

Secure and Resilient Routing: A Framework for Resilient Network Architectures

Deep Medhi and Dijiang Huang[‡]

1 Introduction

A common paradigm for a network service architecture is to provide a number of different services or applications in which all of them share resources; the routing framework provides the best path to all services. A simple twist to this basic notion is if we were to architect in a way to virtualize the network so that different services are clustered into different virtual, adaptive partitions to provide different levels of services. In considering this notion, it is important to note that a service offering can span the entire spectrum from complete sharing to physically dedicated partitioning. Furthermore, even without a shared environment, prioritization is also possible, for example, in packet scheduling for different service classes at a router. Typically, to provide a certain quality of service, which might be agreed upon through a service level agreement, a shared environment requires less network bandwidth due to statistical gain as opposed to a completely dedicated, partitioned environment to each different service—this is from the angle of a network service provider. However, in many instances, a middle of the road approach can be a viable alternative in which through virtualization, services can be offered guaranteed prioritized services, and more importantly, such virtualization can be adaptively configured, without requiring complete physical partitioning.

Our middle of the road approach is motivated by a different service paradigm. Consider the simple scenario of two service classes, survivable critical (SC) services and normal services, in which the objective is to provide a certain level of service quality or protection to SC services even under an attack or a network stress situation. This attack or stress can be either of the following forms: (a) injection of excessive dummy traffic to overload the network or network elements, or (b) overtaking of a part of the network (e.g. some routers) so that some network elements become untrusted for use by SC services. An important driver then is that the network has the framework to consider trustworthiness and the routing protocol has the appropriate functionality that can support trustworthiness as well as resilience.

*D. Medhi is with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City (email: dmedhi@umkc.edu).

†D. Huang is with the Department of Computer Science & Engineering, Arizona State University (email: dijiang@asu.edu).

In this work, we present a routing perspective for a resilient network architecture in which different services with different priority can co-exist in a virtualized environment; for resiliency, it is necessary to provide robustness in the routing architecture to protect against attacks as well as network overload. More importantly, we present a general framework for secure and resilient routing that can be conducive to providing secure traffic engineering as well. We, however, take a sort-of backward-direction approach; specifically, instead of starting with what we require or assume in the network architecture, we start with the need for the service requirement for resiliency in a prioritized environment and work backward to identify what are the different components desirable in the network architecture to support this service paradigm. Implicit in our presentation is then the basic understanding that we do not necessarily consider the components identified to be efficient nor do we claim that all components or solutions have been identified; at times, we leave some problems identified as open, research problems.

We conclude this introduction with a general note about terminology. We use the term ‘router’ as a generic routing node; in other words, it is not to be confused with an IP router and its capabilities as is understood in the context of the current Internet. This also means that the router is not necessarily limited to doing a shortest path based computation for route selection. Similarly, we use the term ‘link state routing protocol’ to indicate that information would need to be exchanged for the purpose of route computation that uses a link state framework. Thus, within this framework we also mean that node state information may need to be communicated as well. Hence, we will use link state update (LSU) and extended node/link update (ENLU) interchangeably. For comparison purpose, however, we consider OSPF to identify and contrast what it provides and what we envision a link state routing framework needs, or how OSPF can be extended for the benefit of the resilient network architecture.

Network resiliency has been addressed over the years by many researchers. We note here two works: (1) resilient overlay networks [3] presents an approach for robustness over the current Internet for path diversity; our work addresses the infrastructure itself for the future Internet both from the perspective of robustness and security of routing information, (2) BGP/MPLS [26] architecture provides the ability to separate between routing update, path determination and resource reservation; our architecture can potentially be deployed in a similar environment while we also address security of routing information exchange as well as how the nodes might be differentiated in the presence of an attack.

The rest of the chapter is organized as follows: In Section 2, we present the overall traffic engineering perspective as a motivating discussion towards resilient architecture, while in Section 3 we briefly identify the components of a resilient architecture. In Section 4, we discuss threats and countermeasures for link state routing. We then present the resilient architecture in Section 5 in which we discuss routing protocol extension, virtualization of routing domain, and preliminary analysis. We then conclude with our summary.

2 Traffic Engineering Perspective and its Relation to Network Robustness

In order to provide prioritization to different services under an attack or an overload condition, a key requirement is allocation of resources. For our discussion here, we will use bandwidth as the resource. While theoretically any network can be overprovisioned to the point where such resources are unlimited, from a practical perspective, this is not so. There are two important issues to consider: (1) the cost of overprovisioning can be prohibitive, (2) a sudden surge in traffic due to a number of events (legitimate or otherwise) can overwhelm networks to the point where it becomes non-functional or poorly functional. The first issue can be addressed somewhat by providing some level of overprovisioning, allowable within the capital and operational expenditure budget. However, the second issue brings out the importance of the point that *not* all events can be predicated, and thus, the network requires some form of protection. Combining both of them we can say that protection or prioritization might not be necessary all the time, but the network must have the capability to have that, if and when necessary.

An important question then is how do we protect resources such as bandwidth in a shared environment. While there are a number of approaches in a shared virtualized environment, we take the approach of hiding resources, for example, by *not* announcing all available information in routing updates in a link state routing environment. Before we delve into this aspect on how to do it, we first take a traffic engineering perspective with regard to hiding information in a dynamic routing environment.

To illustrate the hidden information idea in the context of a link state advertisement, we consider a generic link i - j in a network. If $C_{i,j}$ is the total bandwidth of this link, and the currently used bandwidth is $u_{i,j}$, then the free bandwidth is $a_{i,j} = C_{i,j} - u_{i,j}$. Thus, the obvious thing to do is to advertise $a_{i,j}$ in the link state advertisement. This information is flooded through the network and is used by (source) routers when doing routing computation in determining, say, the least-loaded route (e.g., the route with the most available bandwidth, also known as the widest path) to a destination router; this is then used for setting up a new request or a service class.

Suppose that we want to provide a built-in protection mechanism. That is, we want to reserve some bandwidth, say $s_{i,j}$, on link i - j for SC services. The link state advertisement is SC-capable when the available bandwidth advertised is $w_{i,j} = C_{i,j} - u_{i,j} - s_{i,j}$. In effect, we would like normal services to use the quantity $w_{i,j}$ in determining route computation. At the same time, we want to let SC services know that there is really $s_{i,j}$ amount of bandwidth *also* available on link i - j for their use in route computation *and* for traffic flow; this bandwidth is, however, not available to normal services. Note that the SC traffic has access to both the available bandwidth for normal traffic and the bandwidth set aside for SC traffic. To accomplish this, in the link state advertisement, we actually want to advertise $w_{i,j}$ *as well as* $s_{i,j}$; while the normal service sees $w_{i,j}$, the $s_{i,j}$ amount is "hidden" to them. On the other hand, the SC service sees $s_{i,j}$, but based only on a properly specified authorization. Furthermore, if some routers are untrusted, we do not want them to "see" this hidden bandwidth. This mechanism provides a way to accomplish service priority to critical services in computing and selecting

routes that uses the hidden bandwidth. This can be critical during an attack or an overload. Given the assumption that some routers (or users) may be untrusted, we propose to transmit the hidden bandwidth information in an encrypted mode. In this way, only the trusted routers can decipher this information. Note that while we focus here on available bandwidth information to be hidden, other resources that are critical to SC services could also be encrypted and disseminated through a link state update.

The basic paradigm works on a semi-reservation based mode. We use *semi* in two different ways. If in the above mechanism, $s_{i,j}$ is set to zero, then there is no special reservation for SC services. Secondly, updating $w_{i,j}$ with a new value does not mean that a hard reservation is needed; a soft reservation mechanism suffices. The network state maintenance at the routers can be threshold-based with regard to access control. For example, access control may be activated if the link utilization is above a certain value, and/or if notified by the intrusion detection system.

Finally, despite the hidden bandwidth protection mechanism, it is possible that an attacker may still want to consume resources by generating excessive dummy traffic. In the earlier scenario, we assume that the attacker may not use the hidden bandwidth in route computation since it cannot decipher this information. This, however, does not stop the attacker from still trying to inject traffic into the “hidden” parts of the network. To avoid this attack traffic from spreading, we need an access control mechanism at the entry point to check if this traffic should be allowed and a reservation mechanism that protects resources dedicated to SC services. Thus, a dynamic access control mechanism to check such requests is needed. The second possible scenario is that the attacker somehow manages to access the SC traffic class (at some source router), and starts injecting excessive traffic to a certain destination (resulting in, for example, a distributed denial-of-service attack). In this case, through active monitoring, it is desirable to substantially limit the injection of this traffic at different entry points rather than letting it reach the destination – this allows us to contain the attack (closer to the source) rather than let it spread throughout the network for an extended period of time. Either of these scenarios would require some adaptive feedback mechanisms between different entry point routers and possible destinations (choke points) in a network, and a new level of trust generation among routers.

It may be noted that a feature similar to SC service bandwidth protection, called trunk reservation, is currently used in dynamic call routing telephone networks [1]. The trunk reservation (TR) concept [2, 15, 30] is essentially equivalent to the protection we have described here. Note that TR in such settings is typically used to avoid bi-stability in the network [15] rather than the type of protection we are considering. Also, note that TR has been discussed in the context of QoS routing in the Internet [20].

2.1 An Illustrative Example

We consider an example to address some of the issues related to adaptiveness, routing and the SC service protection. Consider a three node network. We will assume a flow-based reservation for the purpose of this illustration; note that this is *not* a requirement in general in semi-

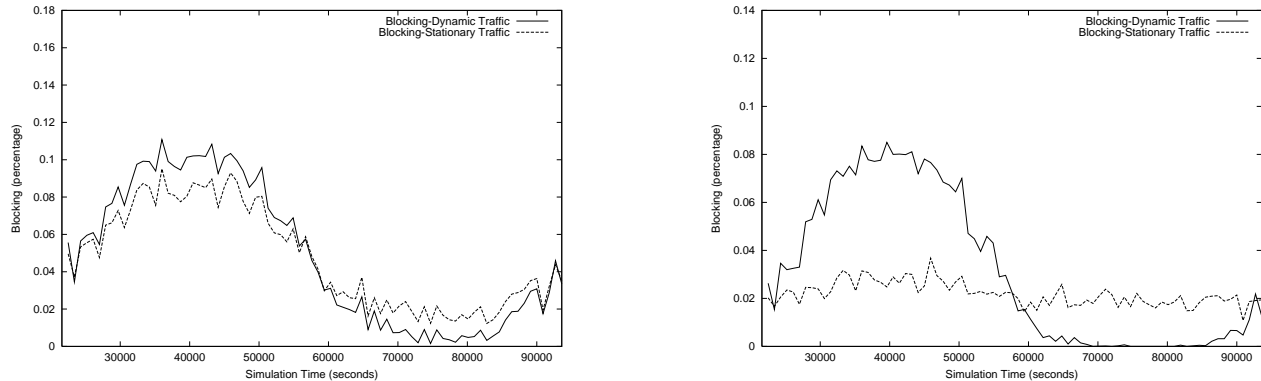


Figure 1: (a): Service Denial (low reservation) (b): Service Denial (extended reservation).

reservation based mode. A new flow request requires a fixed bandwidth rate for the time duration of the flow. Note that in general, a flow does not mean a call or a micro-flow in our framework; it may be a request that is the result of a virtual tunnel that needs to be setup with certain guarantees for a certain time duration, which in turn serves as a bearer of micro-flows. However, for the purpose of this simple illustration a flow may be thought of as a micro-flow or a call.

We assume dynamic flow-based routing (a flow request for a traffic node pair can either use the direct path, or use the alternate two-link path). The traffic between two traffic pairs (i.e., between nodes 1 and 2 denoted by pair $\{1:2\}$, and between nodes 1 and 3 denoted by pair $\{1:3\}$) is assumed to be stationary (and has the traffic volume). On the other hand, the third pair (i.e., between nodes 2 and 3 denoted by $\{2:3\}$) has overloaded traffic caused by an attack – this is represented by a dynamic traffic stream using a time-dependent arrival rate that follows a sinusoidal behavior. Now assume that all of the traffic for both pairs $\{1:2\}$ and $\{1:3\}$ are for SC services, while the dynamic traffic behavior for pair $\{2:3\}$ is for the normal service which contains traffic due to the attack. It may be noted that this is a special case of the framework discussed in Section 2. Specifically, in this case, we have reservations for normal and SC-services with the following condition: $s_{2,3} = 0$ while $s_{1,2} > 0, s_{1,3} > 0$.

We have performed a simulation of this scenario using the MuSDyR simulator [18]. In Fig. 1, we plot the flow blocking probability for the normal service and the SC service over time. As can be seen from Fig. 1(a) (where the SC service protection amount is low), it is clear that as the dynamic traffic for the normal service peaks, it *also* drastically affects the SC services by increasing the denial rate even though the traffic for the SC service did *not* have increase in traffic during this period (since a stationary traffic pattern was used). On the other hand, in Fig. 1(b) (by adjusting the SC service protection parameter values appropriately), we can show that the SC service is receiving essentially the same service guarantee during the entire time while the attack traffic is dynamically increasing during this period, and in fact, the attack traffic is facing heavy blocking as is desirable during this time. On the other hand, when the attack traffic dies away (the right part on each plot), having a low level of protection for SC services can be acceptable.

It is important to note that the protection amount for the SC-service needs to be adjusted although the traffic itself may remain at the same level in a dynamic traffic environment. The second observation is that the reservation parameter values need to be adaptive based on the traffic changes in the network. In other words, a semi-reservation based mode can be the operational mode.

3 Components of a Resilient Network Architecture

Based on the discussion presented so far, we ask ourselves the basic question of what needs to be addressed in a resilient network architecture for such a service paradigm. Our thesis is that (a) the network needs to operate in a semi-reservation-oriented mode, (b) security and authentication need to be a built-in factor, (c) routing can operate and hide information from untrusted elements, and (d) entry points to the network have the ability to do authorization checks and/or control traffic when necessary so that SC services can have priority (under stress).

Consider first one of the basic premises of our approach: the semi-reservation-oriented operation of the network. This means that the network has the ability to reserve resources to be made available to some services (e.g. SC services) *when* they need them. Note that the reservation need not be on a per flow basis [5] as in `int-serv`; a per class-based type reservation as in `diff-serv` [4] suffices when needed. For brevity, we will refer to this network as the *semi-reservation-oriented network (SRON)*. In this network, any reservation which is set does not necessarily remain static over a long duration; in fact, the SRON, in our view, operates in an adaptive reservation mode, and can run without reservation until needed.

Secondly, security is a dynamic built-in factor in the SRON. This allows us to do at least the following: (1) to determine if a router cannot be trusted, (2) to check if a request has the authority to use reserved parts of the network, and (3) to hide certain information in the network through encryption which can only be “seen” by authorized users. This aspect affects the operation of the routers and information exchanged among themselves. Since our interest is to provide priority to SC services, especially under a stress/attack, it is imperative that we periodically check the routers to see if they are trustworthy. Furthermore, we want the capability to hide resources (e.g. bandwidth) in the network that can be used by critical services at the right time using the trusted routers.

Since the network operates in the SRON-mode and for the routers to determine good/acceptable routes, the routers need to exchange information about resource availability. For simplicity of illustration, we will use available bandwidth to be the resource of interest. For the purpose of this illustration, we assume that a link state protocol is used to disseminate this information. The question then is what should be included in this link state advertisement, and more importantly, how do we ensure that the information advertised cannot be deciphered by an entity in the middle of the network. Given this premise, we next consider both insider and outsider threats, and countermeasures in depth in a link state routing framework.

4 Threats and Countermeasures in Link State Routing

4.1 Link State Routing Model and Threat Model

In this section, our discussion focuses on the origination, verification, and transmission of routing data. We start with a brief description of link state routing and security threats specific to it. We discuss the security mechanisms that are known, as of now, to prevent some of these threat actions from taking place. Furthermore, we discuss the vulnerabilities of existing network routing security mechanisms and the improvements that can be done by using stronger authentication and a new method to manage routing data through encryption.

4.1.1 Link State Routing Model

The link state routing model is composed of physical entities (routers and communication links) and logical entities (the link state routing protocol running in the routers). Within a link state routing domain, each router generates the link state information for the link that has the direct connection with the router (the link state information is directional) and floods¹ this information to its neighbors. A receiving router will forward the routing information (unmodified) via flooding again. Therefore, each router will have the same view of the network. When a router joins the network, it needs to synchronize the link state database with its neighbors. The routing information carried by a link state routing protocol is typically the link state of a router's interface. This information is called the *link state advertisement* (LSA). During flooding, multiple LSAs can be encapsulated in a single *link state update* (LSU) routing packet.

The security issues related to the link state routing model can be broadly classified as security for the network device, operational security, and communication security. Security for the network devices concerns the physical access to the routers and communication links. Operational security includes the access control of the operating system of a router, privilege mode of a router, and so on. Communication security is related to the transmission, reception, and processing of routing data (LSAs and LSUs). Note that all data security related issues discussed here are based on routing data but not on user data and we focus on the communication security aspect of the link state routing protocol.

4.1.2 Threats to Link State Routing

In order to categorize the security threats to the routing protocol, we first need to identify the possible threat sources and their actions. We will follow definitions (such as threats, insider/outsider, etc.) provided in RFC2828 [29] for this purpose and use them in the context of network routing and routing protocols.

Threat Sources: The threat sources for link state routing can be through communication links and routers. In the context of a deliberate attack, a threat source is defined as a motivated,

¹Flooding provides robust data transmission, in which a router forwards the link state routing information packet to every interface except the one it receives the link state routing information packet.

capable adversary. Attacks can come from outside as well as from inside. As such, it is equally important to provide adequate safeguards for both internal and external threat sources. Here, internal threat sources are called *insiders* and external threat sources are called *outsiders*. In other words, the legitimate participants in network routing are called insiders. On the contrary, the illegitimate participants in the network routing are called outsiders. The outsiders can reside anywhere in a network and have the ability to observe routed traffic on a link or send attack packets to routers. Note that an outsider can masquerade to generate routing information as an insider. However, an outsider has no valid identifier and is not authorized to perform routing functions. An insider can also masquerade as another authorized router and generate forged routing information. It has a valid identifier, but it is not authorized to impersonate other routers or forge other routers' routing information (by "authorized" we mean the permission for overall routing operational functionalities). In this sense, we can say an outsider is unauthorized and an insider is authorized.

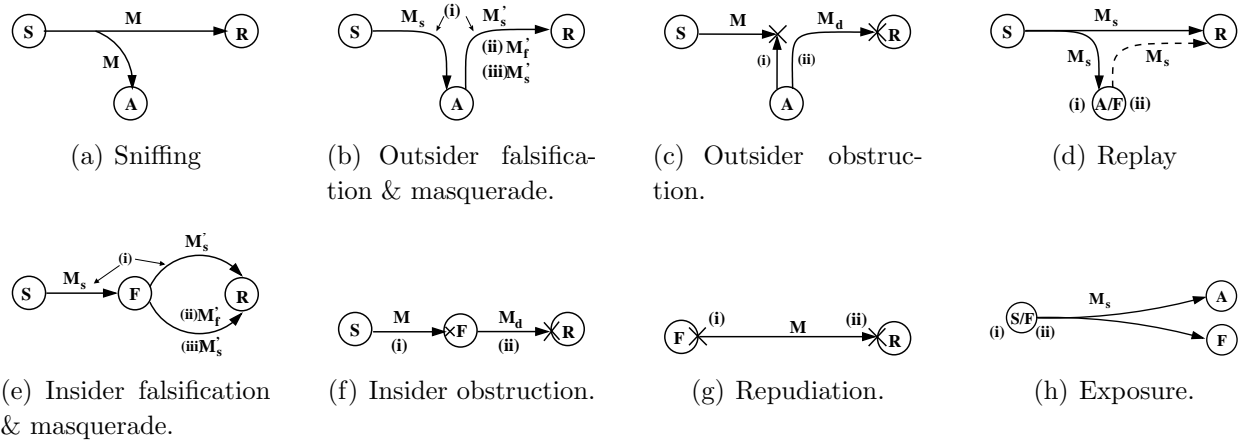


Figure 2: Outsider attacks (a) to (c) and (d.1), Insider attacks (d.2) and (e) to (h), (S:sender R:receiver (or victim) A: outsider (attacker) F: insider (attacker) M : any routing message M_s : routing message from sender M'_s : forged routing message of sender M_f : malicious routing message generated by subverted routers M_d : dummy routing traffic that cause overload $--\rightarrow$: delayed transmission).

Threat Actions: Threat actions are also called attacks. Here, all our discussions focus on the origination, verification, and transmission of routing data. The attacks can be active or passive. Defined in RFC 2828, “an *active attack* attempts to alter system resources or affects their operation; a *passive attack* attempts to learn or makes use of information from the system but does not affect system resources”. Attacks can also be classified based on threat sources – insiders and outsiders. This classification helps to categorize corresponding preventive cryptographic countermeasures which will be discussed in Section 4.2. Note that we focus on the attack model that an attacker uses to compromise other routers' resources. Thus, we exclude the discussion of insider attackers that overclaim/underclaim/misclaim the network resources that are under its control. For example, a subverted router claims that the bandwidth attached

to one of its interfaces is w , but in fact, the actual bandwidth is r , where $w \neq r$.

Outsider Attacks:

- (a) Sniffing (passive): Monitoring and recording routing data transmitted on the communication links among routers; see Fig. 2(a).
- (b) Falsification and masquerading (active): This attack can be of three kinds:
 - (b.1) Substitution: altering or replacing valid routing information with false routing information; see (i) in Fig. 2(b),
 - (b.2) Insertion: introducing false routing data that serves to deceive an authorized router; see (ii) in Fig. 2(b),
 - (b.3) Masquerading: impersonating an authorized link/router; see (iii) in Fig. 2(b). Masquerading is usually executed concurrently with substitution and/or insertion.
- (c) Obstruction (active): This attack can be of two types:
 - (c.1) Interference: an attacker can block the transmission link, by cutting off the transmission link or by introducing noise into the transmission link to prevent the victims from receiving the routing information correctly; see (i) in Fig. 2(c),
 - (c.2) Overload: an attacker can place excess dummy routing traffic that can saturate the victim's input buffer or exhaust victim's CPU capacity; see (ii) in Fig. 2(c).
- (d.1) Replay (active): a valid routing data transmission is maliciously or fraudulently repeated by an outsider; see (i) in Fig. 2(d).

Insider Attacks:

- (d.2) Replay (active): A valid routing data transmission is maliciously or fraudulently repeated by insiders, (see (ii) in Fig. 2(d)).
- (e) Falsification and masquerading (active): This is the same as specified in outsiders' threat actions (b) (see Fig. 2(e)). Comparing with an outsider's falsification and masquerading, the insider attack can be more effective. Since the insider is a legitimate participant, he/she might know the shared key or serve as an intermediate forwarder, which makes the attack easier but more difficult to detect.
- (f) Obstruction (active):
 - (f.1) Stop forwarding: the subverted router does not forward received routing packets; see (i) in Fig. 2(f),
 - (f.2) Overload: excessive routing information processing burden is placed on the router in order to saturate the victim's input buffer or exhaust victim's CPU capacity; see (ii) in Fig. 2(f). This is the same as the outsider's overload attack (c.2).
- (g) Repudiation (active):
 - (g.1) False denial of origin: a subverted router denies the operations that it has made on

Table 1: Security Mechanisms.

Methods		Label	Description	Protection
Authentication (A)	Packet Level	$PLAP2P^\dagger$	Packet level, point to point authentication	Data & Origin Integrity
		$PLAE2E^\ddagger$	Packet level, end to end authentication	
	Information Level	$ILAP2P^\mathcal{L}$	Information level, point to point authentication	
		$ILAE2E^\ddagger$	Information level, end to end authentication	
Confidentiality (C)		PLC^\S	Confidentiality for the whole packet	Information
		ILC^\P	Confidentiality for the information within the packet	Availability

\dagger : OSPFv2 RFC2328 and OSPFv3. \ddagger : OSPF extension RFC2154.

\mathcal{L} : Proposed in [11]. \P : Not in current literature.

the transmitted routing information; see (i) in Fig. 2(g),

(g.2) False denial of receipt: a subverted router denies receiving the routing data; see (ii) in Fig. 2(g). Although an outsider can repudiate what it has done, it is more critical for insider attacks. A subverted router can cause a more serious problem when it is authorized to perform routing functions. Quickly identifying the subverted routers/links will help to reduce the recovery time imposed by the attacks.

(h) Exposure (active):

(h.1) Non-deliberate exposure: a router unintentionally releases sensitive routing data to attackers (both insiders and outsiders); see (i) in Fig. 2(h),

(h.2) Deliberate exposure: a subverted router intentionally releases sensitive routing data to attackers (both insiders and outsiders); see (ii) in Fig. 2(h).

We note that in our classification all attacks originating from outsiders occur on the routing transmission links, e.g. Fig 2(a) to 2(d). Among these, attacks (c.1) and (d.2) need to get access to the transmission link first and then attackers can launch attacks. Attacks originating from insiders are generated by the subverted routers, e.g., Fig 2(d) to 2(h). When an outsider successfully takes over an authorized router, it becomes an insider (or subverted router). In other words, the outsider usurps the rights of a legitimate router.

4.2 Preventive Cryptographic Countermeasures Against Attacks

The challenges posed due to the enormity and diversity of threats has led to various research activities in recent years that address techniques to safeguard a network. On the other hand, the current standards for network routing protocols have not incorporated all of the techniques required to make them as foolproof as possible. A set of unplugged security holes remain that an adversary can use to paralyze the network. In this section, we analyze the possible preventive cryptographic countermeasures first and then describe how they can completely or partially prevent attacks from taking place.

4.2.1 Preventive Cryptographic Countermeasures

Computer security rests on confidentiality, integrity, and availability. Table 1 enlists two preventive cryptographic countermeasures that are described in the literature, including those

that have found a place in protocol standards. The two main preventive cryptographic countermeasures that have been suggested for routing protocols are confidentiality and integrity. Confidentiality ensures that no unauthorized entities can decipher the routing information on its way to the destination. Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change.

Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called authentication). The interpretations of integrity and authentication vary, as do the contexts in which they arise. In a link state routing context/setting, authentication is generally considered as both data integrity and origin integrity. For example, a keyed-hashing message authentication (HMAC) code can be used as the cryptographic authentication for OSPF (see RFC 2328 [22]). It provides data integrity and only authorized user (possesses a shared key) can generate and verify the HMAC. Similarly, digital signatures for OSPF (See RFC 2154 [23]) also provides both data integrity and origin integrity. In the following, we consider authentication as providing both data integrity and origin integrity.

Availability refers to the ability to use the information or resource desired. The aspect of availability that is relevant to security is that someone may deliberately arrange to deny access to data or to a service by making it unavailable, such as a denial-of-service (DOS) attack. Preventive cryptographic countermeasures, by themselves, can do little to prevent DOS attacks. Most of current solutions to DOS attacks are reactive solutions, i.e., solutions that depend on intrusion detection systems (IDS), which is beyond the scope of this research. We will show later on how to create multiple trusted routing domains to mitigate the consequence of DOS attack.

Here, we focus on the following two preventive cryptographic countermeasures: confidentiality and authentication. These two countermeasures can provide protection at either the packet level (PL) or at the information level (IL). If we assume a routing packet to be a bus filled with a group of passengers, PL and IL represent the cryptographic countermeasures being provided for the bus and each individual passenger, respectively. Besides authentication and confidentiality, there are two other important concepts we need to introduce; they are point-to-point (P2P) and end-to-end (E2E). In terms of authentication, P2P means that the generation and verification of an authentication code are performed by every forwarding router. E2E means that the generation of an authentication code is performed only at the source; all the forwarding routers and termination routers are part of the end system, and they only perform verification. In Table 1, we provide a summary of the two main preventive cryptographic countermeasures for link state routing protocols.

In the link state routing protocol, pieces of routing information, *link state advertisements* (LSAs), are encapsulated in a *link state update* (LSU) packet. Most of the current implementations fall into the category of PLA_{P2P} . If PLA_{P2P} is provided for the entire LSU packet, then PLA_{P2P} can guard against the “man-in-the-middle” attack [29]. But, in link state routing, flooding is used for distributing LSAs within a link state routing domain; PLA_{P2P} cannot prevent any intermediate subverted router from modifying forwarded LSAs, or a router from originating forged LSAs. E2E is more desirable to provide stronger protection for LSAs. But another difficulty of link state routing is that multiple LSAs are encapsulated within a single

Table 2: Threats and corresponding cryptographic preventive countermeasures.

Threats		Attack Types		Preventive Countermeasures	Remarks	
Threat actions (Attacks)	Label	I/O	P/A			
Wiretapping		(a)	O	Passive	PLC or ILC	★★
Outsider falsification & masquerade	Substitution	(b.1)	O	Active	PLA_{P2P}	✓
	Insertion	(b.2)	O	Active	PLA_{P2P}	✓
	Masquerading	(b.3)	O	Active	PLA_{P2P}	✓
Outsider obstruction	Interference	(c.1)	O	Active	PLC or ILC	★★★
	Overload	(c.2)	O	Active	PLA_{P2P}	✓
Replay	Outsider replay	(d.1)	O	Active	New keys	★★
	Insider replay	(d.2)	I	Active	New keys & ILC or ILA_{E2E}	★★★
Insider falsification & masquerade	Substitution	(e.1)	I	Active	ILA_{E2E}	★
	Insertion	(e.2)	I	Active	n/a	✗
	Masquerading	(e.3)	I	Active	ILA_{E2E}	★
Insider obstruction	Stop forwarding	(f.1)	I	Active	n/a	✗
	Overload	(f.2)	I	Active	ILC	★★★
Repudiation	False denial of origin	(g.1)	I	Active	ILA_{E2E} or ILA_{P2P}	★
	False denial of receipt	(g.2)	I	Active	PLA_{P2P}	★★★
Exposure	Insider un-deliberate exposure	(h.1)	I	Active	PLC^\dagger or ILC^\ddagger	★★
	Insider deliberate exposure	(h.2)	I	Active	ILC	★★★

I/O: Insider/Outsider (attacks). P/A:Passive/Active (attacks). †: Guard against outsider attacks.

‡: Guard against insider attacks. ✓: Solvable via well known solutions and widely deployed.

★: Solvable via well known solutions but less deployed. ★★: Solvable via solutions proposed here.

★★★: Partially solvable via our proposed solution.

✗: Unsolvable via authentication and confidentiality.

LSU packet and the content of each LSU that originated from different routers may be *different*. This prevents PLA_{E2E} from being implemented efficiently. Hence, ILA_{E2E} and ILA_{P2P} are required to provide information level protection. OSPF with digital signatures [23] is an example of ILA_{E2E} , while the double authentication scheme [11] is an example of ILA_{P2P} .

For confidentiality, too, we differentiate between packet level and information level, which is shown in Table 1. OSPF running over IPsec [7] is an example of providing PLC , which provides confidentiality for the IP payload. Providing confidentiality for each LSA individually is represented by ILC .

4.2.2 Using Preventive Cryptographic Countermeasures to Guard Against Attacks

Next, we analyze how to use cryptographic countermeasures presented in Table 1 to guard against threat actions illustrated in Fig. 2. Table 2 presents the mapping of threats and corresponding countermeasures. Threat actions that are marked with ✓ are solvable via well known solutions and are outsider attacks. Attacks presented in (b) can be easily guarded against by using PLA_{P2P} . The dummy routing traffic due to attack (c.2) can be filtered out using PLA_{P2P} . Although, a cryptographic-based operation can aggravate the CPU computation burden, the overload attack is usually limited within a small range of where it happens. This is because the

excess routing traffic cannot get through a router. This may be useful in preventing *distributed denial of service* (DDoS).

It may be noted that preventive countermeasures, such as authentication and confidentiality, cannot prevent attacks that are marked with ✕ (see (e.2) and (f.1)). These attacks need other security mechanisms such as admission/access control, intrusion detection, and so on.

Our discussion here will focus only on the countermeasures marked by ★, ★★, or ★★★. The countermeasures marked by ★ are specified in the current literature, such as OSPF with digital signatures [23]. We note that end-to-end authentication is considered as a strong preventive cryptographic countermeasure. A widely accepted proposal uses the public key scheme to sign each LSA. The reason we separate it from *PLA*, marked by ✓, is because of the deployment difficulty of digital signatures, which comes with a high computation overhead compared with the traditional authentication schemes, e.g., the keyed hash function (HMAC) [14]. Countermeasures marked by ★★ are barely addressed in the current literature, while countermeasures marked by ★★★ have not been addressed so far.

Guarding Against Attacks on Communication Links

As shown in Table 2, attacks from (a) to (d.1) are injected on the communication link. Here, we investigate the possible use of preventive cryptographic countermeasures when attacks (a), (c.1) and (d.1) occur (marked with ★★ and ★★★).

- *Outsider Wiretapping Attack (a)*: *PLC* or *ILC* can be used to prevent outsiders from sniffing packets containing routing information. This is a straight forward method to prevent passive attacks. When *PLC* is provided for the entire IP payload, the outsider would not know general information, such as link state type, advertising router, sequence number, which are contained within the routing packet header. This information can help an attacker to derive network topology and traffic patterns. *ILC* cannot prevent an attacker from knowing the information within the routing packet header, but it can prevent subverted routers from decrypting the routing information when they use different encryption/decryption keys. The combination of *PLC* and *ILC* provides strong security features to guard against ineligible entities.
- *Outsider Interference Attack (c.1)*: We assume that there is an admission control mechanism to prevent outsiders from using network tools to derive the network topology. This can also be done by simply disabling those network services. Then an attacker might arbitrarily wiretap any possible communication links to intercept the routing information. Plain text routing information can help attackers to derive the network topology and traffic patterns. Due to flooding of the routing information used by the link state routing protocol, tapping one link can help attackers to intercept all flooded LSAs within its routing domain. The intercepted routing information can be valuable for attackers to decide the location of an attack target, such as the weakest communication links or the partition routers. We note that providing confidentiality cannot prevent an attacker from doing active attacks. However, without the network topology and traffic pattern

information, it is hard for attackers to deploy attacks successfully. Note that most of the active attacks presented here require the network topology or traffic pattern information.

- *Outsider Replay Attack (d.1)*: Although link state routing protocols typically use a non-decreasing sequence number to prevent replay attack, the replay attack can still take place when the sequence number is rolled over or a router reboots.

Guarding Against Attacks on Routers

We next discuss possible use of preventive cryptographic countermeasures when attacks take place on a router. Insider attacks (d.2), (e.1), (f.2), (g), and (h) are some examples of attacks on routers (marked by ★, ★★ and ★★★).

- *Insider Replay Attack (d.2)*: The analysis of attack (d.2) is similar to the outsider replay attack (d.1). The difference is that information level confidentiality is required. We illustrate the reason through a simple example; the LSA contains the bandwidth information of a particular link l which is c_l . When all routers share a common key to sign/encrypt LSAs, any subverted insider can replay an LSA. Preventing this form of attack in the current framework is difficult. Therefore, a new framework should be such that one can allocate a different set of keys to routers and sign/encrypt only a sub-portion of bandwidth of link l , say c'_l . The insider attacker can only replay the old routing information c'_l , which only affects parts of the network resources. This would mean that not all routers within the link state routing domain share the same network resource information. If we consider a link state routing domain as a trust domain (TD), through *ILC* we can differentiate it as multiple sub-trust domains. Hence, if one of the sub-routing systems is compromised, it does not affect others or can only cause minimal damage to other sub-trust domains. Since routers within a TD share the crypto key, the replay attack can still occur and updating the crypto key will not be helpful to prevent the subverted routers from replaying the old routing packets. Thus, building multiple trust domains through *ILC* can only limit the effect of attack (d.2).
- *Insider Substitution Attack (e.1) and Insider Masquerading Attack (e.3)*: In the case of attacks (e.1) and (e.3), *ILA_{E2E}* can prevent subverted routers from substituting the routing information and masquerading as other routers when data origin authentication is provided for each LSA and is guaranteed end-to-end.
- *Insider Overload Attack (f.2)*: This scenario occurs when there is an excessive routing information burden on routers, overloading the routers' input buffer or CPU. It is different from outsider overload attack (c.2) which only aims to overload the neighboring routers' input buffers or CPU. Insider overload can cause more damage, because the attacked routers will forward the excessive routing information to the next hop due to flooding. Although some link state routing protocols set a minimal arrival interval to limit LSA updating, a subverted router can circumvent this protection via inventing new LSA instances constantly. Thus, the attacking traffic can be spread throughout the network due to flooding.

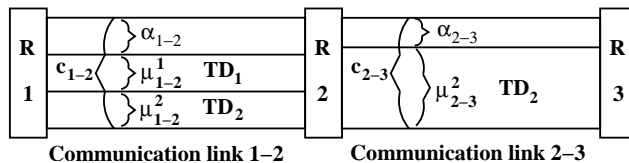


Figure 3: Communication link under attack.

In order to limit the routing data traffic overload attack, we again need to divide the link state routing domain into multiple smaller link state routing (sub-)domains. We assume here that the network has the link bandwidth management capability. An illustrative example is shown in Fig. 3 which represents a network segment composed of three routers and two links. The capacity of a communication link 1-2 is c_{1-2} . TD_1 and TD_2 are configured to run through link 1-2. The bandwidth allocated for these two TDs are μ_{1-2}^1 and μ_{1-2}^2 (the superscript is the identifier of a TD and the subscript is the identifier of a link). The available bandwidth of link 1-2 is given by α_{1-2} . Then, we have $c_{1-2} = \mu_{1-2}^1 + \mu_{1-2}^2 + \alpha_{1-2}$. Similarly, the capacity allocation of link 2-3 is $c_{2-3} = \mu_{2-3}^1 + \mu_{2-3}^2 + \alpha_{2-3}$. Note that the reserved bandwidth is guaranteed through both bandwidth management and confidentiality provided for each TD. A particular encryption/decryption session key is used to provide confidentiality for a TD. In our example, both routers R1 and R2 can decrypt LSAs for TD_1 and TD_2 , and R3 can only see the available bandwidth allocated for TD_2 . R3 does not possess the session key used by TD_2 , and it cannot forge routing information to announce allocated bandwidth on link 2-3 for TD_1 . If the subverted router R3 overloads R2, then R2 would not forward the excessive routing traffic which exceeds μ_{1-2}^2 . The traffic control is done through bandwidth management and the TD identification is done through providing confidentiality (*ILC*) to hide the network resources.

- *Insider False Denial of Origin Attack (g.1)*: In the case of attack (g.1), $ILAE2E$ can prevent insiders from denying the original sources which sent the false routing information. The authentication code should provide evidence that the sender cannot deny, e.g., a digital signature.
- *Insider False Denial of Receipt Attack (g.2)*: The acknowledgement mechanism of a link state routing protocol is neighbor-to-neighbor based. Multiple LSAs can be acknowledged by a single link state acknowledgement packet. The acknowledgement packet can use a shared key between the communication peers to authenticate the received packet. However, by using a shared key based neighbor-to-neighbor authentication mechanism, there is no way to explicitly determine who generates the packets. Use of $ILAE2E$ and $ILAP2P$ for acknowledgement of every LSA is impractical and unnecessary. Moreover, the receiver can stop sending back acknowledgements. Thus, using $PLAP2P$ for acknowledgement is optional and can only benefit in preventing the “man-in-the-middle” attack.
- *Insider Undeliberate Exposure Attack (h.1)*: An insider may unintentionally expose routing information to outsiders or other insiders that are not necessarily receiving the routing

information (for example, if the communication is via a wireless link). The analysis of this scenario is the same as scenario (a). *PLC* ensures no outsider can reveal the content. Within the multiple TDs framework, *ILC* ensures only eligible TD members can reveal the content within their TD.

- *Insider Deliberate Exposure Attack (h.2)*: An insider can deliberately expose the routing information to anyone. But, with the routing information protected by *ILC*, a subverted/compromised router cannot expose the routing information of other TDs to which it does not belong.

5 Resilient Architecture: Virtualization and Routing

Having discussed so far our original premise along with the routing protocol framework and the attack models in detail, we are now ready to discuss the resilient network architecture and the role of routing which has the benefit of enabling virtualized environment.

5.1 An Enabling Framework for Adaptive and Secure Virtualized Networking

Network virtualization to provide prioritized critical/emergency services is a critical need for cybertrust in next generation networks. In this section, we present a new secure, extended node/link update (ENLU) framework by extending a link state routing framework through a secure group communication approach for enabling network virtualization. This scheme allows dissemination of ENLU messages to be encoded in such a way that only nodes with the proper key can take advantage of the encoded information for prioritized services. We invoke a many-to-many group communication keying scheme to virtualize network resources to support multiple service domains; the scheme has been described in detail [9] and is summarized in Appendix A.

5.1.1 Cryptographic Approaches for Network Resource Prioritization

In general, confidentiality ensures that no unauthorized entities can decipher the routing information on its way to a destination; integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called authentication).

As discussed earlier, our approach considers two preventive cryptographic countermeasures: confidentiality and authentication. These two countermeasures can provide protection at either the packet level (PL) or the information level (IL), shown in Fig. 4(a). If we assume a routing packet to be a bus filled with a group of passengers, PL and IL represent the cryptographic countermeasures being provided for the bus and each individual passenger, respectively.

To date, there has been no information level confidentiality (ILC) schemes proposed for routing protocols that can be used for network virtualization. Our approach is to use ILC

and information level authentication (ILA) as the foundation to build a new secure routing framework. To deploy ILC, routing information (i.e., metric) is categorized by multiple groups. By carefully assigning group keys to nodes, we can partition network resource into multiple routing domains. For example, consider a node with several outgoing links; it can encrypt routing metrics (RMs) for some links using one key and encrypt RMs for other links using another key. Thus, only nodes that have the correct key can decrypt the routing information. This strategy can also be applied to a single link, i.e., a node can partition the bandwidth of a link into multiple portions and create/encrypt an RM for each portion. This approach has several benefits:

- It prevents outsiders’ sniffing attacks (we assume that the crypto key length is long enough to prevent brute force attack within a maintenance cycle, i.e., periodic updating window of the crypto keys).
- It mitigates outsiders’ traffic-analysis attacks: Since extended node/link attributes are encrypted and a node may or may not possess the decrypting key, nodes can maintain different network topology information and shortest path tree or other provisioned paths. Thus, the data flow may not follow the same shortest path, which can prevent attackers from deriving the correct network topology or traffic allocation pattern.
- An insider has limited information of the network, which can mitigate routing analysis and deliberate exposure attacks.

To implement ILC and ILA, an efficient secure group key management scheme, which supports many-to-many secure group communication, is needed. Many-to-many secure group communication requires that each group member (node in our case) with a group population of size n can communicate with any subgroup of members securely and on a real-time basis without requiring a new set of key for each subgroup communications. In general, this means a group member would need to possess $2^{n-1} - 1$ keys; however, our secure many-to-many group communication keying scheme [10], summarized in Appendix A, has been designed for the purpose of virtualization and has much less complexity. Briefly, our many-to-many secure group communication keying scheme has the following advantages:

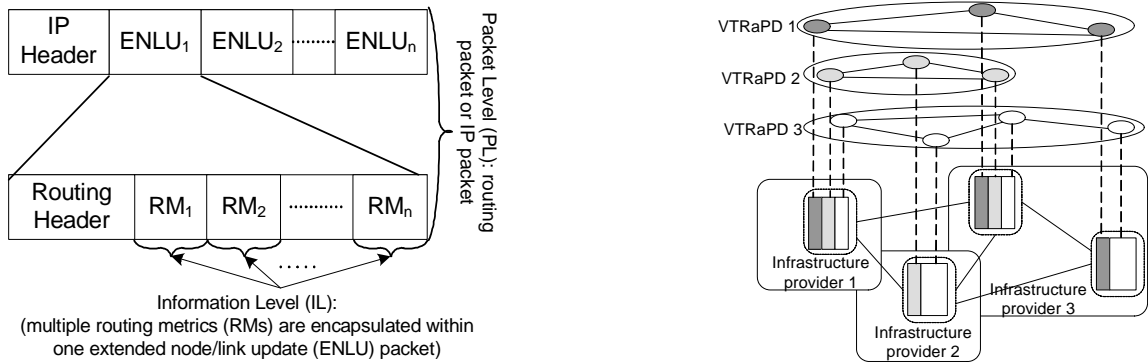


Figure 4: (a) Granularity of attributes at the packet level and information level, (b) VTRaPD: Conceptual framework.

1. During the communication phase, group members can self-derive desired subgroup keys,
2. No group/subgroup setup delay, although there is some processing overhead incurred due to the key agreement protocol,
3. Less communication overhead incurred compared to other methods,
4. It is suitable where subgroup formation is frequent,
5. A node cannot partner with another node to move to a different (unauthorized) subgroup.

The above advantages are ideal for our the encrypted, extended node/link update (ENLU) framework since it allows us to disseminate ENLU messages in a way that is meant only for a subgroup of nodes. Recall that we initially considered two services, normal services and survivable critical (SC) services, with the important requirement that survivable critical services encompass normal services as well. This can be accomplished by defining two subgroups for extended node/link state dissemination using the many-to-many secure group communication scheme we have developed. Note that although there are only two subgroups in this case, which are mapped to two categories of services, the formations of subgroups can be changed frequently with respect to the use of different subgroup keys.

Note that our keying scheme addresses the problem of undesirable node partnering with other nodes to read all attributes of an advertised LSA, thus preventing a node from moving from normal service state to a prioritized services without having the credentials (i.e., the set of secrets to derive desired group/subgroup keys).

Our keying scheme has an additional advantage since it allows dynamic subgroup formation. This means that if a network wants to define multiple prioritized service levels dynamically, our approach allows it with the added advantage that a node can be in different prioritized groups and yet, it cannot become an undesirable node (that is to move to a higher prioritized service class).

It may be noted that if the number of nodes in the network is likely to grow, then our scheme can be deployed with over-provisioned keys. For example, if the overall group size is currently around 50, and it is expected to grow to about 100, then the initial deployment can be done assuming the total group members to be, say, 101 since our scheme requires the overall population size to be odd numbered. This would mean that there would be some fictitious group members, to start with. This approach then avoids re-distribution of keys frequently (at the expense of over-provisioning) to nodes in the entire network. Our current approach has the following limitations: i) the storage complexity of the keying scheme is $O(n^2)$, the overall group size is restricted to a few hundred nodes, ii) a centralized key server is required to do initial key pre-distribution. An important research goal is to overcome these limitations.

5.1.2 Virtual Trust Routing and Provisioning Domain (VTRaPD)

Using our many-to-many secure group keying scheme and information level encryption and authentication approach, the entire routing domain can be divided into multiple routing/provisioning

sub-domains. We refer to such a sub-domain as a virtual trust routing and provisioning domain (VTRaPD) (Fig. 4(b)). The framework may not need/imply the division of the administrative domain into VTRaPDs.

Every node that belongs to a particular VTRaPD will have complete routing information of its own VTRaPD, but not others. We use the cryptographic techniques ILA and ILC to build the VTRaPD framework. Each node can be provided by a different infrastructure provider; however, each node would need to support our framework that includes secure many-to-many communication as well as the capability of link bandwidth control and virtualization. For example, the bandwidth of a communication link of each node would be divided by using different encryption/decryption/authentication keys. While bandwidth partitioning is not directly available in most of today’s routers, this can be accomplished through the concept of multiple virtual links due to availability of the virtual link concept in the current generation of routers. Thus, a subset of network resources, which is composed of multiple network links using the same encryption/decryption/authentication key, will build a VTRaPD.

We now briefly discuss how the overall system framework is affected (see Fig 5). Typically, from a systems perspective, traffic management and network resource management components are necessary for monitoring and managing a network. For resilient environment, there are three additional components involved: intrusion detection system, key management, and VTRaPD. Note that intrusion detection is outside the scope of the present work; however, its role is important in the overall system framework.

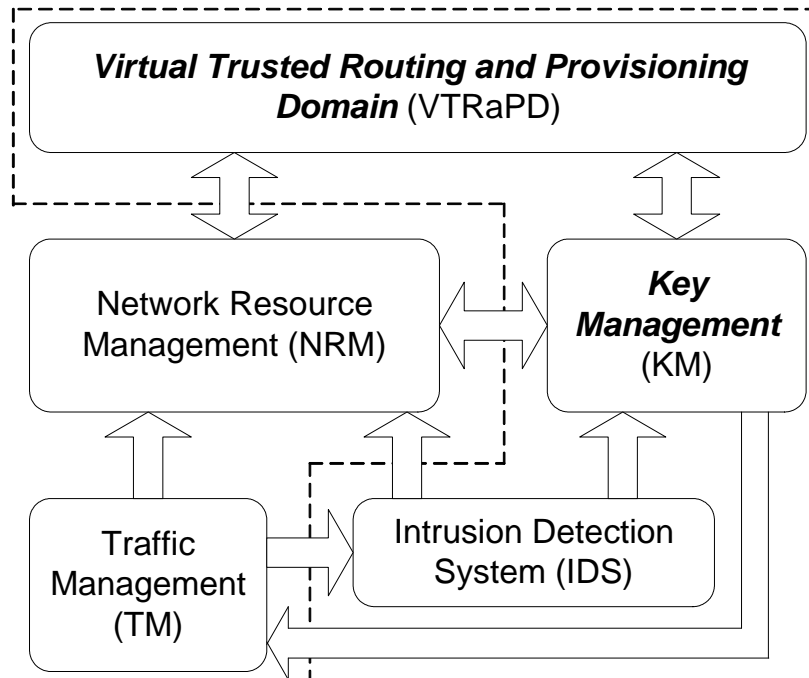


Figure 5: System Framework.

5.2 Routing Protocol Extension: OSPF-E

In this section we present an extension to OSPF, to be referred to as OSPF-E, for use in a VTRaPD environment. A basic notion is that our approach benefits from traffic engineering extension of OSPF, known as OSPF-TE. Here, we go beyond that to address the encryption of information. To achieve this, we also use the opaque LSA option (RFC2370 [6]). The opaque LSA consists of a standard LSA header followed by application specific information and it provides a generalized mechanism to allow for the future extensibility of OSPF. We include the details of our proposed LSA packet format. We then introduce a key numbering scheme in order to identify the trust level of the routers and the key that has been used to encrypt the routing information. We also discuss processing overhead of the LSU packet which is critical in an operational environment.

5.2.1 OSPF Opaque LSA: Extension

Opaque LSA [6] provide three LSA types - type 9, type 10, and type 11. These three LSA types provide the advertisement – within a network, area and autonomous system respectively. We define the opaque LSA format to provide confidentiality for these three types of advertisements. In addition to these three types of opaque LSAs, we can define other types of opaque LSAs for particular use, such as a key distribution LSA, a routing control LSA.

In our scheme, authentication is based on the packet level of LSA, since the LSA header is not encrypted. We do not intend to provide a way to prevent insider attack; rather, we show how OSPF-E works by introducing confidentiality. Murphy et. al. [23] uses digital signatures for each LSA to prevent impersonation attacks, which can be added to our scheme when needed. Here, we assume the network has the capability to detect insider attacks.

We now summarize the changes we propose in opaque LSA. The modified opaque LSA header is shown in Table II.

Table 3: 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			
...			

- 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 S bit in the same position to indicate the confidentiality provided.

2. LSA type (8 bits): Three types of Opaque LSAs exist (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. The 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, etc.
5. Key ID (8 bits): This field specifies the type of cryptographic scheme used to encrypt a propagated LSA. Some of the schemes are shared key, public key and so on.
6. LSA sub-header: The LSA sub-header just follows the opaque LSA’s header and identifies what encryption/decryption key is used.

Table 4: 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		
	...		

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

1. Format (16 bits): This field specifies what type of key is used: it can be Globe/level/sub-group key or individual key etc.
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 each level’s information from the top down to the level in which the key is located; here, 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 by the individual or globe/level/sub group key. The length is variable and depends on the field of "SKey len".

5.2.2 Key Numbering Scheme

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

A OSPF-E key numbering example is shown in Fig. 6 in which both the Trusted Group (TG) and the keys are labeled with a "dot notation". The leading number stands for the top level of the group. The succeeding numbers following the "." represents sub-group, sub-sub-group members and so on. So, it can be conveniently represented as: *root.group.sub-group.sub-sub-group...*

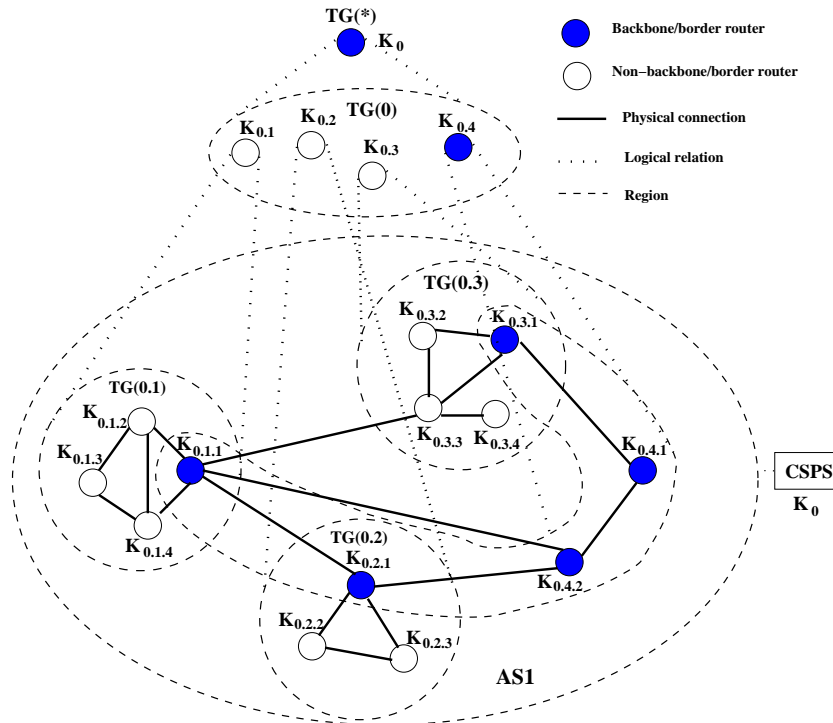


Figure 6: Key allocations of OSPF-E hierarchical trust structure.

When a router receives an encrypted routing information packet with key label as *a.b.c.d*, it compares its key label with the one received to find out the common parts. In the example, in group TG(0.1), there are four routers with key labels 0.1.1, 0.1.2, 0.1.3, and 0.1.4. The router labeled with $K_{0.1}$ is trusted group leader (TGL) of TG(0.1). Using its predistributed key set, the TGL can derive all the keys that are distributed to its group members (using the group key management scheme discussed in Appendix A). We use suffix 0 in a key label to represent a group key. Thus the group key for this TG is given by 0.1.0. If 0.1.2 and 0.1.3 want to set up sub-group communication, then they can use sub-group key with the label 0.1.2-3, where "-" denotes communication within a sub-group.

5.2.3 OSPF-E and LSA Packet Processing

In order to incorporate the encryption/decryption process into the LSU packet processing, the flow chart presented [27] for OSPF needs to be modified. Fig. 7 represents the new flow chart for the OSPF-E processes initiated, once the LSU packet is received by the router. It can be observed that on receiving an LSU packet, the router processes the LSAs contained in that packet one-by-one. It looks up the LSA packet header to find 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 LSU packet. If the key belongs to the same level or lower, the router will either use its own key or generate the key using the group keying scheme presented in Appendix A and decrypt the LSA. Once the LSA is decrypted, it is checked for duplicates. OSPF-E updates its link state database for every new LSA it receives. The flooding of LSA is similar to OSPF. It also needs to schedule the best route calculation module.

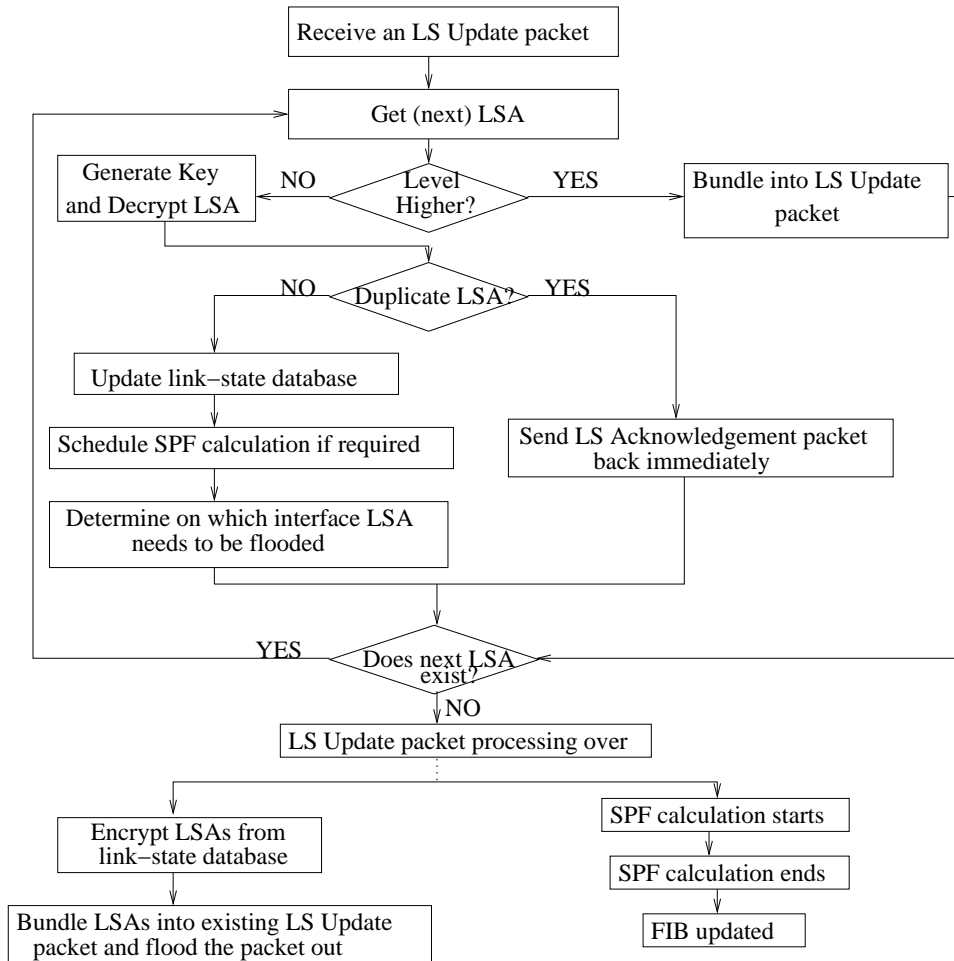


Figure 7: Flow Chart depicting the OSPF-E processes initiated on receipt of LSU packets.

Before the router creates the LSU 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 LSU packet and flooded through the interface.

5.3 Network Analysis: Preliminary Results

An important question is: can we quantify the benefit in a network that protected services do get allocate prioritized provisioning in a network virtualization framework based on secure encryption? To be able to quantify such benefit, we have recently started the development of a virtual network *analysis* simulator (VNAS), which is extended from MuSDyR [18].

In VNAS, we have incorporated protected and dynamic network virtualization by allowing for different service classes, such as a survivable critical (SC) service class over the normal service class. In our preliminary prototype, we have implemented a rudimentary version of the ENLU message passing framework for different service classes. In our current implementation, virtualization is performed on a per link basis; that is, to simulate the affect, a user can decide which links to be considered for virtualization. If it is not considered for virtualization, then all services share the link equally. For activating network virtualization, attributes values of a link are encoded differently for the prioritized service on a link basis compared to the normal services; this is done so that ENLU messages are recorded by nodes as appropriate for different services in computing routes and service provisioning.

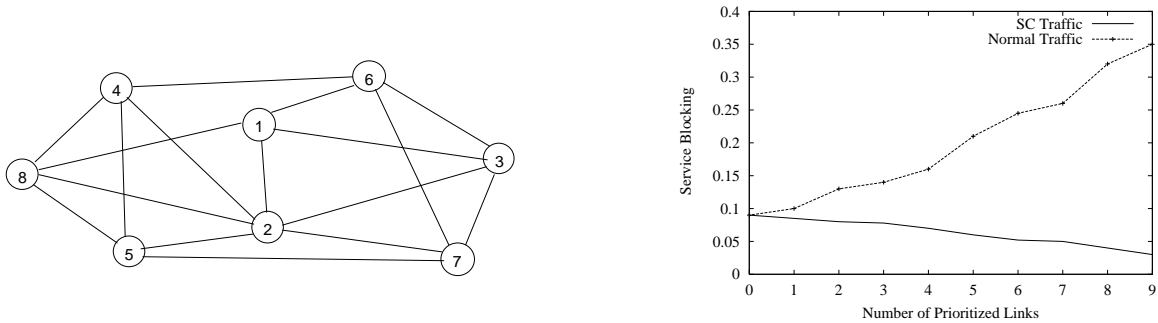


Figure 8: Network topology and service performance: (a) 8-node network topology, (b) Performance: Service Blocking.

For our preliminary study, we have considered an 8-node network (see Figure 8(a)). Traffic load was considered in such a way that the network is in a stressed situation where the prioritized service requires better performance than the normal service while both have the same amount of offered load. In Figure 8(b), we plot service blocking performance for prioritized and normal services. At the left end is the case when all links in the network are considered as fully shared by both service classes; thus, naturally, both service classes have the same performance. Then, we increase the number of links virtualized (one at a time), and plot service blocking for both classes. Thus, using VNAS, we can observe that service blocking performance is significantly lower for the prioritized class over the normal service class as more and more links are virtualized; more importantly, our tool allows us to quantify *how much* is the benefit. Currently, additional feature development on VNAS is planned in order to study a series of scenarios.

6 Summary

In this chapter, we presented traffic engineering perspective for resilient services, including an illustration. We then discussed the key components needed in the resilient network architecture so that such an adaptive environment is possible, especially to guard against both insider and outsider attacks on the routing infrastructure.

To summarize, the basic problem domain leads us to consider a virtualized trusted routing and provisioning domain, which is complemented with an adaptive traffic engineering mechanism in a semi-reservation oriented mode to provide prioritized services, especially for resiliency under attacks or overload conditions. We have presented mechanisms to extend the link state routing protocol to facilitate such an environment.

There are many issues that remain to be addressed. For instance, when many virtualized domains are created, is there any routing stability problem? In a multi-provider environment, how does the trustedness of different information can be extended, and how can the secure group communication framework be applied? Furthermore, scalability of secure group communication in a large network environment is another direction to explore. We are currently investigating these issues.

Acknowledgment

We thank Amit Sinha, Lien Harn, Cory Beard, and Tian Li for many discussions. The preliminary analysis result presented in Section 5.3 is discussed in [17]. We thank David Tipper and James Joshi for carefully reading a draft of this paper. This work was partially supported by a grant from the University of Missouri Research Board.

References

- [1] G. R. Ash. *Dynamic Routing in Telecommunication Networks*. McGraw-Hill, 1997.
- [2] J. M. Akinpelu. The overload performance of engineered networks with nonhierarchical and hierarchical routing. *AT&T Bell Labs Technical Journal*, vol. 63, pp. 1261–1281, 1984.
- [3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, R. Morris. Resilient Overlay Networks. *Proc. 18th ACM SOSP*, Banff, Canada, October 2001.
- [4] S. Blake et. al. An Architecture for Differentiated Services. *IETF RFC 2475*, December 1998.
- [5] R. Braden, D. Clark, S. Shenker. Integrated Services in the Internet Architecture: An Overview. *IETF RFC 1633*, July 1994.
- [6] R. Coltun. The OSPF opaque LSA option. *RFC 2370*, July 1998.

- [7] M. Gupta and N. S. Melam. Authentication/confidentiality for OSPFv3. *Internet Draft* (<http://www.ietf.org/internet-drafts/draft-ietf-ospf-ospfv3-auth-05.txt>), October 2004.
- [8] F. Harary. *Graph Theory*. Addison-Wesley Publishing Company, Inc., 1969.
- [9] D. Huang, Q. Cao, A. Sinha, M. Schniederjans, C. Beard, L. Harn, and D. Medhi. New Architecture for Intra-Domain Network Security. *Communications of the ACM*, vol. 49, no. 11, pp. 64–72, 2006.
- [10] 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.
- [11] D. Huang, A. Sinha, and D. Medhi. A double authentication scheme to detect impersonation attack in link state routing protocols. In *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 1723–1727, 2003.
- [12] D. Huang, A. Sinha, and D. Medhi. On Providing Confidentiality in Link State Routing Protocol. In *Proceedings of IEEE Consumer Communications and Networking Conference*, pp. 671–675, 2006.
- [13] C. Karlof and D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Elsevier’s AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, vol. 1, pp. 293–315, September 2003.
- [14] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. *RFC2104*, February 1997.
- [15] R. S. Krupp. Stabilization of alternate routing networks. In *Proc. IEEE ICC’82*, pp. 31.2.1–31.2.5, June 1982.
- [16] C. Landwehr, C. Heitmeyer, and J. McLean. A Security Model for Military Message Systems: Retrospective. *ACM Transaction on Computer System*, vol. 2, no. 3, pp. 198–222, 1984.
- [17] T. Li. *Virtual Route: An Analysis to Ensure Resource Availability for Critical Services in A Dynamic Routing Environment*. M.S. thesis, University of Missouri–Kansas City, May 2005.
- [18] D. Medhi et. al. MuSDyR: Multi-Service Dynamic Routing simulator, v 1.1. Computer Networking Research Lab, University of Missouri–Kansas City, <http://conrel.sice.umkc.edu/simulator/Musdyr1.1.pdf>.
- [19] D. Medhi. QoS routing computation with path caching: a framework and network performance. *IEEE Communication Magazine*, vol. 40, no. 12, pp. 106–113, December 2002.
- [20] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, Morgan Kaufmann Publishers, 2007.

- [21] S. Mittra. A Framework for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM*, pp. 277–288, 1997.
- [22] J. Moy. OSPF version 2. *RFC2328*, April 1998.
- [23] S. Murphy, M. Badger, and B. Wellington. OSPF with digital signatures. *RFC2154*, June 1997.
- [24] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. *Lecture Notes in Computer Science*, vol. 2139, pp. 41–62, 2001.
- [25] K. Rhee, Y. Park, and G. Tsudik. An Architecture for Key Management in Hierarchical Mobile Ad-Hoc Networks. *Journal of Communications and Networks*, vol. 6, no. 2, pp. 156–162, 2004.
- [26] E. Rosen and Y. Rekhter. BGP/MPLS IP virtual private networks (VPNs). *IETF RFC 4364*, February 2006.
- [27] A. Shaikh and A. Greenberg. Experience in black-box OSPF measurement. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop (IMW)*, November 2001.
- [28] A. T. Sherman and D. A. McGrew. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 444–458, 2003.
- [29] R. W. Shirey. Internet security glossary. *RFC2828*, May 2000.
- [30] J. H. Weber. A simulation study of routing and control in communication networks. *Bell Systems Technical Journal*, vol. 43, pp. 2639–2676, 1964.

A Secure Group Communication

For the secure link state routing protocol problem, we have recently proposed a novel key-chain based secure group key management scheme [10]. Our approach captures the merits of both centralized and distributed approaches and targets minimizing the communication and delay overhead due to key management which is critical in link state flooding. Briefly, a set of secrets are first distributed to each user (node) from a centrally controlled key server before the secure group communication phase. During the actual communication, every group member can self-derive desired group/subgroup keys and then communicate with any possible subgroups *without* any help from the key server. Our work in many-to-many secure group communication [10] helps to develop secure link state routing protocols [12] to construct future reliable and trusted routing frameworks [9].

A.1 Using One-way Function Chain To Build Key Chain

In order to reduce the storage overhead, we build a key-forest structure via multiple key chains. Each “tree” (a linear key chain) in the key-forest structure represents a particular key derivative relation among all group members. Here, we first show how we can use a one-way function to construct a one-way function chain, and subsequently, to form a key chain. Finally, we present how to systematically distribute key elements from multiple key chains to each group member.

We start with function $h(\cdot)$ which is a one-way function that maps an arbitrary length message M to a fixed length message MD . It satisfies the following properties:

1. Function $h(\cdot)$ is publicly known.
2. Given M , it is relatively easy to compute $h(M)$.
3. Given MD , it is computationally infeasible to find a message M such that $h(M) = MD$.
4. Given M and $h(M)$, it is computationally infeasible to find a message M' ($\neq M$) such that $h(M') = h(M)$.

A one-way function chain (OFC) is a sequence of values with the linear derivative relations among these values. Thus, we use $h^j(\cdot)$ to denote the result of one-way function $h(\cdot)$ on a message j times. The outputs of the multiple one-way operations construct an OFC. An OFC maintains a linear derivative relation among multiple values, i.e., $h^j(\cdot)$ can derive $h^r(\cdot)$ when $j < r$.

Consider message M_i to be the i^{th} initial key element that is used by a user to generate the cryptographic key denoted by k_{i_0} , where $k_{i_0} = f(M_i)$, and where $f(\cdot)$ is a key generating function; this key generating function is publicly known and generates the exact length of the cryptographic key. Thus, we have the following relation:

$$k_{i_j} = f(h^j(M_i)) \quad (j > 0).$$

Since $h(\cdot)$ is publicly known, we know that the derivative relations of an OFC are:

$$M_i \Rightarrow h^1(M_i) \Rightarrow \cdots \Rightarrow h^j(M_i) \Rightarrow \cdots \Rightarrow h^r(M_i). \\ (1 < j < r)$$

Then, using function $f(\cdot)$ the key chain derivative relations corresponding to the above are (see also Fig. 9):

$$k_{i_0} \Rightarrow k_{i_1} \cdots \Rightarrow k_{i_j} \Rightarrow \cdots \Rightarrow k_{i_r}, \quad \text{where } 1 < j < r. \quad (1)$$

The derivative relations among multiple keys form a linear hierarchical structure from the top level (k_{i_0}) to the bottom level (k_{i_r}). In Fig. 9, a straight forward key allocation example is shown that is composed of 6 keys ($r = 5$). If we allocate a key to each group members from u_c to u_{c+5} , each group member can derive its lower-level keys. Then, a key chain can be formed by 6 subgroups, denoted by \mathcal{S}_{i_0} to \mathcal{S}_{i_5} (Fig. 9). Note that these subgroups are *non-colluding* subgroups, that is, a subset of members cannot *collude* or *partner* to derive the keys used by a specific subgroup unless at least one of the members belongs to that subgroup.

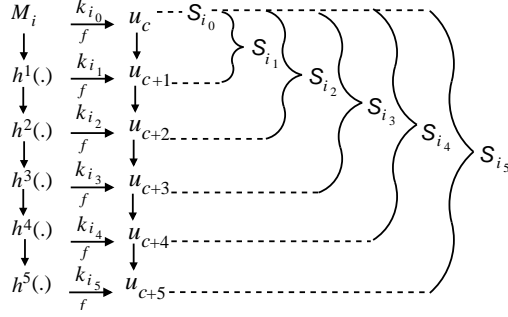


Figure 9: An example of keys allocation ($n = 6, r = 5$).

A.2 Key Distribution

We extend the linear structure of a key chain to a non-linear structure by combining multiple key chains (a key forest). These properties provide critical ingredients to our keying science due to efficiency and flexibility. We describe how to utilize graph theory to model our key distribution scheme.

Assigning more than one key from multiple key chains to a user is very similar to arranging the group member's positions in different key chains. To give a more visualized view of our scheme, we consider every group member to be a vertex of a graph $\mathcal{K}(\mathcal{E}, \mathcal{V})$; that is, each group member in \mathcal{U} is mapped to a vertex in \mathcal{V} : $v_c \in \mathcal{V}$, $u_c = v_c$ and $c = \{1, 2, \dots, n\}$. Assume that a group has n members; thus, we can associate them to n vertices in a graph, when n is odd, we can write $n = 2s + 1$ for some s . These n vertices can form a fully connected graph which we refer to as \mathcal{K}_{2s+1} . We now state the following important graph theory result [8] which is applicable in our work:

Theorem A.1 *The fully-connected graph \mathcal{K}_{2s+1} is the sum of s spanning cycles, where $s \in \mathcal{N}$. (For Proof, refer to [8], p. 89)*

Theorem A.1 states that any fully connected graph \mathcal{K}_{2s+1} can be constructed by s independent *spanning* cycles. These s *spanning* cycles have no overlapping edges, and each cycle has exactly n edges. Since our keying scheme is based on a sequence of one-way function values, we use the term *Hamiltonian* cycles² instead of *spanning* cycles. There are many ways to create s independent *Hamiltonian* cycles. We present an algorithm that creates s *Hamiltonian* cycles for graph \mathcal{K}_{2s+1} , where $2s + 1$ is a prime.

Algorithm A.2 (Hamiltonian cycles creation algorithm)

²A *Hamiltonian* cycle visits each vertex exactly once in sequence.

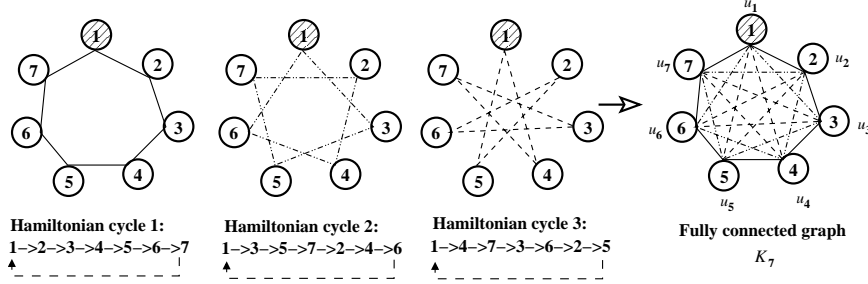


Figure 10: An example of *Hamiltonian* cycles for 7 nodes ($n = 7, s = 3$).

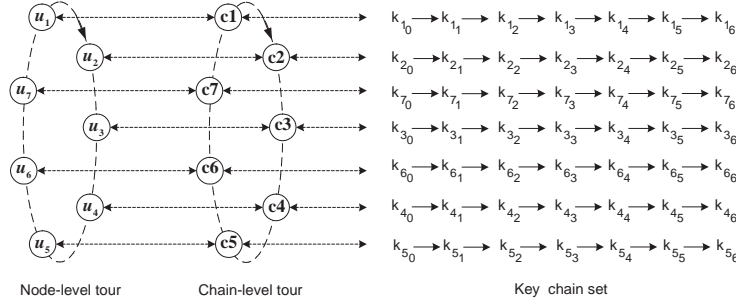


Figure 11: An example of key distribution tour for 7 group members ($n = 7, s = 3$).

<i>Initial</i>	Given n entities: v_1, v_2, \dots, v_n , where $n = 2s + 1, s \in \mathcal{N}$
<i>Procedures</i>	construct a set of s paths denoted by \mathcal{P}_t , on the points $v_1, v_2, \dots, v_{2s+1}$ as follows:
	(1) create s paths: $\mathcal{P}_t = v_1 v_{t+1} v_{2t+1} v_{3t+1} \dots v_{2st+1}$, where $t = \{1, \dots, s\}$. All the subscripts are taken as integers (mod $2s + 1$)
	(2) the Hamiltonian cycle \mathcal{Z}_t is then constructed by joining v_1 to the endpoints of \mathcal{P}_t
<i>End</i>	

For any group of users \mathcal{U} ($|\mathcal{U}| = n$ and $n = 2s + 1, s \in \mathcal{N}$), we can always find s *Hamiltonian* cycles (or *spanning* cycles or *permutation* cycles) which have no overlapping edges. For illustration, we use an overall group with seven members. This illustration is important due to its connection to the key agreement protocol described later.

An example of keying protocol

Consider 7 group members as 7 vertices in a graph as shown in Fig. 10. Using Algorithm A.2, we can build 3 separate *Hamiltonian* cycles. Node 1 is the starting point of each of these *Hamiltonian* cycles. The group key distribution requires two levels of cycle tours: the node-level tour and the chain-level tour. Fig. 11 shows the two-level tour of key distribution based on the first *Hamiltonian* cycle shown in Fig. 10. For example, the key distribution tour starts from the node-level tour and the beginning node is u_1 . Then the tour goes to the chain-level tour: the first key element in chain 1 (the head of the key chain $c1$) is distributed to u_1 ; the second key element in chain 2 ($c2$) is distributed to u_1 ; this process continues until the 7th key

Table 5: $n = 7$ group (only subscript i_j of k_{i_j} is shown in this table).

\mathcal{K}_7 nodes	1	2	3	4
Member	u_1	u_2	u_3	u_4
cycle 1	$1_0 2_6 3_5 4_4 5_3 6_2 7_1$	$1_1 2_0 3_6 4_5 5_4 6_3 7_2$	$1_2 2_1 3_0 4_6 5_5 6_4 7_3$	$1_3 2_2 3_1 4_0 5_6 6_5 7_4$
cycle 2	$1_0 2_6 3_5 4_4 5_3 6_2 7_1$	$1_4 2_3 3_2 4_1 5_0 6_6 7_5$	$1_1 2_0 3_6 4_5 5_4 6_3 7_2$	$1_5 2_4 3_3 4_2 5_1 6_0 7_6$
cycle 3	$1_0 2_6 3_5 4_4 5_3 6_2 7_1$	$1_4 2_4 3_3 4_2 5_1 6_0 7_6$	$1_3 2_2 3_1 4_0 5_6 6_5 7_4$	$1_1 2_0 3_6 4_5 5_4 6_3 7_2$
\mathcal{K}_7 nodes	5	6	7	
Member	u_5	u_6	u_7	
cycle 1	$1_4 2_3 3_2 4_1 5_0 6_6 7_5$	$1_5 2_4 3_3 4_2 5_1 6_0 7_6$	$1_6 2_5 3_4 4_3 5_2 6_1 7_0$	
cycle 2	$1_2 2_1 3_0 4_6 5_5 6_4 7_3$	$1_6 2_5 3_4 4_3 5_2 6_1 7_0$	$1_3 2_2 3_1 4_0 5_6 6_5 7_4$	
cycle 3	$1_6 2_5 3_4 4_3 5_2 6_1 7_0$	$1_4 2_3 3_2 4_1 5_0 6_6 7_5$	$1_2 2_1 3_0 4_6 5_5 6_4 7_3$	

element in chain 7 ($c7$) is distributed to u_1 . After the chain-level tour is completed, the key distribution tour goes back to the node-level tour and visits node u_2 . Next, the tour goes to the chain-level tour: the first key element in chain 2 is distributed to u_2 , the second key element in chain 3 is distributed to u_2 and so on, until the 7th key element in chain 1 is distributed to u_2 ; the key distribution tour then goes back to the node-level tour and visits node u_3 . This process is continued until the tour is completed at node 7. Thus, each *Hamiltonian* cycle corresponds to a set of key chains. For $n = 7$ and $s = 3$, we need $3 \cdot 7 = 21$ key chains in total and each group member possesses 21 keys.

In Table 5, each group member has 21 keys (in a column) which belong to three different cycles (for ease of readability, we only list subscripts (i_j) of the k_{i_j} in each row). Here, $K_i^{(t)}$ is a key chain that is constructed by following the t^{th} *Hamiltonian* cycle ($t = \{1, 2, \dots, s\}$) to allocate keys to group members. Similarly, we can add notation $^{(t)}$ to key $k_{i_j}^{(t)}$.

Fig. 10 shows that any two-group members are connected by a direct-link and they both can derive a two-member subgroup key to form a secure communication channel. For example, if we randomly select u_1 and u_4 , then, for the *Hamiltonian* cycle 3, u_1 and u_4 are adjacent, and in the third key chain, u_4 has the key $k_{1_1}^{(3)}$ and u_1 can derive it from its $k_{1_0}^{(3)}$. These relations can be applied to any combination of two group members. Note that these *Hamiltonian* cycles cannot cover every combination of selected group members. In a *Hamiltonian* cycle with n entities, the number of ℓ different group members in a continuous path is n , where $\ell < n$. Thus, in a group with n members, using Algorithm A.2 there will be s cycles formed. The number of adjacent members in s cycles is $s * n$. In the example, when $s = 3$, the number of 3-member subgroups is $\binom{7}{3} = 35$ and $s * n = 21$. Thus, we can infer that three *Hamiltonian* cycles cannot completely cover every possible combination of subgroup members. We have developed a novel way to build relations among the cycles to solve this problem—this is accomplished through the key agreement protocol described in the next section.

A.3 Key Agreement Protocol

We can use a session key k' to encrypt data and use subgroup keys as *key-encrypting key* (*KEK*) to encrypt the session key that is attached to the encrypted data. Multiple subgroup keys can be used as *KEKs* to fulfil desired subgroup composition. Using the encrypting function $E(\cdot)$ and decrypting function $D(\cdot)$, we can encrypt the data to be transmitted as follows:

$$\langle [i_{1j_1}, \dots, i_{rj_r}, E_{k_{i_1j_1}^{(t_1)}}(k'), \dots, E_{k_{i_rj_r}^{(t_r)}}(k')], E_{k'}(Data) \rangle \quad (2)$$

$$(i_1 \dots i_r \in \{1, 2, \dots, n-1\}; j_1 \dots j_r \in \{1, 2, \dots, n-1\}; t_1 \dots t_r \in \{1, 2, \dots, s\})$$

The receivers check the key chain list $\{i_1, \dots, i_r\}$ to see if they belong to the group. The checking method is straight forward: a receiver compares each element in the list with j_k in the key chains that she/he possesses; she/he belongs to the group if and only if $j_r \geq j_k$; then she/he can use one-way function $j_r - j_k$ times on the key $k_{i_rj_k}^{t_r}$ to derive key $k_{i_rj_r}^{t_r}$. To decrypt the data received, she/he first decrypts the session key:

$$k' = D_{k_{i_rj_r}^{(t_r)}}(E_{k_{i_rj_r}^{(t_r)}}(k')) \quad (3)$$

and then uses k' to decrypt the received cypher data:

$$Data = D_{k'}(E_{k'}(Data)) \quad (4)$$

We have proved the following key result (see [10] for a proof):

Theorem A.3 *Using the proposed key agreement protocol, s sets of key chains can be built where group size is n and $s = (n-1)/2$. For those arbitrarily selected subgroup members, we can always find b ($0 \leq b \leq s$) encryption relations that provide secure subgroup communication.*

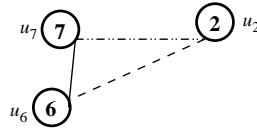


Figure 12: Sub-graph \mathcal{K}_3 .

In the 7-member example, if we consider subgroup members u_2 , u_6 and u_7 , we cannot find any continuous path in the three *Hamiltonian* cycles for these three group members. We prune graph \mathcal{K}_7 to \mathcal{K}_3 which is shown in Fig. 12. When u_2 wants to send messages to subgroup $\{u_2, u_6, u_7\}$, u_2 sends out the cypher text as follows:

$$\langle [5_1, 4_1, E_{k_{5_1}^{(3)}}(k'), E_{k_{4_1}^{(2)}}(k')], E_{k'}(Data) \rangle$$

The encrypted session key is attached to the cypher text. Only u_6 and u_7 can decrypt the encrypted session key and thus can decrypt the message. In the worst case, the number of encryption/decryption operations is equal to the number of *Hamiltonian* cycles. We have proved

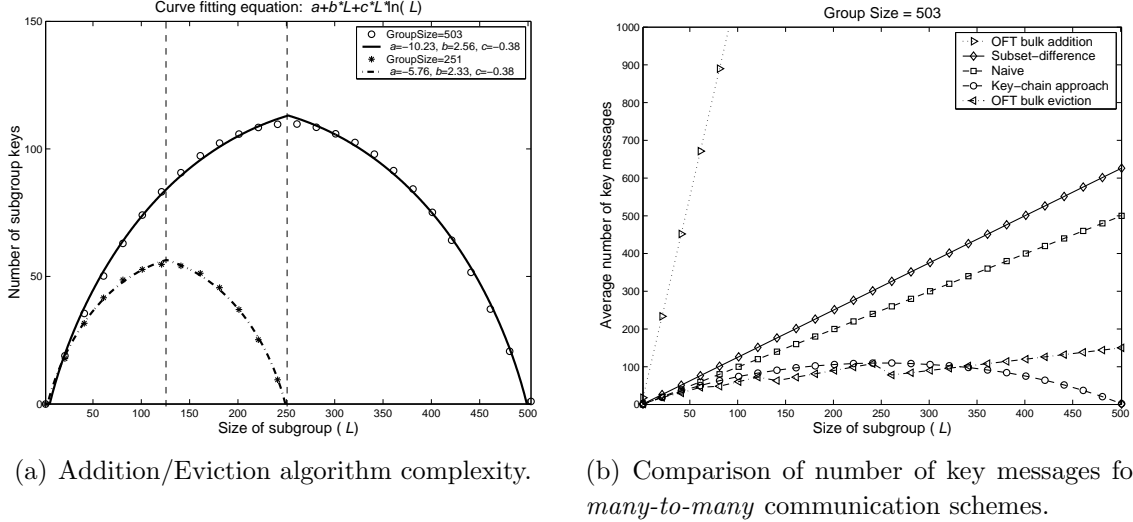


Figure 13: Performance of keying protocol.

that based on the predistributed keys, a group member can derive all possible subgroup keys (for proof, see [10]). It is important to note that this subgroup keying scheme can completely avoid *KDC* being involved in the subgroup’s communication, and the secure subgroup’s communication can be freely accomplished.

It may be noted that the work described here is for a group with n members where n is odd. We can easily take care of a group with even number of members by adding a fictitious member to increase the group size to odd number of members.

A.4 Assessment

For assessment of our key agreement protocol, we use \mathcal{S} to represent the desired subgroup that user $u_i \in \mathcal{S}$ wants to set up, where $\mathcal{S} \subset \mathcal{U}$ and $|\mathcal{S}| = L$. User u_i (the sender) is required to find r ($1 \leq r \leq L$) keys to encrypt session key k' ; then, the session key is used to encrypt the message. To find the minimal number of subgroup keys to cover all possible desired group members is an NP-complete problem (can be reduced from the minimal set covering problem). We have developed two heuristic search algorithms to find subgroup keys. They are: (1) *Addition algorithm*: when $|\mathcal{S}| \leq (|\mathcal{U}| + 1)/2$, we have $L = |\mathcal{S}|$; (2) *Eviction algorithm*: when $|\mathcal{S}| > (|\mathcal{U}| + 1)/2$, we have $L = |\mathcal{U}| - |\mathcal{S}|$. This is equivalent to evicting L group members from the system. For details, refer to [10].

Using either the addition or the eviction algorithm, each group member can derive the desired subgroup keys. We have tested the addition and eviction subgroup key searching algorithms. Through regression fitting, we have found that the function $f(L) = a + bL + cL \ln(L)$ matches our results from this test. The test results and curve fitting function are shown in Fig. 13(a) (results are shown for overall group size of 251 and 503 members). In Fig. 13(b), we compare our scheme with three typical shared key based group keying scheme: 1) a naïve scheme, 2) subset difference scheme [24], and 3) one-way function tree based scheme [28]. In

terms of communication overhead in networking environment requiring frequent subgroup formations, our key-chain based scheme performs better than these schemes.