

# On Providing Confidentiality in Link State Routing Protocol

Dijiang Huang  
Computer Science & Engineering Dept.  
Arizona State University  
Tempe, AZ 85287-8809, USA  
Email: dijiang@asu.edu

Amit Sinha, Deep Medhi  
Computer Science & Electrical Engineering Dept.  
University of Missouri–Kansas City  
Kansas City, MO 64110-2499, USA  
Email: {amit.sinha, dmedhi}@umkc.edu

**Abstract**—In current network routing domains, routing information exchange usually lacks protection based on confidentiality. This makes network routing vulnerable to a variety of security attacks. In this paper, we present a framework to provide confidentiality for a link state routing protocol. This framework involves creation of a trust structure among routers as well as key management. Routing information is encrypted so that it can be accessed only by authorized routers. We present an implementation framework for our approach by extending *Open Shortest Path First* (OSPF), a commonly deployed link-state routing protocol. Based on our performance assessment, we have found that the additional cost in implementing our scheme has fairly moderate impact on the overall performance.

## I. INTRODUCTION

A key function of a link-state routing protocol is to exchange information about link-status in a network. In most cases, this information is exchanged in a clear-text mode, although authentication as an optional feature has been proposed; for example, see RFC2154 [1] and RFC2328 [2]. On the other hand, the issue of confidentiality of information communicated during routing information updates has not received much attention. Recently, for the most commonly used link-state routing protocol, OSPF, confidentiality has been proposed in an Internet draft [3]; this confidentiality is provided through IPSec. This approach has two implementation difficulties. First, multicast is often used by link state routing protocols for flooding routing information; on the other hand, secure multicast of IPSec is still under development which is itself a difficult issue that IPSec needs to conquer. Second, IPSec provides only IP packet level authentication. Relying solely on IPsec can still allow a subverted router to forge routing information without being detected.

We have recently proposed a secure routing infrastructure that uses a link state routing protocol environment [4]. In this paper, we present an overview of our approach. We discuss here the issue of confidentiality in this approach and show how OSPF can be extended to incorporate this approach. Briefly, our approach depends on building trust among routers using a secure channel through a *many-to-many* secure group keying scheme [5]. We present OSPF-E which is an extension to OSPF by incorporating encryption and extended information. We also present an analysis on the impact on network router's memory and Central Processing Unit (CPU) to show that

the additional cost of introducing confidentiality through our extension into network routing is fairly moderate.

The rest of the paper is organized as follows. In section II we present the trust relation and key management framework for a link state routing protocol. In section III, we present the extension proposed for OSPF. In section IV, we discuss the results of our performance assessment of a single router under the proposed scheme. Finally, in section V, we present the conclusion.

## II. SECURE ROUTING INFRASTRUCTURE

In order to provide confidentiality on routing information generated by a router from a subset of routers, we need to build up trust among network routers. The trust relations with other network routers helps a router to determine the network service level and trust level to which the received/transmitted routing information belongs. It also helps to build the secure key distribution relationship among network routers. When network trust relation needs to be rebuilt, the trust structure can help reconstruct the trust relationship. In this paper we do not discuss how to detect the network external intrusions and insider attacks, we assume that there exists a system to detect intrusion, and we can rebuild our trust relations as needed.

In [4], we have presented a secure routing framework. From an operational point of view, this framework requires the following elements in the trust infrastructure an example is shown in Fig. 1):

- (1) Certificate server/Policy server (CSPS): The CSPS is used in building trust (through distribution of keys or key certificates) and distribute network security policies within the network routing domain. There may exist multiple CSPSs if routing domain is large.
- (2) Trust Group (TG): A TG is formed by a group of routers. Usually, we define routers within the same area as fundamental TG members. For an extremely large area, we can divided a TG into several smaller TGs.
- (3) Trust Group Leader (TGL): A TGL is the representative of its trust group and takes trust actions for its group. The leader is at a higher level than its group members. There may be multiple TGLs for one TG. Some of these TGLs may share responsibilities or act as backups. For example, in an OSPF routing domain, we may define the

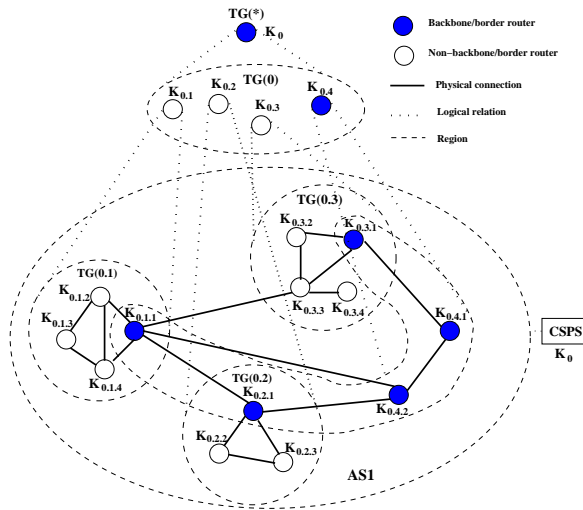


Fig. 1. Key allocations of OSPF-E hierarchical trust structure

area border router and Designated Router (DR)/Backup Designated Router (BDR) as the TGL.

- (4) Multiple levels trust structure is built based on trust relations among routers. In our example, routers within an OSPF area form the bottom level trust group, all TGLs for each bottom level trust group form a logical higher level trust group, and so on. A trust group is represented by *dot notation*, one level to another level in descending order. For example,  $TG(1) \rightarrow TG(1.1)$  where the top level is represented by  $TG(*)$ .

Utilizing the secure *many-to-many* group keying scheme presented in [5], a TGL can derive all possible subgroup keys below its level. Each TG member can derive all subgroup keys within its level, in which the subgroup is involves. Thus, the secure *many-to-many* group keying scheme makes it convenient to fulfill all types of communication and to reduce the key management communication overhead. Due to page limitation, refer to [5] for technical details on the secure *many-to-many* group keying scheme.

### III. OSPF-E

In this section, we discuss how to extend OSPF protocol to OSPF-E that incorporates functionalities needed for the trust structure described above. To achieve this, we use opaque LSA option in OSPF; see RFC2370 [7]. An opaque LSA consists of a standard LSA header followed by application specific information; it provides a generalized mechanism to allow for future extensibility of OSPF. We introduce a key numbering scheme in order to identify the trust level of routers and the key that has been used to encrypt routing information. We also include here processing of the LS Update packet which is critical to the protocol. This section also includes the details of the LSA packet format that is being proposed.

#### A. Key numbering scheme

We use *dot notation* to present our key numbering scheme. The *dot* separates the levels of the hierarchical key structure.

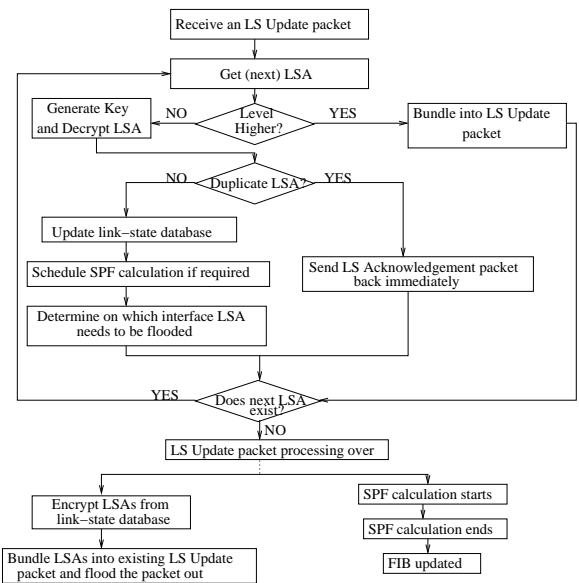


Fig. 2. Flow Chart depicting the OSPF-E processes initiated on receipt of LS Update packets

An OSPF-E key numbering example is shown in Fig. 1 where both the TG and the keys are labeled by the dot notation. The leading number stands for the top level of the group. Succeeding numbers following the dot represents sub-group, sub-sub-group members, and so on. Thus, it can be conveniently represented as: *root.group.sub-group.sub-sub-group...*

#### B. OSPF-E and LSA packet processing

In order to incorporate the encryption/decryption process into the LS Update packet process, the flow chart provided by Shaikh and Greenberg [8] for standard OSPF processing requires modification. We present the new flow chart in Fig. 2 where we show the initiation of the OSPF-E processes once an LS Update packet is received by a router. It can be observed that on receiving an LS Update packet, the router processes the LSAs contained in that packet one-by-one. It looks up the LSA packet header to find at which level the key resides in. If the key (the one used to encrypt the LSA) resides at a higher level than the router, then it simply bundles the LSA into the LS Update packet. If the key belongs to the same level or lower, the router will either use its own key or generate the subgroup key to decrypt the LSA. Once the LSA is decrypted, it is checked for duplicates. Furthermore, OSPF-E updates its LS database for every new LSA it receives. The flooding of LSA is similar to OSPF. It also needs to schedule the routing computation module.

Before the router creates the LS Update packet to be sent on its interfaces, it encrypts the information using various keys. The choice of the encryption key for a particular LSA depends on the level of the router and also the scope of the information as defined by the network policy. Once all the LSAs are encrypted, they are bundled into the LS Update packet and flooded through outgoing interfaces.

Table I  
Opaque LSA Header

1					31
LS age		Options		LSA Type	
OType	Pri	EType	Key ID		
Advertising Router					
LS sequence number					
LS checksum		Length			
LSA sub-header					
...					

Table II  
Opaque LSA Sub-header

1					31
Format	Levels (n)	Num of bits			
...					
Var 1 ~ 16 * (n - 1) ...					
...					
SKey len	...				
Var/Encrypted session Key ...					
...					
Encrypted data					
...					

### C. OSPF opaque LSA

Opaque LSA [7] provide three LSA types - type 9, type 10, and type 11. These three LSA types are used for advertisement within a network, an area and an autonomous system, respectively. We define the opaque LSA format to provide confidentiality for each of these three types of advertisements. In addition to these three types of opaque LSAs, we can define other types of opaque LSAs for a particular usage, such as key distribution LSA, routing control LSA, and so on.

In our proposed scheme, authentication is at the LSA level. The modified opaque LSA header is shown in Table I.

- (1) Options: In RFC2370, “O” bit of option field is set in the database description packet to indicate that the router is opaque capable. In the LSA’s header, we use the S bit in the same position to indicate that confidentiality is provided.
- (2) LSA type (8 bits): Three types of Opaque LSAs are already defined (type 9, 10 and 11), each of which has different flooding scope. We provide confidentiality for these three types.
- (3) OType (8bits): This field specifies the LSA type encrypted. Various OTypes are defined as follows: 1- Router-LSAs, 2- Network-LSAs, 3- Summary-LSAs (destination to network), 4- Summary-LSAs (destination to AS boundary routers), 5- AS-external-LSAs.
- (4) EType (8 bits): The encryption type specifies the encryption/decryption method used, such as DES, 3DES, AES, and so on.
- (5) Key ID (8 bits): This field specifies the type of cryptographic scheme used to encrypt a propagated LSA. Such a scheme can be either shared key-, or public key-based.
- (6) LSA sub-header: The LSA sub-header follows the opaque LSA’s header and identifies the encryption/decryption key that is used.

We also propose changes to the LSA sub-header (Table II) which is discussed below:

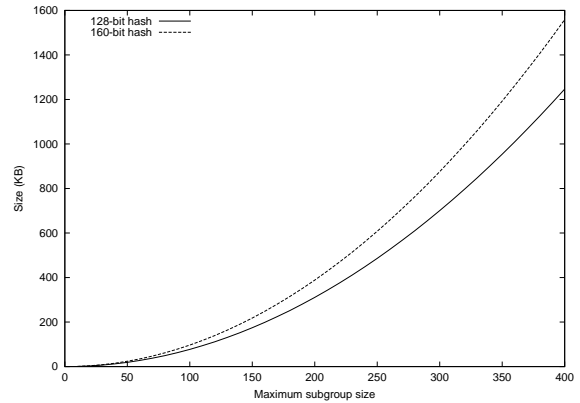


Fig. 3. Memory requirement for key storage

- (1) Format (16 bits): This field specifies what type of key is used: it can be a Global/level/sub-group key or an individual key.
- (2) Levels (n): This field represents the number of levels between the key used and the top level key.
- (3) Num of bits (16 bits): Based on our proposed master key scheme, this field specifies the number of concatenated hash values.
- (4) Var 1 ~ 16 \* (n - 1): This field identifies the location of the key in the hierarchical key structure. It contains information of each level from top to the point where the key is located and n is specified in the “Levels” field.
- (5) SKey len (8 bits): This field specifies the length of a session key when it is used.
- (6) Var/Encrypted session key: The session key is encrypted at the individual or global/level/sub group key. The length is variable which depends on the field “SKey len”.

## IV. PERFORMANCE ASSESSMENT

Any implementation of our proposed scheme would require additional CPU usage at each router due to key generation, encryption and decryption functionalities. In this section, we analyze the storage requirement and the new CPU processing overhead at each router in order to understand overhead incurred. First, we have done a worst case analysis for both the storage complexity and the CPU time complexity. For this work, we have extensively used the information available in the form of benchmark [9] for speeds of various hashing and encryption/decryption algorithms and the relevant data reported by Moy [10], [11]. We have also benefited from results and equations given by Shaikh and Greenberg [8], and by Basu and Riecke [12], in formulating our assumptions and in deriving our results.

### A. Storage Requirement

The analysis by Moy [10] shows that a router with 2 MBytes of memory is good enough to support 10,000 external LSAs (requiring 640 KB of memory space). In our scheme, apart from storing the LSAs, the router also needs to store a set of KGs and a set of most frequently used keys. Now, if a routing

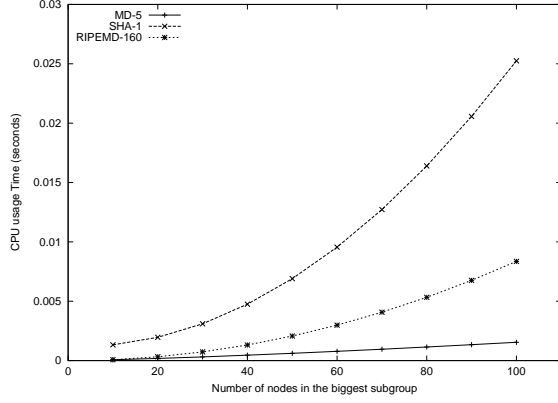


Fig. 4. CPU usage by hashing algorithms for Generating the decryption key for LSAs

domain consists of “ $N$ ” routers, then a simple approach would lead to formation of “ $2^N - 1$ ” subgroups. Each router in the domain would need to store its own key, the global group key and keys of all the subgroups that it belongs to, leading to a total of “ $2^N - 1$ ” keys. Hence the storage complexity grows exponentially with the number of routers in the routing domain. To avoid this extreme growth rate the dynamic keying approach based on Hamilton cycle was proposed [5]. This approach ensures that the storage complexity for each router grows quadratically with the biggest subgroup size in the routing domain.

Fig. 3 represents the storage requirement for hashing algorithm that generates 128-bit (MD2, MD4, MD5 [13]) and 160-bit (SHA-1, RIPEMD-160 [13]) key. It is advisable to have a subgroup size equal to the size of an OSPF-E area. This is because dividing an OSPF-E area into multiple groups will generate a lot of communication overhead, since the communication between two subgroup will have to be routed through the higher level routers. Moy’s survey of vendors [11] states that maximum number of routers in a single area in a real OSPF deployment is 350. But, if all the 350 nodes are put in a single group, the memory required on each router to store the KGSSs would be around 1 MBytes (Fig. 3). If the area with 350 nodes is divided into three groups containing 100 nodes and the last group consisting of the remaining 50 nodes, then the memory required to store all the keys by each router reduce to around 100 KB or less.

## B. Router CPU

Router CPU utilization is an important factor to consider under our suggested framework. The analysis presented here is a high level analysis. From the flow chart in Fig. 2 it can be observed that the extra complexity introduced into the existing system is due to a) generation of the decryption key, b) decryption of LSAs while processing the LS Update packet, and c) encryption of LSAs retrieved from the LS database before bundling them into an LS Update packet for flooding. In the worst case, there can be “ $N^2$ ” LSAs contained in a single LS Update packet, in a network of  $N$  routers running the OSPF

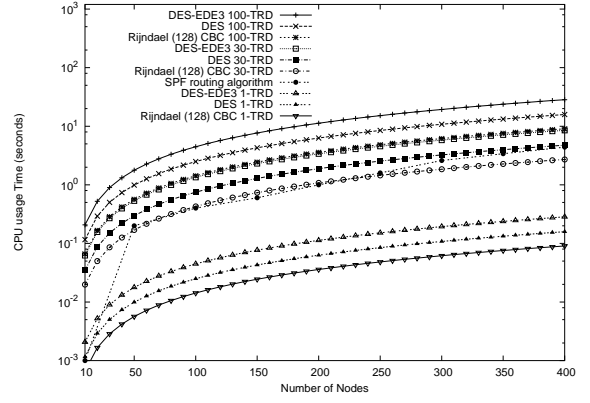


Fig. 5. CPU usage by various algorithms for Encrypting LSAs

protocol. Throughout our analysis, we assume that we have an optimal network architecture as defined by Aho and Lee [14]; this reduces the number of LSAs from  $O(N^2)$  to  $O(N^{4/3})$ . However, before we delve further into the processing time requirement at a router’s CPU, we first introduce additional time factors in the equation given by Basu and Riecke [12]. The time taken to process an OSPF-E packet ( $t_{er}$ ) is given by

$$t_{er} = \begin{cases} t_{ph} + l * t_{eup} & \text{if LS Update packet} \\ t_{ph} & \text{otherwise} \end{cases}$$

where  $t_{ph}$  denotes the fixed time to process any OSPF-E packet, similar to the approach in [12]. Here “ $l$ ” denotes the number of LSAs in an LS Update packet. Time  $t_{eup}$  to process a single LSA in the LS Update packet is given by

$$t_{eup} = \begin{cases} t_{up} + t_g + t_d & \text{if key belongs to lower level} \\ t_{up} + t_d & \text{if key belongs to same level} \\ 0 & \text{if key belongs to higher level} \end{cases}$$

where  $t_{up}$  is the LSA processing time as defined in [12],  $t_g$  is the time taken for deriving the lower level key, and  $t_d$  is the time required for decrypting the LSA information. We assume that the time required to find the level of the key used for encryption, by looking up the LSA header, is accounted for in the fixed processing time. We assume that the processing time for an LS Update packet is the dominant factor. This assumption is made on the fact that there are additional functions to be carried out with each LS Update packet (Fig. 2). Basu and Riecke in their work [12] have approximated the processing time for other packets (HELLO, Database Exchange, LS request and LS Ack) to 0; we make the same assumption.

Since we have some information about  $t_{up}$  [8], we present here the new overhead introduced due to the two new time factors. In order to see how “ $l * t_g$ ” and “ $l * t_d$ ” varies with the number of nodes, we have used the benchmark for speeds of various encryption/decryption algorithm [9] provided by an e-security firm. The benchmark is computed on a Windows NT system version 5.0, using a x86 processor. We assume that the processor speed of the router is either similar or higher than x86. This assumption is valid since x86 processor speed ranges

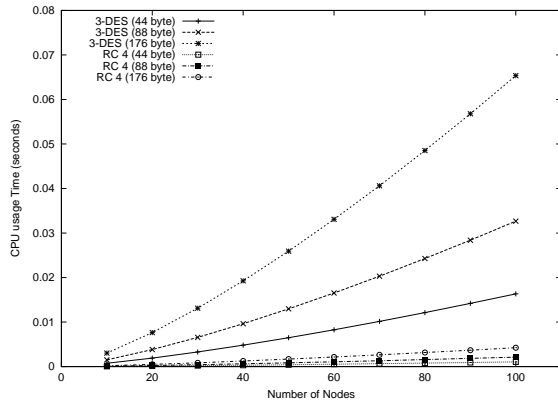


Fig. 6. CPU usage by various algorithms for Decrypting LSAs

from 12 MHz to 100 MHz, and the Cisco 7500 series router uses RSP2, RSP4+ and RSP8 whose internal clock speed is 100 MHz, 200 MHz and 250 MHz, respectively.

Fig. 4 represents plots for the CPU processing time utilized by various hashing algorithm as the number of nodes increases. We assume that for decrypting each LSA, the router is required to generate the subgroup key by using the the key management scheme proposed in [5].

Fig. 5 and Fig. 6 are plots to evaluate the performance of various algorithms for encryption and decryption, respectively. We observe that with increasing LSA size, the complexity due to encryption and decryption increases for all algorithms. We also observe that the best performance (as far as the CPU usage time is concerned) is given by RC 4, while 3-DES is the most time consuming algorithm for all cases; the other algorithms that we studied (DES and RC 2), fall in between. It may be noted that our intention here is to show the performance of algorithms for encryption/decryption rather than suggesting which among these should be used.

According to Moy [10], the router CPU usage is dominated by the length of time it takes to run the shortest path calculation. The complexity of this process grows quadratically with the number of routers in the routing domain. Our scheme has introduced new functionalities for computation by the router's CPU processor due to key generation, decryption and encryption processes. Thus, here we draw a comparison as to which of the two tasks, namely, Dijkstra's calculation and the new overhead (NOH) introduced due to our scheme, is more dominant. Fig. 7 indicates that with increasing LSA data size, the complexity of the NOH increases. This is because the encryption/decryption process depends on the size of the LSA. We also observe that for RC 4 & MD5 combination, the Dijkstra's computation is still dominant for the CPU processing time. While we report on the dominance, it may be noted that these two computational steps are not performed at the same time by a router: the NOH step is performed as soon as an LSU packet is received while there may be a time gap before the next computation of the shortest path tree.

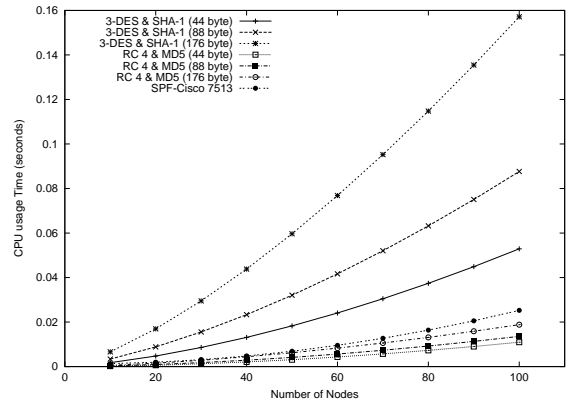


Fig. 7. CPU usage for by Dijkstra's calculation and NOH vs number of nodes

## V. CONCLUSIONS AND FUTURE WORK

In this paper, we present how to address confidentiality in a link-state routing protocol. We present a framework on how to build a hierarchical trust structure.

We note that our introduced encryption/decryption operations will not change the normal operational behavior proposed by OSPFv2 (RFC 2328); only impact is due to computational overhead. Our next step will be to do a prototype implementation based on our proposed approach to explore feasibility of such an approach.

## REFERENCES

- [1] S. Murphy, M. Badger, and B. Wellington, "OSPF with digital signatures," *RFC2154*, June 1997.
- [2] J. Moy, "OSPF version 2," *RFC2328*, April 1998.
- [3] M. Gupta and N. S. Melam, "Authentication/confidentiality for ospfv3," *Internet Draft* (<http://www.ietf.org/internet-drafts/draft-ietf-ospfv3-auth-05.txt>), October 2004.
- [4] D. Huang, Q. Cao, A. Sinha, M. J. Schniederjans, C. Beard, L. Harn, and D. Medhi, "Addressing intra-domain network security issues through secure link-state routing protocol: A new architectural framework," accepted by *Communications of The ACM*, 2006.
- [5] D. Huang and D. Medhi, "A key-chain based keying scheme for many-to-many secure group communication," *ACM Transactions on Information and System Security*, vol. 7, no. 4, pp. 523 – 552, 2004.
- [6] —, "Secure group communication in hierarchical mobile ad-hoc networks," *submitted (available at http://conrel.sice.umkc.edu/HRP/reports/group-adhoc-network.pdf)*, 2004.
- [7] R. Coltun, "The OSPF opaque LSA option," *RFC 2370*, July 1998.
- [8] A. Shaikh and A. Greenberg, "Experience in black-box OSPF measurement," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop (IMW)*, NOV 2001.
- [9] Baltimore, "Keytools v5.1," (<http://download.baltimore.com/keytools/docs/v51/proj-j-docs/html/devguide/projdevguide-C.3.html>).
- [10] J. Moy, "OSPF protocol analysis," *RFC1245*, July 1991.
- [11] —, "OSPF standardization report," *RFC2104*, April 1998.
- [12] A. Basu and J. Riecke, "Stability issues in OSPF routing," in *Proceedings of ACM SIGCOMM*, 2001, pp. 225 – 236.
- [13] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 2003.
- [14] A. V. Aho and D. Lee, "Hierarchical networks and the LSA N squared problem in OSPF routing," in *Proceedings of IEEE GlobeCom*, ser. 27, vol. 1, 2000, pp. 397 – 404.