An Integrated Solution to Protect Link State Routing against Faulty Intermediate Routers

He Huang
Nortel, 4008 E Chapel Hill-Nelson HWY,
RTP, NC 27709, USA

Shyhtsun Felix Wu
Computer Science Department, UCDavis,
Davis, CA 95616, USA

Abstract— **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 variant attacks. One of the severe attacks is from the faulty intermediate router (FIR) which intentionally compromises the LSA messages passing by and pollutes the routing tables of its downstream routers. Current security mechanisms are either too expensive or vulnerable to prevent this type of inside attack. To address the FIR attack, in this paper, we present a novel cost-reduced integrated solution which combines both fault-detection operations from routers and fault-tracing response from network management components. The significant properties of our system are the detectability of the abnormal behavior toward the LSAs and the traceability of the FIRs generating those bogus LSAs. The analysis of the memory requirement and the communication cost in our design demonstrate the feasibility and efficiency of our system.**

Keywords-component: faulty intermediate router, fault detection, fault tracing, confidence value

I. 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 of its neighboring routers and the connectivity situation such as connection status, link delay, etc. 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 [12]. 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 paper. 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 [1], or be subverted because of the weak passwords and the latent software vulnerability [2][3]. The faulty routers can attack the network in various aspects such as packet dropping, delay, reorder, alteration, denial of service, etc. 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 impacts on the network but are also not easily detected. In this paper, we focus on the solution to detect and trace the FIR attacks.

While some 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 attack, the expensive computation [12] prevents its wide adoption. The IETF recommends sharedKey-MD5 in OSPF v4 [4] and IPSec protection between the neighbors in OSPF v6 [5]. 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 paper, we present a novel cost-reduced integrated solution which combines both fault-detection operations from routers and fault-tracing responses from network management components to address the FIR attack. The significant properties of our system are the **detectability** of the abnormal behavior toward the LSAs and the **traceability** of the FIRs generating those bogus LSAs. Specifically, we first introduce an efficient authentication mechanism, by which the originating router authenticates its ***M*** subsequent LSAs with a session key without prior key exchange; Each router in the network is required to log the important fields from unverified LSAs. If bogus LSAs are detected with the lately disclosed session key, either centralized or decentralized fault-tracing operations from the network management components are conducted 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 solution 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.

The rest of the paper is organized as follows. In section II, we present the system models. In section III, we introduce new and efficient operations in each router to detect the mechanism of credit based authentication, then propose two tracing schemes running on network management components to search the FIRs. In section IV, we conduct the analysis of the memory requirement and the communication cost. In section V, we review the related work. Finally, we conclude the paper and discuss some potential future work in section VI.

II. SYSTEM MODEL

Network Model This work considers the network consisting of routers running link state routing protocols, such as OSPF. A network is described with an directed 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 random number is used as the session key to authenticate the router's own LSAs. We will discuss it 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, and alter the routing packets sent by other legitimate routers. The FIR is able to masquerade other routers 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 paper will not work. We assume at least a good path with no FIR existsing between any pair of routers in the network.

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 manipulations happened. Through identifying the disruptive link, we can narrow the location of the FIRs.

Notation 1

Symbols	Meanings
K_s	A session key used to authenticate the originating LSAs
M	The number of LSAs to be authenticated with one session key
(Id, Sq, T, Pl)	A brief representation of one LSA packet, where Id: the identity of the originating router. e.g., IP address Sq: the unique sequence number of this LSA T: the timestamp of this LSA Pl: the payload data of this LSA
MAC_i	The keyhashed value of LSA i computed with session key K_s
$Sg_{pri}(P)$	The signature of one packet P with the private key
H(P)	A function to compute message authentication code of packet P
$H\_K_s(P)$	A function to compute message authentication code of packet P with session key K_s
δt_i	The propagation time of LSA i from the originator to the intermediate receiver
Hv_i	The hash value of LSA i, $Hv_i = H(Id \mid Sq_i \mid T_i \mid Pl_i)$
Ng_i	The neighboring router who forwards the LSA i

Design Goal Our goal is to develop an efficient integrated solution to protect 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.

III. PROTECT LINK STATE ROUTING AGAINST FIR

A. *Efficient Fault-Detection Operations in Rrouters*

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, then 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 authentication scheme.

Each time a router generates a session key K_s which is used to compute the hash value of its own LSA_s with symmetric authentication scheme, for example, by using Keyed MD5 algorithm. The LSA packet, e.g., (Id, Sq_i, T_i, Pl_i), is broadcast into the network but the hash value MAC_i is kept locally, where $MAC_i = H\_K_s(Id \mid Sq_i \mid T_i \mid Pl_i)$. The session key will not 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} (P))), where P = H(Id| T| K_s | $\sum_{i=1}^{M}$ (Sq_i, MAC_i)).

Similar to the concept of 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 if the credit is available for this originator. The routing table will be updated as well if needed. 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 value Hv_i of the LSA instead of the entire LSA packet. Besides, the sequence number, the propagation time $\boldsymbol{\delta t}$, and the ***Ng*** will be recorded into a LSA log table LLT as described below,

LSA log table ***LLT*** (***Id***, ***Sq***, $\boldsymbol{\delta t}$, ***Hv***, ***Ng***)

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 in the SKDM. Specifically, the receiver first picks up a tuple, e.g., (Id, Sq_i, δt_i, Hv_i, Ng_i) from LLT and key-hashes the Hv_i value with the session key Ks in the SKDM, then it compares the result with the value from the hash of the MAC_i value associated with the same sequence number in the SKDM. 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 originator 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 one originating router and one receiver.

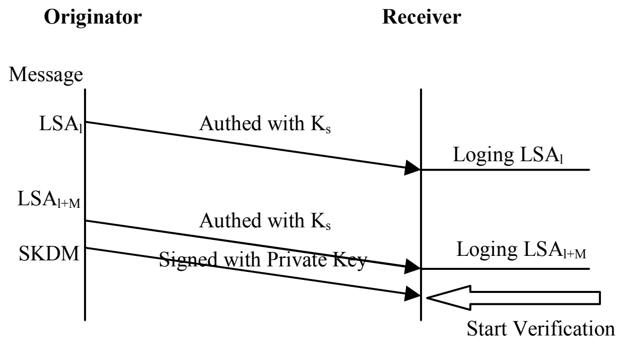

Figure 1 An example of session key usage and message processing

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 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. The credit will be added back with the same number of reported bogus LSAs as described below. In the next section, we will present two practical tracing schemes to locate the FIRs.

B. 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 happening 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.

1) *Cemtralized 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. 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** (**ANT**) 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 if the matched bogus LSA 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. The correctness of this tracing idea is present in the appendix section.

Ctrace() is the centralized algorithm to trace the FIRs. The CMS conducts the comparison with the received neighbors' bogus log tables by starting any affected router, and goes through all the affected routers in the **ANT**. 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. Those values are 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. Section *C* 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 isolate the adjacent FIRs.

While most of the time, 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.

```
1. Generic Verification Procedure

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

//The function check 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.t is larger than pTp.t AND Tp.H is equal to pTp.H
 (1.6)            AND Tp.Neig <> pTp.Neig   //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.Neig Є V1)
(2.7)         THEN
(2.8)             e(i, tp.Neig).disruptive
                      = + CheckPreviousRouter(tp, BLLT(tp.Neig);
(2.9)         ELSE e(i, tp.Neig).disruptive ++
(2.10)        ENDIF
(2.11)    ENDFOR
(2.12)ENDFOR
```

Figure 2 Centralized Tracing Algorithm

Theorem 2 (Complexity) In the worst case, the runtime of the verification in centralized tracing algorithm is Θ(|E|+|V1|*M*logM) if 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 report up to M bogus LSAs. With the binary search technique, the runtime for checkPreviousRouter() is Θ(logM). From the algorithm in Figure 2, 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. Assume 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).

□

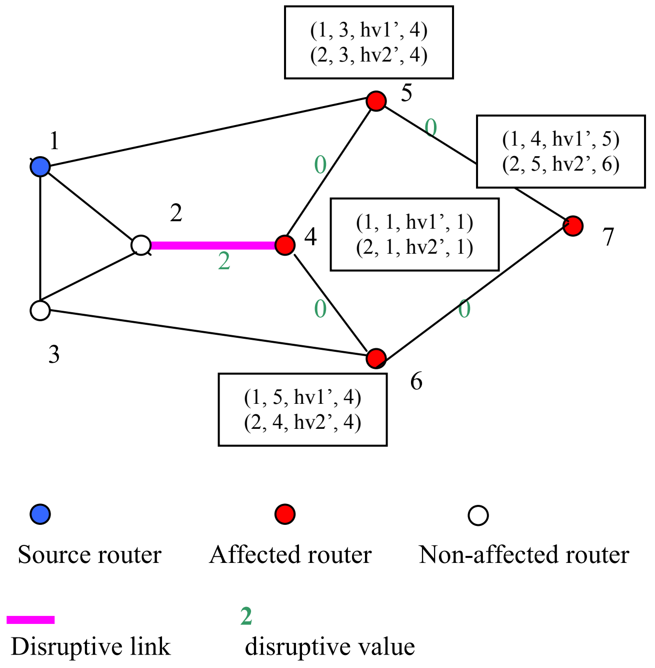

Figure 3 An example of ANT

As an example in Figure 3, router 4 is a FIR and it compromises two LSAs from router 1. Four fields of each tuple of the compromised LSA, (Sq, δt, Hv, Ng) are displayed orderly in the squares. After the session key is disclosed by router 1, routers 5, 6, and 7 detect the bogus LSA log and report to the CMS. In the meantime, assume FIR router 4 also reports a bogus LSA log 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 router 5 and 6. Upon continuous tracing from the routers 5 and 6, the disruptive values over those links are kept as 0 because the 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 doesnot report any bogus LSA. Eventually, the tracing algorithm returns finding one disruptive link between router 2 and 4, and the disruptive value is 2.

2) *Decentralized Tracing Scheme*

In some situation 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 that 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 generic idea into the scenario with collusive FIRs.

We called an affected router as 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 LSA 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 LSA 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 set 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 splitted *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 shall 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 up the reason for this link where no response is received or the response fails in verification and wraps the received evidence information and send 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 empty log table signed with its private key and announce 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.

```
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.neig
(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 Decentralized Tracing Algorithm

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 form 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 good router. However, the 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 $\Theta(M*\log M)$ and $\Theta(M\log^3 M)$ if 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 $\Theta(\log M)$. On one hand, if no matched 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 $\Theta(M*\log M)$. On the other hand, if found all matched, 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 $\Theta(\log M)$. Thus, the upper bound of the decentralized verification algorithm is $\Theta(M\log^3 M)$.

□

As an example, consider figure 2 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, and 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 claim 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 hops away from the responder, where k is the upper-bound number of collusive FIRs in the network, and check if those evident information can support the innocence of the responder through the same method described in the above. This would, however, substantiately 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.

C. 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 other are inadvertently and temporary. In real network, the occasional bad network situation such as network congestion or link noise could lead to the rare and very mild packet discrepancy. A straight removal of such disruptive links may lead to the 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 followed,

$$\text{ConfidenceValue} = 100\% - \frac{d}{N}$$

Where, d is the disruptive value associated with this link and N is the number of LSAs 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 on the network routing. How to handle the disruptive link with less 100% confidence value is out of the scope of this paper but will be in our future research.

Though link state routing protocol is well known to its self stabilization, the FIRs are able to modify the LSAs at their wills and thus seriously impact the network stability, even if the source router notices the change of the LSA and fights back with updated LSA. 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 voiding the possible false alarms. The removal of the disruptive link requires table changes in each routing table, but it reduces the possibility for the FIRs to pollute the rest of the network further and thus achieves more stable network environment.

IV. 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 5 fields in the log table as described in section III-A. 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 [4]. Thus, each tuple of the log table requires 235 bits, around 30 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*30). 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 750 Kbytes which is much small under 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 the each affected router to the CMS. As described in section III-B.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 receive 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). In the worst case where the existence of the FIRs compromise all LSAs, each router in the network will submit its log table for k originators, the network will see a total traffic with O(N*k*M*30) though in reality, the distributed submission will require much less network bandwidth. As an example, let the values of M and N as the above worst and pick k as 10, the total traffic to trace the FIRs is 7.5 Mbytes in the worst case. Such traffic is quite trivial in 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 exchange of the bogus LSA log tables between the requesters and the responders will not flood 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 same fields of bogus LSA messages are exchanged as those in centralized scheme. In the worst case all received LSAs are bogus from one UR, therefore, the size of one request will be 30*M. It is obvious that the response from the responder will include up to 30*M bytes for either the matched log table or the evidence approval log tables. Overall, totally 90*M bytes log information is passed by over one link. In the worst case when the FIRs are so strong that they manipulates all LSAs passing by, then all routers in the network will conduct the query for all other originators. Thus, the overall communication cost in network is O((N-1)*90*M). As an example, let's pick the same values of N and M as those in the above, the total traffic to trace the FIRs over one link is 2.25 Mbytes.

V. RELATED WORK

Recently quite a few researches have being done on securing link state routing against malicious operations and discovering the faulty origin after the violation of the routing and forwarding services.

The earliest security work on link state routing protocol can be traced back to Perlman's seminal work[8]. In her PhD thesis[8], Perlman 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 [13].

Murphy etc [9] 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 be sent out. However, recognizing the high expense with the digital signature in each routing packet, they suggested either adding the extra hardware or delay 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, etc [10] present an efficient authentication mechanism with one-way hash chain function which was originally developed by Merkle, Lamport [6]. By hashing the time and a secrete 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 can not derive the rest of 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 with the order of magnitude cheaper, but as mentioned by the author, some limitations such as authentication for multiple-values link remains.

To solve the limitations mentioned in [10], Cheung [7] 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. Though his proposal is similar to our distributed scheme, our approach presents some advantages. One of them is the localized data exchange. The data exchange in our approach mainly happens in the affected routers thus avoid the global impact due to the flooding; besides, our proposal has the capability of acquiring the confidence values to measure the severity of the disruptive link so that we're able to reduce the possibility of false alarms.

Bradley, etc [11] 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 proceeded to discover the faulty origin. However, WATCHER suffers variant limitation as detailed in [13]. More recently, Mizrak, etc [14] 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 that how the detection protocols countermeasure the compromised routers in the network.

VI. CONCLUSION AND FUTURE WORK

In this paper, 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 memory requirement in our proposed mechanism is small and the communication cost is acceptable.

We can identify several future work. In the first, we would like to do formal analysis of the effectiveness of this mechanism in order to better understand its strength and weakness. Second, we would like to modify our mechanism into infrastructureless and no central administrative mobile ad hoc network where every node is roaming and self-organized. We believe that the properties built in our decentralized tracing scheme such as distributed and localized make it suitable to work in such dynamic network environment. Besides, the confidence value is able to give the flexibility in measuring the trust under more complicated wireless network environment.

ACKNOWLEDGMENTS

We would like to thank Anne Nilsson, Ning Peng, Rudra Dutta, and Alexander Keller for helpful comments. We appreciate Gary Weinberg for proof-reading. We also thank the anonymous reviewers for many valuable comments.

REFERENCES

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

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

[3] 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

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

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

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

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

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

[9] 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

[10] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the cost of security in link state routing. In Proceedings of the Symposium on Network and Distributed System Security (SNDSS'97), February 1997

[11] 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

[12] 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

[13] 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

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

APPENDIX

CORRECTNESS AND COMPLETENESS OF THE TRACING SCHEME

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,

- **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;
- **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;
- **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.

- **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,

- 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.
- 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.
- 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,

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

□