP protocol to collect the needed CA certificate information in MANET, and discuss security vulnerability and countermeasures of OCCP. We present our implementation and simulation results in section 5.5. Finally, we conclude this chapter in section 5.6.

5.2 Challenges in Certificate Path Discovery in MANET

We borrow concepts and terminology from [10] to discuss certificate path and the challenges of certificate path discovery in MANET. As we know the hierarchical PKI trust model consists of a root CA, delegated multiple layered CAs, each of which is able to certify subordinate CAs or subscribers and subscribers [33][33][34]. The certification relation indicates the certifying party believes that the certified party really links with the public key and really holds the capability included in the certificate. If the certified party is a trusted entity, the certification relation also indicates the certified party is trustworthy to issue certificates.

Table 3 Notation Used in Certificate Chain Discovery

Notation	Meanings
KU_A	Party A's public key
KR_A	Party A's private key
<<>>	Certification relation
KR_A<<KU_B>>	Party B's public-key certificate signed by CA A, briefly A<<B>>

For the purpose of simplicity, each node's certificate is represented with the signature of this node's public key as seen in the notations in table 3. Based on the certification direction, we can categorize two types of certificates: forward certificate (FC) signed by one CA to either its subordinate CA or its subscribers, and reverse certificate (RC) signed by one CA to its parent CA. As one example in Figure 20, B is a subordinate CA of A, and it also certifies two subordinate CAs, E and F; thus, certificates related to CA B are listed as follows,

FC: KR_A<<KU_B>>, KR_B<<KU_E>>, KR_B<<KU_F>>
RC: KR_B<<KU_A>>, KR_E<<KU_B>>, KR_F<<KU_B>>

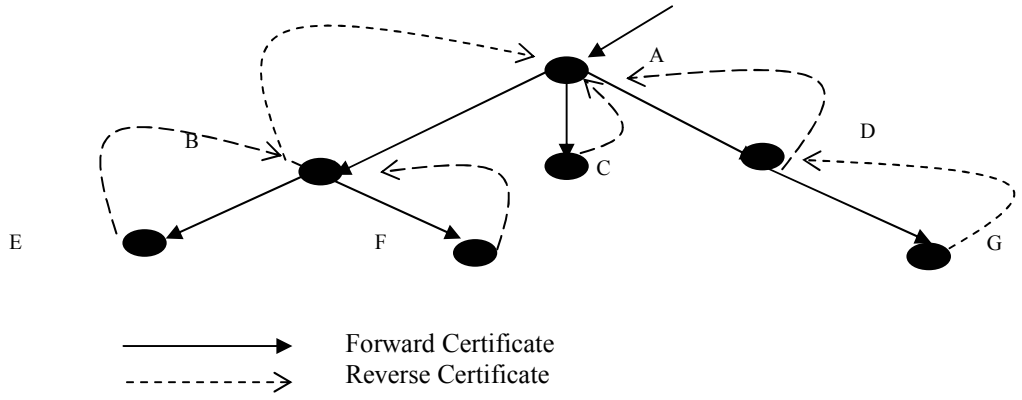


Figure 22 Examples of certification relation in Hierarchical PKI model

Assume each subordinate CA will generate the reverse certificate to its parent CA. Figure 22 illustrates the certification relation in Figure 21 with directed edges.

A certificate path is needed within certificate validation when the certificates of two communication parties are signed by different CAs. Generally speaking, certificate path is an ordered chain of CA certificates where each public key can be certified by the previous entity in the chain. Definition 1 gives the formal statement of certificate path.

Definition 1. Certificate Path is a chain of CA certificates KR1<<KU2>>KR2<<KU3>>…..KRi <<KUi+1>> …KRn-1<<KUn>> for n≥1, where i is a CA node in one hierarchical PKI diagram.

A certificate path implicitly indicates a transitive trust relation that a trusted entity is capable of issuing certificates. The public key in one certificate is used to verify the validation of the next certificate in the ordered list. Therefore, with a valid certificate path, one party is able to retrieve the authentic public key of the other party's CA, and then verify the signature of the other party's certificate. An example of certificate path in Figure 22 is $KR_A<<KU_B>>KR_B<<KU_E>>$. After being presented with this certificate path, one host holding a certificate signed by CA A is able to validate certificates signed by any of these three CAs. A certificate path does not exclude the coexistence of one CA's FC and RC. In a hierarchical diagram, such coexistence means that the path passes a CA node more than once. It certainly adds redundancy in the certificate path. Apparently, a desired certificate path should just carry the non-redundant certificate information. Such a certificate path is called an optimal certificate path.

Definition 2. Optimal Certificate Path P is a chain of certificates $KR_1<<KU_2>>KR_2<<KU_3>>\ldots..$ $.KR_i<<KU_{i+1}>>\ldots KR_{n-1}<<KU_n>>$, for $n \geqslant 1$, where i is a CA node in one hierarchical PKI diagram and for all $KR_i<<KU_j>>$ in P, $KR_j<<KU_i>>$ NOT in P

Based on the certification relation, we can model hierarchical PKI model as a directed graph G=(V, E), where V is a finite set of distinguishing CA nodes and E is a set of directed links representing the certification relation from one node to the other. The objective of certificate path discovery is to find a directed path between two CA nodes in the graph G. Our goal here is to search an optimal certificate path efficiently and effectively.

Definition 3. *The distance of any pair of certificates in one certificate path is the number of the certificates between these two certificates.*

With full certificate hierarchy in a central directory, an optimal certificate path problem can be solved in O(|E|+|V|) with a graph algorithm such as DFS [11]. However, with our assumption of an infrastructureless environment, each host could only store some CA information of recent interest. In other words, each host may have only a small piece of knowledge of the certificate hierarchy. Therefore, sometimes locally cached CA certificate information may not be sufficient to generate a complete certificate path. In order to find a certificate path, one has to collect more CA certificate information from other nodes. However, given any two certificates in any certificate hierarchy, there are several possibilities of how to expand the certificate collection between their two CAs. In Figure 23, we show four possible relative locations of any pair (X and Y) of CAs.

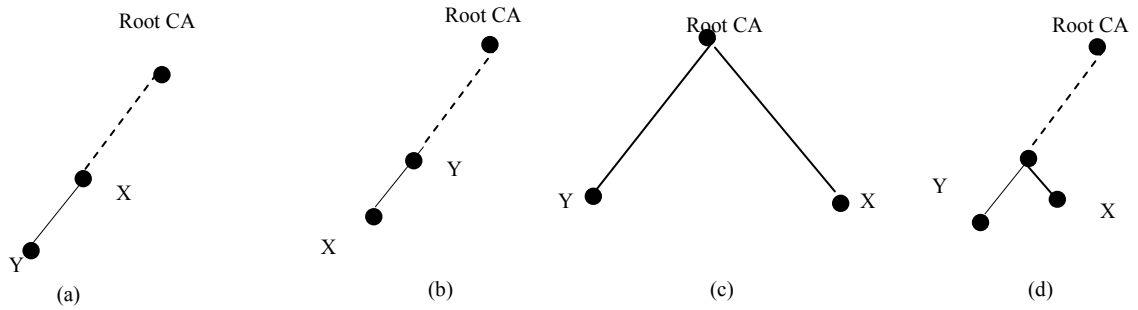


Figure 23 Possible relative positions of two CAs in hierarchical PKI model

The uncertainty of relative location of two CA_Xs in certificate hierarchy highlights the complexity and difficulty in certificate expansion for the discovery of an optimal certificate path. Also, it is a fact that doing a search in ad hoc networks is very complicated and expensive. Blindly expanding certificate collection may lead to a certificate path with redundant certificates, as even worse, may have to suffer fail-and-retry several times. An intuitive approach, we call *Straight-up,* intends to discover a certificate path by exhaustively expanding collection from both of two parties up to the root [33] [34]. It's proved in our simulation to cost much time and consume much bandwidth and energy; therefore, it is also undesirable. To avoid such exhaustive or 'fail-and-retry' searching, a more efficient approach to discovering the certificate information along an optimal path is desired. To address this issue, we propose a dynamic certificate path discovery approach. In the next section, we introduce an encoding strategy to help narrow down the searching scope of a certificate path, and then we present an on-demand certificate collection protocol to complete the certificate path discovery on the fly.

5.3 Encoding Public Key Certificate Hierarchy

Though the certificate relation in one PKI hierarchy is modeled as a directed graph, the single-rooted hierarchical PKI trust model with only forward certificates can be treated as a tree structure. Due to the constraint in geography and the complexity in implementation, apparently it could be very difficult to ask each CA in the certificate path to sign one public key before issuing a certificate. However, it might suffice if the path could be marked in each certificate, which is our proposed idea, including an encoded path from the root to the current CA node inside each certificate. For flexibility, we consider that each CA is able to assign an arbitrary number of subordinate CAs. There are different approaches to address tree encoding [26][27][28]. In this chapter, we adopt the child-sibling approach from [29] to construct a binary tree from an arbitrary multiway tree. In this child-sibling binary tree representation, the left pointer of a node accesses its first child, and the right pointer of a node accesses its next sibling. The link to the left child node is labeled with bit 0 and the link to the right child with bit 1.Each CA node in the hierarchy holds a codeword consisting of the accumulated 0-1 sequence from the root to the CA node in question. Each parent CA assigns to its subordinate CAs with its codeword plus a new distinguished code. A generation of codeword for each CA is represented as follows,

$$H_j = \begin{cases} H_i0 & k=1 \quad (1) \\ H_i0\,(1)^{(k-1)} & i,j \geq 1,\ k>1 \quad (2) \end{cases}$$

where CA i is parent of CA j, CA j is k-th child of CA i.

In Figure 24, we show an example to represent Figure-1 with child-sibling binary tree and generate a codeword for each CA node. Thus, we have a codeword for each CA node as follows,

A(X)
B(X0) E(X00) F(X001)
C(X01)
D(X011) G(X0110)

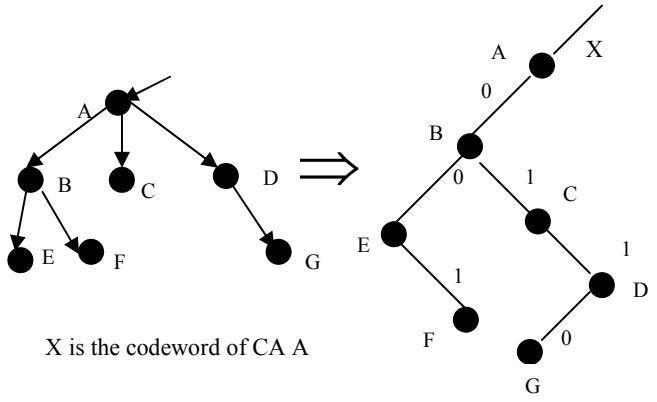


Figure 24 An example of codeword generation from a tree-structured CA certificate hierarchy with only forward certificates

Root CA has codeword '0'. With codeword, one certificate can be represented as $KR_i(KU_j, H)$, where party i is a CA, H is party j's codeword if party j is a CA as well; otherwise, H is party i's codeword.

The encoded public key certificate hierarchy with codewords shows some useful characteristics,

Lemma 1. *The codeword of parent CA is the prefix code ending before the last '0'.*

Proof:

As shown in generation of Codeword from (1), the first child ends with a '0' and its sibling is distinguished with a number of '1''s. □

Lemma 2. *The number of '0' in codeword represents the level in the tree.*

Proof:

See generation formula of codeword in (1).

Theorem 1. *The codeword for each CA in public key certificate hierarchy is different.*

Proof:

First, let's prove that the codeword of each node in the same level is unique.

(i)In level =2, the codeword in level 2 can be expressed in 00(1)*. It is obvious that each codeword is different.

(ii) Assume level=k, where k>=2, each CA's codeword is different from others. Then, in Level k+1, each parent CA in level k assigns its codeword plus a new distinguished code generated from (1) to its subordinate CAs. Because each parent CA 's codeword is distinguished and each children CA's new codeword is uniquely generated with (1), new codeword for each CA in level (k+1) is distinguished.

Second, let's prove that the codeword of each node in a different level is unique. Assume it is observed that two nodes from different levels in the same diagram have the same codeword. According to Lemma 2, these two nodes should have the same number of '0' in their codewords; thus, these two nodes should come from the same level. It contradicts with the assumption.

In summary, we prove the a bove statement. □

Theorem 1 claims each CA node in hierarchy owns a unique codeword. Since the public key certificate identifies each individual CA node, to some degree, it can simply be represented with a codeword. Obviously, not only does codeword represent a unique position of each node in a hierarchical diagram, but more importantly to the purpose of certificate path searching, it also indicates a certificate path from root to this CA node. Codeword specifies CAs forming the certificate path. Each child CA inherits the certificate path from its parent CA. When two codewords are compared, the relative position between these two CAs in this public key certificate hierarchy will be found out, as we will see below. This eliminates the cost of guessing the possible direction to expand certificate searching or try exhaustive searching. After that, an optimal certificate path between two parties can be computed as the result of comparing two codewords.

Theorem 2 The maximum length L of codeword in a PKI hierarchy is between $k_1\lfloor \log_{k1} N \rfloor$ and $k_2\lfloor \log_{k2} N \rfloor$, where the number of subordinate CAs in each parent CA is in the range $[k_1, k_2]$, $k_2 >= k_1 > 1$, and N is the total CAs in that PKI hierarchy.

Proof:

As observed, under each CA, the longest distinguished codeword is that one in its last child. The length is in $[k_1, k_2]$ [11]. Since there are N CA nodes and average k children of each parent CA, the height of PKI hierarchy is between $\lfloor log_{k1}N \rfloor$ and $\lfloor log_{k2}N \rfloor$. As a result, the longest codeword in this hierarchy will be between $k_1\ log_{k1}N$ *and* $k_2\ log_{k2}N$. □

Let lcp (H_X, H_Y) represent the largest common prefix between two codewords H_X and H_Y.

As we discussed in section 5.2, the challenging issue in certificate path discovery is to find out which direction shall be taken for the expansion of the certificate collection and what certificates are needed for a complete certificate path. This issue can be solved with CCPD (Codeword-based Certificate Path Discovery) algorithm as described in figure 25.

CCPD will return both the relation of two compared codewords and the largest common prefix of two codewords. The returned results from CCPD are explained as follows,

Case (1), if $H_X \subseteq H_Y$, then two nodes are in the same branch of the certificate diagram. Party X is superior to Party Y. The relative location between X and Y is in Figure 22(a). If X wants to validate Y's certificate, then a chain consisting of forward certificates from X to Y is needed.

Case (2), if $H_Y \subseteq H_X$, then two nodes are in the same branch of certificate diagram, and Party Y is superior to Party X. The relative location between X and Y is in Figure 22(b). If X wants to validate Y's certificate , then a chain consisting of reverse certificates from X to Y is needed.

Case (3), If $H_X \cap H_Y = H_C \neq \phi$, the relative location between X and Y is in figure 22(c) and 22(d). To verify party Y, party X needed a certificate path consisting of a reverse certificate chain from X to C and a forward certificate chain from C to Y.

Case (4). If $H_X \cap H_Y = \phi$, certificate validation fails.

```
Function ccpd  (H_X, H_Y)
{
        int k=min(len(H_X), len(H_Y));

        H_C = lcp (H_X, H_Y);
        // get the length of the largest common prefix
         j = len ( H_C );
        if(j= =0) then
            return ('ϕ', null);  //return failure in verification
        if(j<k) then
             //return a list of two joint certificate paths
            return('∩', H_C)
        if(j==k) then
        {
            //two codewords from the same CA
           if (len(H_X)=len(H_Y)) then return('=', (H_X));
   // certificate C_X is located upper to C_Y
            else if(k== len(H_X)) then return ('⊆', H_C);\
   // certificate C_Y is located upper to C_X                else return('⊇', H_C);
        }
} // end of Function ccpd
```

Figure 25 Algorithm Codeword-base Certificate Path Discovery

As discussed above, the result from algorithm CCPD explicitly indicates the relative position of two CA certificates in the hierarchical diagram. Moreover, this algorithm can help find how many certificates are needed for this validation and what those certificates are. Therefore, the path search becomes deterministic other than blind. As an example, assume one source host certified by CA F with codeword X001 wants to verify a strange host with a certificate claimed to be signed by CA G with codeword

X0110. After the CCPD algorithm is run with these two codewords, the results are the "$\cap$" relation and H_C = X0. It implies that a list of CA certificates corresponding to a list of codewords, X001 (F), X0 (B), X (A), X011 (D), X0110 (G) must be collected for this activity of certificate validation. The optimal certificate path from source host to destination host is $KR_F<<KU_B>>KR_B<<KU_A>>KR_A<<KU_D>>$ $KR_D<<KU_G>>$.

The encoding strategy brings several benefits to a public key certificate system. (1) Codeword makes certificate searching independent from centralized entities. As shown above, codewords have already explicitly presented a straight certificate path from the root CA to its direct CA of the source host. If a certificate path is needed to validate a strange node, it can be computed only based on these two codewords from both certificates with CCPD algorithm; thus, there is no need to rely on centralized entities for how to expand certificate collection. (2) Second, codeword narrows the searching scope. Instead of searching an entire tree for a non-optimal certificate path between two parties, the problem becomes to collect and verify the certificates along an optimal, possibly the shortest, certificate path since the path can be determined in advance. It apparently eliminates much time in path searching, especially in ad hoc networks. (3) Without the codeword, the path will be expanded with only one certificate each time since there is no clue for other certificate information in this path except the closest one anytime. It is less efficient in certificate collection, especially in MANET environment. In our assumption, each host may cache a set of CA certificate information, and it likely contains more than one needed CA certificates; therefore, it is possible to add multiple certificates into this path at one time if we know which certificates are needed for a complete certificate path. That will certainly accelerate the certificate collection and improve the efficiency in an

Ad Hoc network environment. Apparently, encoding strategy helps make it possible to add multiple certificates in one time.

The encoding public key certificate hierarchy is a generic idea for reducing the cost in certificate path searching not only in Ad Hoc networks but also in conventional directory-based public key systems. It obviously accelerates path discovery since the path information is already contained in each certificate.

Remark

- Hierarchical PKI trust model is popularly adopted in one big company or one organization's network. Though we study certificate path discovery in hierarchical PKI model in the above, the proposed solution can be also easily extended to other trust models as well. One of the popular PKI models is bridge PKI trust model [34] [35] which meets the need for the intercommunication among different organizations or business partners. In this model, a trusted entity, called bridge CA, is usually introduced to realize the interaction among different PKI hierarchies with a single root. Bridge CA could be a standalone entity or one of CA nodes within one PKI hierarchy. Bridge node is certified by the connected two certificate hierarchies; in other words, there exist forward/reverse certificates between one bridge node and its contactors. Still, in bridge PKI trust model, certificate path discovery is needed to verify the authenticity and validation of one strange mobile host's certificate. Different from certificate path in one domain, the certificate path will contain the bridge node and the path will cross the different domains. Actually, the crossing

certificate path can be viewed as two pieces of the certificate path which are located in different domains; Thus, codeword-encoding approach can be applied into the certificate path discovery in the same domain and the discovery activity of the two pathes can be done in parallel. The discovery of the bridge node across two domains is out of the scope of this chapter, but it will be within our further study.

- As we notice, the length of codeword in a hierarchy affects searching efficiency and cost. With theorem 2, both the range of number of subordinate CAs and the total number of CAs in that hierarchical diagram affect the maximum length. One possible limitation for the proposed encoding approach is that the maximum length of the codeword in PKI hierarchy is sensitive to the tree structure. Therefore, some optimization is needed to shorten the codeword. For example, if it is a k-complete tree for which internal nodes have k children and all path from leaves to the root have the equal length, the maximum length of the codeword is the $[k\log_k n]$ since we only need to mark the position of the node in the tree. More efficient representation of the generic tree structure is under further investigation. However, in practice, each hierarchy has a limited number of CAs and the total number is not large. For example, the envisaged Federal PKI has 100 CAs [31].

5.4 On-demand Certificate Collection Protocol (OCCP)

As discussed in section 5.1, the central PKI service may be unavailable due to network dynamics in ad hoc networks. The assistance from other nodes therefore is expected for one mobile host who cannot establish a certificate path between its certificate and the being-verified certificate with its locally cached CA certificate

information. Our approach to discover a valid certificate path is 'on-demand' which means a query for CA certificate information is propagated only when there is a need for those certificates.

Since in the beginning, the source host has no idea which valid mobile host(s) has certificate information to satisfy the query of a certificate path partially or completely, a flooding mechanism is adopted to disseminate the request message and collect the certificate information in the path. Once a reply is ready to be sent back, it is desired that the route back to the source is ready as well. However, in ad hoc networks, the network topology can be changed dynamically and unexpectedly. The mobile hosts do not always stay in one location. Instead, they will move arbitrarily. Our approach does not rely on any routing protocol. However, similar to route discovery phase in reactive routing in MANET, a reverse path is established while the query packets are forwarded. The details are described below.

Section 5.4.1 will describe OCCP protocol and explain the behavior of each participant in this protocol activity. Section 5.4.2 will discuss the security vulnerability and countermeasures of OCCP.

5.4.1 Protocol Description

OCCP allows for dynamically collecting CA certificates along certificate paths derived from the CCPD algorithm as described in section 5.3. If the source host does not have the complete CA certificate information to validate a new certificate locally, it will generate a request message which contains a monotonically increasing Request ID, destination host's certificate, its own certificate, and expiration time of this packet, and

then broadcast this packet to all its neighbors. As shown in section 5.3, Codewords from these two certificates implicitly tell which certificates are needed for this validation activity.

When one intermediate node (IN) receives the request, it processes the request as follows:

(1) If it has desired and valid certificates to complete the request, then it will generate a reply and send back to the source; otherwise, go to step (2),

(2) If it has received the same request before by comparing the request ID, source/destination certificate pair, then go to step (4); otherwise, go to step (3),

(3) If it is able to fill out some part of the certificate path, then append its certificate chain into the request message. Record the request message and previous hop, and continue to propagate the request,

(4) After comparing the certificate chain with the saved one, if not changed, the IN discards the request message. Otherwise, it updates the stored request information, generates a reply with the partial certificate path, and sends it back along the shorter route to the source host.

Upon the receipt of the request, each IN will extract the information of queried certificate path by running the CCPD algorithm with the two codewords from certificates in request packet and determine if it has the valid CA certificate information to fill in the certificate path. As an example shown in Figure 26, since IN A and C cannot provide useful certificate information for the request from the source host, they simply pass the request to their neighbors. IN B is able to contribute two certificates in the certificate path; thus, it appends this information along with the request packet. Finally, host D completes the certificate path and generates a reply message back to the source host.

Once the source host receives the reply, it first checks the validity of four CA certificates as described in section 5.2. If that succeeds, it will use the collected path to verify the destination certificate.

Notice that, because of knowing the complete certificate path in advance with the help from the codeword in certificates in the request packet, both IN E and D are able to append more than one needed CA certificates and the certificate path discovery activity is done faster than one by one appending approach.

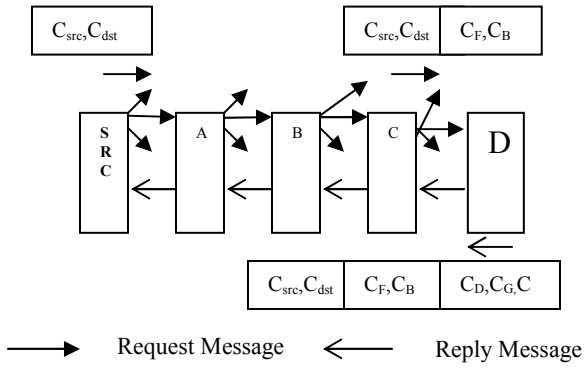


Figure 26 OCCP message transmission

Once one IN finds out it has such certificate knowledge to complete the request, it will generate a reply packet with the collected CA certificate information. Since each node has recorded the reverse path to the source with the previous hop, the reply packet can be sent back to the source without extra source path discovery. When each IN receives the reply with a complete path, it may save that useful certificate information locally. After certificate validation activity has been done, the related information is removed, and memory is released for other activities.

As shown in step (4) from the above, one IN may generate a reply with partial certificate path if it has seen such a request before. The reply with partial path is critical to reduce the protocol overhead in the network. The most significant overhead introduced by on-demand protocols is the flooding of request packets. Instead of flooding again possibly through the entire network, the reply packet is sent back along the reserved route among which the hop count is much smaller than the number of nodes in the entire network. The other benefit from the reply with partial path is that the IN could transfer the collected certificates in time prior to the network topology change which may break the link in the reserved route.

By adopting some optimized approaches within certificate path discovery, we are able to reduce the packet overhead and accelerate the path discovery in OCCP. For example, each active mobile node will make use of the promiscuous feature [5] to overhear bypassing OCCP packets. If packets contain new CA certificate information and the certificates are authenticated (see next section for message protection in OCCP), that useful certificate information will be saved into the local cache of this overhearing mobile host. Apparently, locally cached certificate information can help complete the request quickly and reduce the request propagation.

5.4.2 Protocol Correctness Proof and Complexity Analysis

For the purpose of analysis, we assume the network is static at the time (t_0, $t_0+\Delta t$) when one certificate discovery query is propagated in the network. Though practically the dynamic nature in MANET will change the network topology and break the reserved

routes, we usually rely on new-round query to provide better coverage if there is no correct reply within a predefined expiration time.

Lemma 4: *OCCP query will fully cover all reachable mobile nodes in mobile network.*

Proof:

Assume at least one mobile node X does not receive the OCCP request. Since node X is not separate from the network, node X's neighboring node(s) should have not heard the OCCP request either. In the same way, we can extend the statement to include more mobile nodes that have not received the OCCP request. Finally, because we assume the network is static and each mobile node is reachable within the period (t_0, $t_0+\Delta t$) when the certificate discovery activity is going on, we should be able to deduce that the source node has not received the request. Of course, such statement is not true since this request really came from the source host. Therefore, we prove all the reachable nodes will receive the OCCP query. □

Lemma 5: *OCCP query is able to collect the needed CA certificates from each mobile node in the network.*

Proof:

With Lemma 4, each IN will receive the request and check its local CA certificate repository for possible resolution. There are two possible results: one is that the attaching does not happen because either this IN does not have CA certificates to meet the need of this request or the CA certificate(s) known in this IN is(are) already in the attachment of the request packet; the other possibility is that this IN is able to provide a list of needed CA certificates (maybe only one certificate). It is meaningful to study if the source host

will eventually receive those newly attached certificate(s). The consequence of these certificate(s) is discussed in the below.

Based on the protocol, there are three possible results for this list of CA certificates, (1) this list of CA certificates is contained in a complete certificate path; (2) this list of CA certificates is propagated into IN which has seen this request before but the recorded request did not cover the certificates in this list; (3) this list of CA certificates is propagated into IN which has seen this request before and the recorded request did contain the certificates in this list; Of course, in the case (1), this list of certificates will be sent back to the source host; in the case (2), the receiving IN will generate a reply with a partial path containing this list of certificates and sent back to the source host along the shorter route as described in OCCP protocol. Finally, in the case (3), the request containing the list of certificates will be discarded, but the reply containing the list of same certificates either has been sent back or will be sent back to the source soon. In summary, the source mobile host will eventually receive the list of needed certificate(s) attached in the request packet. □

Lemma 6: *the CA certificates collected in OCCP is along the optimal path*.

Proof:

Since each IN will resolve the request based on the computing with CCPD algorithm which helps construct the certificate path without redundancy, the CA certificate in the certificate list will be in the optimal path. Therefore, the collected certificates are guaranteed along the optimal path. □

Corollary 1: *OCCP claims the capability of collecting the needed CA certificates along the optimal path from the entire network.*

Proof:

With Lemma 5 and Lemma 6, the corollary can be induced. □

Like on-demand routing protocols such as AODV, OCCP relies on the Hello packets to update connection with neighboring mobile nodes. As described in [19], the overhead caused by the Hello packets is a linear factor of the number of network nodes. For simplicity, we will ignore the common overhead from Hello packets but discuss the protocol overhead only caused by the OCCP RREQ and RREP packets.

Notation 4.4 Network Parameters

N	Number of mobile nodes
M	Average Adjacent of each node
D_i	Hop Number of transmitting reply packet
d	Distance between Source CA certificate and Destination CA certificate

Theorem 5 (Complexity of OCCP)

In worst case, each OCCP activity will generate O((M+d) N) protocol packet overhead.*

Proof:

With Lemma 4, all reachable mobile nodes will receive at least one request and one identical request packet can be propagated only once by each receiving mobile node to its

neighboring nodes as described in protocol design intent in section 5.4.1. Thus, the worst overhead caused with request packet is

$$O(N*M) \quad (5.4.2\text{-}1)$$

Assume the certificate distance of the requested two certificates associated with the two codewords is d. If one mobile node is in the middle of the reserved reverse route, it can forward up to d replies to the source because any IN will discard the reply packet with the certificate information having been sent back to the source by this IN. Besides, instead of flooding, the reply packet will be sent back along with one reserved path. Assume k set of distinguished reserved path are used to sent back the reply packets. The i-th path consists of D_i hops. Then the total worst overhead caused by reply packets will be,

$$d*D_1+d*D_2+\ldots+d*D_k=d*(D_1+D_2+..+D_k) <=O(d*N) \quad (5.4.3\text{-}2)$$

With (5.4.2-1) and (5.4.3-2), we conclude the total overhead in worst case in each OCCP activity is

$$O((M+d)*N) \quad \square$$

5.4.3 Security Threats and Countermeasures

Once an OCCP packet is transmitted into the air, it could be compromised either intentionally or unintentionally. Unintentional compromise is caused by transmit error, noise interference, etc. The intentional one is called an attack. The attack can be launched from externally or internally. External threat comes from outsiders who are not authorized in this network. Internal threat is formed by the compromised hosts. In this chapter, we consider that each mobile node has shared network key (Network ID) for authentication in medium access layer (MAC). If the packets are from outsiders who do

not know the shared key, they will be dropped within the transmission. Therefore, the challenge to the security of OCCP mainly comes from internal attackers or external attackers who are able to sneak the network key.

After a source host generates a request message which contains a request ID (R_{id}), the sender's certificate (C_s), the other party' certificate (C_d) and the expiration time (T_{exp}), it will compute and authenticate the checksum of the request message with algorithms such as KMD5, then use its private key to sign the checksum. Request message is represented by

$$RREQ = (R_{id}|C_s|C_d|T_{exp})|KMD5\ (R_{id}|C_s|C_d|T_{exp})$$

The signature with the private key prevents impersonating the source host for generating the request. The Request ID is a moronically increased unique sequence number for each mobile host. It is used to block obsolete request packets and prevent the reply attack.

The important security requirement for this protocol is to guarantee the reply information is correct, not fraudulent. In order to fool the source host and other hosts who will extract the useful certificate path information, some malicious hosts may intentionally either append fake certificate information or generate a fake response (RREP). As described in section 5.2, a complete certificate path consists of an ordered CA certificate list where every CA certificate can be verified with the public key from the previous certificate. Because the source has the correct public key of its CA, it can use its CA's public key to verify the certificate signed by its CA. By repeating the examination

in this way, the validation of one CA certificate path can be verified; therefore, the certificate path itself provides the authentication and integrity. Due to the nature of certificate path, any fraudulent certificate path can be easily detected because the attack does not know the private key of any authentic CA.

Instead of attacking the protocol itself, some malicious hosts may want to take advantage of the protocol message and affect the network performance. For example, they may intentionally generate fake request messages even with fake source certificates or destination certificates. By flooding many RREQs into the network, the attack may steal the bandwidth of the network and even worse lead to congestion. This is called DDoS which is a hot study topic in wired networks. To prevent the DDoS launched by an external host, a network ID may be used to stop the flooding in the beginning. Because an external host cannot present a valid network ID, its fake request will be dropped in the beginning. Further study is needed for DDoS attack launched by the compromised hosts in an Ad Hoc network.

Recently a few security mechanisms [38][39][40] have been proposed to protect ad hoc on-demand routing protocols such as AODV and DSR. Those state-of-art security mechanisms can be easily adopted to enhance the security strength in this certificate collection activity though OCCP is independent from the existing on-demand routing protocols. Further work will be performed with the integration between OCCP and those promising security mechanisms to protect OCCP protocol messages against potential attacks.

5.5 Implementation and Evaluation

Our OCCP design is implemented with network simulator ns-2 [13]. We use RSAref [14] as a cryptographic toolkit to provide various functions in OCCP for RSA digital signature and data encryption as described in subsection 5.4.

OCCP performance evaluation is executed in a network consisting of 100 mobile nodes roaming in a 2200x800 meter square with a reflecting boundary. Nodes in this network move based on the random waypoint model [15]. The data rate is 2 Mb/s. The moving speed of each mobile node is 10 meters/s. In our experiment, we test a CA hierarchical diagram with 30 CAs and degree k of CA hierarchical tree in 2. To see how cache improves the certificate path discovery, we run protocol simulation in the different initial number (i) of CA certificates in the local cache of each mobile host, namely 2 to 3, and 4 to 5, respectively.

In order to evaluate the performance of OCCP protocol in simulation, we use the following metrics: (1) the ratio between the median of the delay to discover a certificate path (*mddcp*) for a query and the certificate -Distance in the certificate path calculated by CCPD; (2) the median of overhead included in a complete certificate path (*mocp*) in each successful certificate chain discovery; (3) the median overhead of protocol packets over networks (*mopp*) in each successful certificate chain discovery. This overhead comes from the broadcast request packets and sends back reply packets. As we mentioned in section 2, intuitively there are two other approaches: one is the exhaustive approach, also called the straight-up approach and the other is fail-and-retry. The fail-and-retry approach certainly is the worse choice because it requires more query sessions and then generates

more traffic in the network. Thus, we implemented the straight-up approach and run performance comparison between these two approaches based on the above metrics.

Figure 27 describes how the certificate path affects the delay to collect a complete set of certificates. In general, the longer a certificate path, the more time it costs to

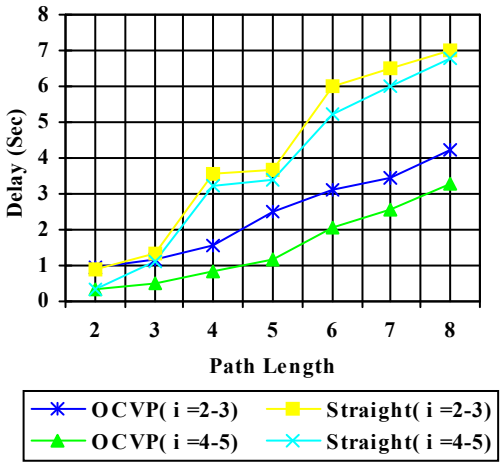


Figure 27 Ratio between MDDCP and Path Length

collect the needed CA certificates and complete the certificate path discovery. Collection activity will be done faster in OCCP than in the Straight-up approach when the path length becomes larger. For example, with the number of local cached CA certificates between 2 to 3, on average it takes 3.21 seconds to finish CA certificate collection for certificate path discovery in OCCP but 5.61 seconds for the Straight-up approach. This is due to the advantages from OCCP approach, such as unorderly certificate collection. The figure also explicitly illustrates that more CA certificates in local cache are able to quicken the certificate path discovery.

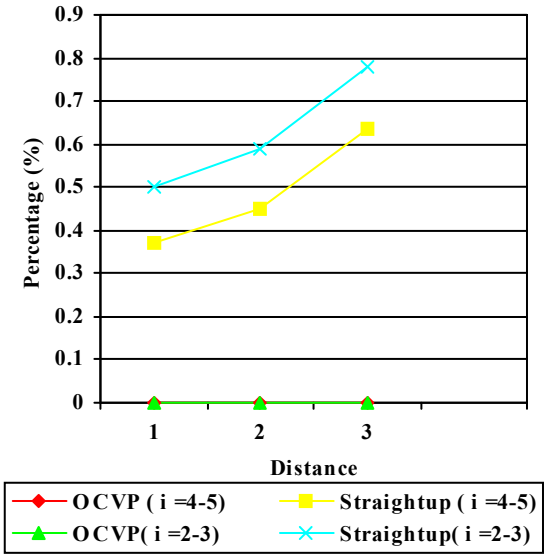


Figure 28 Percentage of Overhead in Certificate Path

The MOCP is caused by non-optimally collecting the CA certificates. As observed, with the same length of ceritifcate path, the MOCP is affected by the distance between the root and the one node closer to the root in the diagram. For example, in Figure 22, the distance means the smallest hops between the root and node Y in (a), between the root and node X in (b), or between the root and the joint of node X and Y. Apparently, the distance in (c) is 0 because the joint node is the root. In figure 26, the value in Axis X represents the distance. As shown in Figure 28, OCCP has no overhead in the collected certificate path. In the Straight-up approach, more than 50% of the complete certificate paths contain overhead. The percentage of such overhead increases as the distance becomes larger.

Therefore, the Straightup approach creates a non-optimal solution for certificate path discovery problem.

Figure 29 shows the average protocol overhead measured with packets including request messages and reply messages. The longer certificate path could require collecting more certificate information; thus, more packets are propagated into networks. Compared with the Straightup approach, OCCP has less protocol overhead, around 30% lower. The reason is that OCCP can find the complete path faster and reduce the further broadcast of RREQ. Besides, the caching also helps to reduce the protocol overhead. Because the

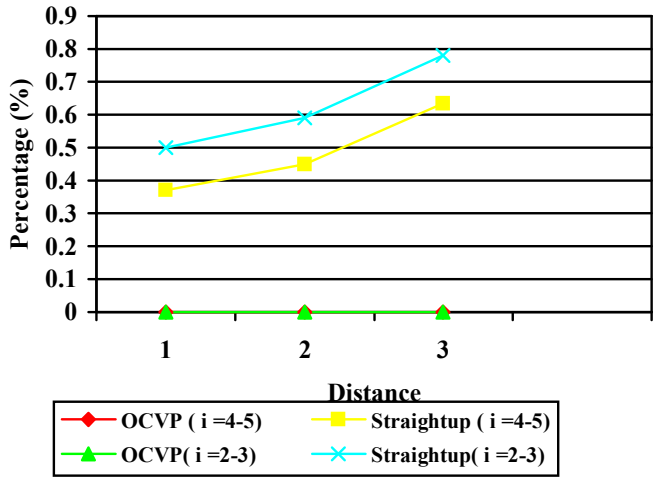


Figure 29 Ratio between MOPP and Path Length

more CA certificates in the local cache, the faster the certificate path discovery is completed.

5.6. Chapter Summary

Due to network dynamics in ad hoc networks, the conventional central PKI service may not be available to discover a certificate path for public key verification in the hierarchical PKI trust model. To find out an alternative approach, we explore a fully self-organized public key management system in mobile ad hoc networks, focusing on certificate path discovery. In this chapter, we study the problems in certificate path discovery in the absence of centralized certificate authority facilities and derive an efficient approach to discover the optimal certificate path even without centralized certificate entities; then we propose an on-demand protocol OCCP for CA certificate collection in ad hoc networks. The core of our approach is to encode each individual CA with a unique codeword which also identifies the identical position of this CA in the PKI hierarchy. The key benefit from this approach is that the needed certificates for a complete certificate path can be computed in advance, thus accelerating the certificate collection activity. Our experiments and simulations have shown positive results for our approach. OCCP is able to discover the certificate path within a small number of hops and with reasonable traffic overhead. As the cached CA certificates increase, the discovery activity is completed much faster. As observed in section 5.5, the distance of two codewords in source CA certificate and destination CA certificate are the main factors affecting the *MSH*. By adjusting the scope of flooding in the network based on the distance of two codewords in the request packet, we can achieve comparably acceptable MOPP overhead mainly caused by RREQ propagation.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Along with the growth of the Internet and the development of computing technology, network attacks are becoming more sophisticated and more powerful. Previous security mechanisms are not sufficient to guard the networks from the new threats. Neither would they protect the newly introduced network applications.

In chapter 3, we classify the variant attacks over the link state routing and concentrate on addressing the FIR attacks. The conventional security mechanisms are either too expensive or explicitly vulnerable to such attacks. A novel authentication mechanism is introduced to efficiently protect the LSAs and ensure that the malicious manipulation of the LSAs from the FIRs is detectable and the faulty origins is traceable. Further, we develop two practical tracing schemes with the different system environments and varied communication cost to trace back the FIRs based on the log information. In addition, the confidence values are computed from the schemes to reduce the false alarms.

The trustworthiness represented in probabilistic such as confidence value is able to more accurately reflect the current understanding of the status of one monitored router, and reduce the false alarms due to rough judgments. However, the probabilistic trustworthiness brings a new challenge of how to handle those uncertain routers, RUM. Motivated by the concern of the existence of the RUMs in the network, in chapter 4 an

extended Dijkstra shortest path algorithm was designed to provide the capability of determining the optimal path with the consideration of the routers' confidence values. Besides, we propose to build a secured and predictable virtual link over the selected RUMs in the network in order to leverage the resource from the innocent RUMs and increase the network reachability. Without sacrificing the security, the virtual link is constructed for data transportation with the efficient hop-by-hop and end-to-end verification; Moreover, a simple and efficient monitoring scheme is developed to continuously audit the behavior of the RUMs. The theoretical analysis proves the feasibility of this approach. The experimental evaluation shows the virtual link solution is practical and workable.

Addressing the issue of certificate verification from the hierarchical public key infrastructure in the dynamic and infrastructureless ad hoc networks, in Chapter 5 we explored a fully self-organized public key management system in mobile ad hoc networks, focusing on certificate path discovery. We studied the problems in certificate path discovery in the absence of centralized certificate authority facilities and derive an efficient approach to discover the optimal certificate path even without centralized certificate entities. Then we propose an on-demand protocol OCCP for CA certificate collection in ad hoc networks. The key benefit from this approach is that the needed certificates for a complete certificate path can be computed in advance, thus accelerating the certificate collection activity. Our experiments and simulations have shown positive results for our approach. OCCP is able to discover the certificate path within a small number of hops and with reasonable traffic overhead.

6.2 Future Work

In this section some potential future works are identified.

(1) In chapter 3, we introduce a preliminary formula to evaluate the confidence of one disruptive link based on the disruptive values. Future work should design a comprehensive evaluation formula which considers not only the present / real-time disruptive values, but also other factors, such as historical disruptive values.

(2) After the efficient protection of the link state routing protocol, I would like to implement several attacks to evaluate the performance and effectiveness of these mechanisms including detecting and tracing the disruptive links. In the future, formal analysis of this mechanism is needed to better understand its strength and weakness;

(3) In chapter 4, the proposed virtual link provides the security service in both hop-by-hop and end-to-end (e.g., between two TERs) to protect the data transmit over those uncertain RUMs. The simple and efficient security solution for the virtual link is also applicable as a potential protection mechanism to other path-preset relaying technologies in various layers, such as MPLS or AODV. In the future, we would like to integrate our secure virtual link mechanism with those network technologies to set up the protection points in each node along the relaying path and detect the faulty nodes.

(4) Besides certificate path discovery, certificate revocation checking is another critical process in PKI. Thus, our further research will study the problem statement and explore a self-organized and efficient approach for certificate refreshness and revocation in ad hoc networks without relying on the fixed infrastructure.

Bibliography

[1] E.H. Spafford, “Crisis and Aftermath.” Communications of the ACM, Vol.32, No.6, June 1989, pp.678-687

[2] CERT Coordination Center, “Melissa Macro Virus.” Cert Advisory CA-99:04, March 27, 1999

[3] CERT Coordination Center, “smurf IP Denial-of-Service Attacks.” CERT Advisory CA98:01, January 5, 1998

[4] C. Perkins and E.M. Royer. Ad Hoc On-Demand Distance Vector Routing. In P*roceedings of the Second Annual IEEE workshop on Mobile Computing Systems and Applications*, February 1999, 90-100

[5] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, T. Imielinski and H. Korth, eds.,Kluwer Academic Publishers, Norwell Mass., 1996, 153-181

[6] C.-K, Toh, A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing. In *Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, March 1996, 480-486

[7] Jean-Emile Elien. Certificate discovery using SPKI/SDSI 2.0 certificates, Master’s thesis, EECS Dept; Massachusetts Institute of Technology, May 1998

[8] Matt Blaze, Joan Feigenbaum, Jack Lacy, Decentralized Trust Management, *1996 IEEE Conference on Privacy and Security*, Oakland, 1996

[9] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks, *IEEE Network Magazine*, November 1999

[10] W. Stallings, Cryptography and Network Security, Principles and Practice, Prentice Hall, 2nd ed., 1998

[11] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, MIT Press/McGraw-Hill, 1990

[12] Srdjan Capkun, Levente Buttyan and Jean-Pierre Hubaux, Self-Organized Public-Key Management for Mobile Ad Hoc Networks, *Technical report EPFL/IC/200234*, June, 2002

[13] http://www-nrg.ee.lbl.gov/ns/

[14] http://www.spinnaker.com/crypt/rsaref/

[15] David Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publichers, 1996

[16] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, a. Qayyum et L. Viennot, Optimized Link State Routing Protocol, *IEEE INMIC* Pakistan 2001.

[17] Lewis, M., Templin, F., Bellur, B. and R. Ogier, "Topology Broadcast based on Reverse-Path Forwarding (TBRPF)", draft-ietf-manet-tbrpf-04 (work in progress), January 2002.

[18] ITU-T recommendation X.509 | ISO/IEC 9594-8: "Information techno;ogy – open system interconnection – the directory: public-key and attribute certificate frameworks"

[19] A. Herzberg and Y. Mass, Relying Party Credentials Framework, In *The Cryptographer's Track at RSA* Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings

[20] M. Blaze, J. Feigenbaum, J. Lacy, Decentralized Trust Management, In Proceeding. Of the *17th Symposium on Security and Privacy*, pp 164-173, 1996

[21] M. Blaze, J. Feigenbaum, J. Ioannids and A. Keromytis, The Key Note Trust-Management System,
Version 2. *RFC 2704*, IETF September 1999

[22] P. Jacquet, L. Viennot, Overhead in Mobile Ad-hoc Network Protocols, TechReport, INRIA, 2000

[23] S. Kent, R. Atkinson, Security Architecture for the Internet Protocol, *RFC 2401, IETF November 1998*

[24] S. Dusse, P. Hoffman, et. al., S/MIME Version 2 Message Specification, *RFC 2311, IETF March 1998*

[25] T. Dierks, C. Allen, The TLS version 1.0, *RFC 2246, IETF*, January 1999

[26] J. Katajainen and E. Makinen, Tree Compression and Optimization with Application, *In International Journal of Foundations of Computer Science* Vol. 1No. 4(1990), 425-447

[27] S. Zaks, Lexicographic Generation of Ordered Trees, *Theoret. Comput. Sci*. 10 (1980)63-82

[28] D. Zerling, generating Binary Trees Using Rotations, *J. ACM* 32 (1985) 694-701

[29] J. Stasko, J. Vitter, Pairing Heaps: Experiments and Analysis, In *Communications of the ACM*, March 1987

[30] R. Housley, W. Ford, et. al, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 2459, IETF, January 1999

[31] S. Micali, Efficient Certificate Revocation, Technical Memo MIT/LCS/TM-542b,

[32] S. Berkovits, S. Chokhani, et. Al, Public Key Infrastructure Study: Final Report. *Produced by*
the MITRE Corporation for NIST, April 1994.

[33] A. Arsenault, S. Turner, Internet X.509 Public Key Infrastructure: Roadmap, IETF Draft, 2003

[34] W.E. Burr, Public Key Infrastructure (PKI) technical Specification: Part A – Technical Concept of Operations, *Working Draft,* September 1998, available at http://csrc.nist.gov/pki/twg/baseline/pkicon20b.PDF

[35] John Linn, Trust Models and Management in Public-Key Infrastructures, *RAS Technical Report*, November 2000, available at http://www.rsasecurity.com/rsalabs/technotes/

[36] Carl A. Gunter and Trevor Jim. Policy-directed certificate retrieval. *Software: Practice & Experieince,* 30(15):1609-1640, Septmeber 1999

[37] N. Li, W. Winsborough, and J. Mitchell, Distributed Credential Chain Discovery in Trust Management, In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, Philadelphia, PA, USA, November, 2001

[38] B. Dahill, B. Levine, E. Royer, C. Shields. A Secure Routing Protocol for Ad Hoc Networks, *10th IEEE International Conference on Network Protocols (ICNP)*, November 2002

[39] Y. Hu, A. Perrig, D. B. Johnson. Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks, *MobiCom 2002*, September 2002

[40] M. G. Zapata. Secure Ad hoc On-Demand Distance Vector (SAODV) Routing *IETF draft-guerrero-manet-saodv-00.txt,* August 2002

[41] R. Barrett, S. Haar, R. Whitestone, Routing Snafu Causes Internet Outage, Interactive Week, April 1997

[42] X. Ao. DIMACS Report: Workshop on Large Scale Internet Attacks, November 2003

[43] K. J. Houle, G. M. Weaver, N. Long, and R. Thomas. Trends in denial of service attack technology. CERT Coordination Center Technical Report, October 2001

[44] J. Moy. OSPF Version 2, IETF RFC2178, 1997

[45] R. Coltun, D Ferguson, and J Moy, OSPF for IPv6, IETF RFC2740, 1999

[46] L. Lamport. Password authentication with insecure communication. Communications of the ACM, Nvember 1981

[47] S cheung. An Efficient Message Authentication Scheme for link State Routing, in 13th Proceedings of Annual Computer Security Applications Conference, San Diego, December 1997

[48] R. Perlman. Network Layer Protocols with Byzantine Robustness. PhD thesis, Massachusetts Institute of Technology, August 1988

[49] S. Murphy and M. Badger. Digital signature protection of the OSPF routing protocol. In Proceedings of the Symposium on Network and Distributed System Security (SNDSS' 96), February 1996

[50] G. Tsudik Message authentication with one-way hash functions. In Proceedings of IEEE INFOCOM' 92, May 1992

[51] K. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson. Detecting disruptive routers: A distributed network monitoring approach. In Proceedings of the IEEE Symposium on Security and Privacy, May, 1998

[52] B. Vetter, F. Wang, S. Felix. An experimental study of insider attacks for OSPF routing protocol. In IEEE International Conference on Network Protocols, October, 1997

[53] J. R. Hughes, T. Aura, and M. Bishop. Using conversation of flow as a security mechanism in network protocols, in IEEE Symposium on Security and Privacy, 2000

[54] A. T. Mizrak, K. Marzullo, and S. Savage, Detecting malicious routers, UCSD technical report 2004

[55] S.L. Murphy and M. R. Badger. Digital signature protection of the OSPF routing protocol. In IEEE/ISOC Symposiums on Network and Distributed System Security, 1996

[56] M. Steinder and A. Sethi. Probabilstic Fault Localization in Communication Systems Using Belief Networks IEEE/ACM Transactions on Networking, Vol. 12 No.5, Oct 2004

[57] N. Dawes, J. Aloft, and B. Pagurek, Network diagnosis by reasoning in uncertain nested evidence spaces, IEEE Trans. Commun., vol. 43, 1995

[58] T. H. Cormaen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, second edition, The MIT Press, 2001

[59] D Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, Resilient Overlay Networks, ACM Symp. On Operating Systems Principles (SOSP) October 2001, Banff, Canada

[60] Eriksson, H. Mbone: The Multicast Backbone. Communications of the ACM 37, 8(1994), 54-60

[61] A. Collins, The Detour Framework for Packet Rerouting, Master's thesis, University of Washington, October 1998

[62] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services, " in Proceedings of ACM SIGCOMM, August 2002, pp 61-72

[63] Z. Fu, H. Huang, T.Wu, S.F. Wu, F. Gong, C. Xu, and I. Baldine, "ISCP: Design and Implementation of An Inter-Domain Security Management Agent (SMA) Coordination Protocol", Proceedings, NOMS 2000, Pages 565-578

[64] S. Cheung and K. N. Levitt. "Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection", In Proc. New Security Paradigms Workshop, Cumbria, UK, September 1997

[65] http://www.openssl.org

[66] www.cs.bu.edu/fac/matta/Research/BRITE

[67] V. N. Padmanabhan, D. R. Simon. Secure traceroute to detect faulty or malicious routing. SIGCOMM Computer Communication Review, 33(1):77-82,2003

[68] T. H. Cormaen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, second edition, The MIT Press, 2001

[69] D. Moore, The Spread of the Code-Red Worm (CRv2), 2001. http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml

[70] H Huang**,** Felix Wu**,** "Cost-Reduced Secure Link State Routing with the Capability of Detecting Disruptive Links", submitted to IEEE/IFIP DSOM, 2005

[71] H Huang, Felix Wu, "Enhance Network Reachability with Confidence Routing and Secure Virtual Links", submitted to IEEE/IFIP DSOM, 2005

[72] H Huang, Felix Wu, "An Approach to Certificate Path Discovery in Mobile Ad Hoc Networks", *ACM CCS (COMPUTER AND COMMUNICATION SECURITY) SANS WORKSHOP* IN OCTOBER 2003

[73] www.blockbox.com/tech_docs/tech_overviews/network_security.pdf

[74] P. Oorschot, “Internet Security: Where Are We?” Presentation in April 29, 2005