An Efficient Message Authentication Scheme for Link State Routing*

Steven Cheung
Department of Computer Science
University of California
Davis, CA 95616
cheung@cs.ucdavis.edu

Abstract

We study methods for reducing the cost of secure link state routing. In secure link state routing, routers may need to verify the authenticity of many routing updates, and some routers such as border routers may need to sign many routing updates. Previous work such as public-key based schemes either is very expensive computationally or has certain limitations. This paper presents an efficient solution, based on a detection-diagnosis-recovery approach, for the link state routing update authentication problem. Our scheme is scalable to handle large networks, applicable to routing protocols that use multiple-valued cost metrics, and applicable even when link states change frequently.

1. Introduction

Routers exchange routing control packets to disseminate their current states. Based on these control packets, routers can construct their routing tables to cooperatively forward packets from source to destination. If routing infrastructure components, such as routers or inter-router links, are faulty, misconfigured, or compromised, the network may suffer from degradation of service or even unavailability.

Potential vulnerabilities of routing infrastructures can be classified as follows:

- Packet generation—Masquerading as a particular router to send bogus routing control packets to other routers. Replaying stale control packets. Flooding the network with excessive control or data packets.

- Packet alteration—Modifying control or data packets in transit. For example, the cost, the ordering, or the freshness information of those packets may be altered.

- Packet removal—Removing control packets to prevent information about network changes from propagating to other routers. Removing data packets in transit to effect denial of service.

- Misrouting—Misrouting control or data packets so that they will take longer (or forever) to reach their destinations.

- Breach of confidentiality—Eavesdropping data and control packets. Performing traffic analysis.

To protect routing control traffic from some of those threats, countermeasures that support data authenticity (used to provide both proof of data origin and data integrity), ordering, and freshness of control packets have been proposed. Examples are Perlman's [16, 17] work on link state routing protocols, Finn's [3] report on dynamic routing protocols in general and Cartesian routing in particular, Kumar's and Crowcroft's [8] paper on inter-domain routing protocols, Murphy's and Badger's [14] paper on OSPF, Smith's and Garcia-Luna-Aceves's [21] paper on BGP, Hauser's, Przygienda's, and Tsudik's [4] paper on link state routing, Sirois's and Kent's [20] paper on Nimrod, and Smith's, Murthy's, and Garcia-Luna-Aceves's [22] paper on distance vector routing protocols.

This paper presents an efficient message authentication scheme for protecting control packets in link state routing. Previous work such as [16, 17, 14, 4] either is very expensive computationally or has certain limitations, which will be discussed in Section 2. We use a detection-diagnosis-recovery approach, which is intrusion detection (e.g., [2, 10, 13, 5]) augmented with system diagnosis and reconfiguration, inspired by work in fault tolerance. This approach is also used in Cheung's and Levitt's [1] paper on protecting routing infrastructures from routers that incorrectly drop packets and misroute packets. Our main goal is to minimize the cost of performing link state update authentication when the network components function normally, which occurs most of the time. In our scheme, a router r uses a key k and a

*DRAFT: To appear in Proc. 13^{th} Annual Computer Security Applications Conference, San Diego, Calif., December 8-12, 1997.

symmetric-key based data authentication scheme (e.g., a keyed-hash scheme) to sign a link state update. The link state update and the signature are disseminated to all other routers. A receiving router optimistically accepts the routing update as if it were authenticated. Later, router r will release the key k. When the key k arrives, the receiving router verifies the authenticity of the key using a secure and efficient method. Then the verified key will be used to verify the authenticity of the link state update using the symmetric-key based data authentication scheme. Note that signature generation and verification can be done very efficiently using a symmetric-key based data authentication scheme. If bogus routing updates are detected, a distributed diagnosis protocol will be invoked to locate the mischievous routers. Then network reconfiguration will be performed to logically disconnect those routers to restore the operational status of the network.

This paper is organized as follows: Section 2 reviews related work on link state update authentication. Section 3 details and analyzes our scheme, called optimistic link state verification. Section 4 compares our work with related work, and discusses variations and limitations of our scheme.

2. Background: Link State Update Authentication

In link state routing[1], every router constructs link state updates[2] (LSU) that describe the status of the links incident to the router, and distributes those updates to all other routers. As networks are generally not fully-connected, a technique known as *flooding* is used for LSU distribution. When a router receives a LSU that it has not received previously, the router forwards the LSU (essentially) unchanged to its neighbors, except the one from which the LSU was received. To make flooding more robust, a router sends an acknowledgement to the neighbor from which it receives a LSU. If the sender does not receive an acknowledgement after a certain time threshold, it will re-transmit the LSU. Based on the LSU received, a router computes the shortest paths to all destinations. Because those computations are performed independently by all routers on the *same* set of LSU, networks using link state routing converge to a stable state quickly (as opposed to distance vector routing). To protect routers from using erroneous LSU to compute their routing tables, data authentication is needed to cope with bogus LSU generation and LSU modification. Specifically, an entity may masquerade as a particular router and generate a bogus LSU. Moreover, a LSU may be modified by a compromised intermediate router or an active inter-router link attack.

[1]Examples of link state routing protocols are OSPF [12], IS-IS [6], and a proprietary routing protocol used in the ARPANET [11].

[2]Link state updates are also called link state advertisements.

Data authentication schemes can be broadly classified as symmetric-key based and asymmetric-key based. In a symmetric-key based data authentication scheme, also called a message authentication code (MAC) scheme, a message is signed and verified using the same key. To use a direct MAC scheme for LSU authentication on a network that has n routers, in the worst case[3], each router needs to maintain $(n-1)$ keys[4] and the network as a whole needs to maintain $O(n^2)$ keys. Moreover, every router would need to sign and to send $(n-1)$ LSU—one for each router—instead of one LSU as in existing link state routing protocols. Thus a direct MAC scheme for LSU distribution may be expensive in terms of processing and network bandwidth overheads.

Perlman's seminal work [16, 17] uses an asymmetric-key based scheme, also called a digital signature scheme, for data authentication in LSU distribution and public key distribution. In a digital signature scheme, a message is signed and verified using different keys. Murphy and Badger [14] proposed a design, based on digital signatures, to securely distribute LSU and public keys in OSPF. A digital signature scheme seems to be a good candidate for solving the LSU distribution problem—only $O(n)$ key-pairs are needed for the entire network, and a signed LSU can be verified by all routers. However, as pointed out by [14], it may be very expensive[5] to generate and to verify digital signatures. The number of signatures needed to be verified by a router depends on several factors: the number of routers in the network, the grouping of routers into areas, the frequencies of link state changes and LSU refresh, the number of internal and external distinguishing subnets[6], and the particular routing protocol used. In OSPF, because the route to each external subnet is advertised in a separate LSU, there may be tens of thousands of those LSU. To relieve the performance impact, Murphy and Badger suggested a few possibilities: using extra hardware in routers, changing the OSPF protocol, and verifying signatures periodically or on demand. This paper explores the last option.

Recently, Hauser, Przygienda, and Tsudik [4] presented a technique to reduce the cost of LSU authentication. Their technique is based on a tool called *hash chains*, which were designed by Lamport [9]. A hash chain of length ℓ is a

[3]Partitioning a large network into sub-networks/areas could relieve the problem.

[4]A direct MAC scheme that does not employ pairwise-keys is not suitable for the LSU authentication problem. Specifically, using a shared key for all routers or having routers to share a key with each of their neighbors and to use hop-by-hop LSU authentication cannot protect the network from compromised intermediate routers. The former is used in OSPF version 2[12] cryptographic authentication; routers on a network/subnet uses a secret shared key and a MAC scheme to authenticate routing protocol packets.

[5]Experimental results reported in [14] state that it takes at least 270 microseconds to verify an RSA [19] signature with the 512-bit key size using a SPARC-20 and the GNU MP library.

[6]A subnet is internal if the subnet and the router reside in the same autonomous system and external otherwise.

list $[H(r), \ldots, H^{\ell-i}(r), \ldots, H^{\ell-1}(r), H^{\ell}(r)]$, where r is a secret quantity, H is a one-way hash function, and $H^{\ell-i+1}(r) = H(H^{\ell-i}(r))$. If $H^{\ell}(r)$ can be sent to a verifier securely, the authenticity of $H^{\ell-i}(r)$ can be verified by applying the function H to $H^{\ell-i}(r)$ one or more times. Examples of proposed one-way hash functions are MD5 [18] and SHA [15]. In Hauser et al.'s scheme, two hash chains with different seeds r_{up} and r_{down} are used to represent the *up* and *down* state of a link. The originating router uses its private key to sign a message that includes $H^{\ell}(r_{up})$, $H^{\ell}(r_{down})$, and the current time T and floods the message. A receiving router can verify the authenticity of that message using the public key of the originating router. Let Δ be the duration of time intervals between consecutive LSU releases. At time $T + i\Delta$, the originating router releases either $H^{\ell-i}(r_{up})$ or $H^{\ell-i}(r_{down})$, depending on the status of the link. Their technique virtually eliminates the need to perform expensive public-key encryption and decryption. Signing and verifying digital signatures are replaced by applications of a hash function, which are orders of magnitude cheaper. Despite the cost reduction advantage, there are a few drawbacks to their scheme. First, their scheme cannot efficiently handle multiple-valued link states because the costs of generating, verifying, and storing many hash chains may be higher than those of using digital signatures [4]. The need for multiple-valued link states arises when link costs depend on traffic load, and when a border router advertises (summarized) link costs for destinations that reside in other autonomous systems or in other areas within the same autonomous system. Second, Hauser et al. showed that the maximum clock skew among routers must be less than 3Δ. Otherwise, an adversary may be able to forge an incorrect LSU that is considered to be fresh and authentic by some routers. Finally, their scheme is not suitable for handling frequent link state changes because the hash chains are pre-computed assuming certain fixed time intervals between consecutive LSU. Choosing Δ to be a very small quantity has two problems—the originating router needs to generate and to store long hash chains, and routers have to be very tightly synchronized (c.f., the clock skew problem discussed above).

3. Optimistic Link State Verification

Our approach is inspired by fault tolerance work—error detection-diagnosis-recovery specifically. Our scheme is called optimistic link state verification (OLSV). In OLSV, a receiving router optimistically accepts a LSU before it can be verified. When the router receives a key used to authenticate the LSU, it will first verify the authenticity of the key. Then the verified key is used to verify the authenticity of the LSU. If the verification process detects a bogus LSU, routers report from which neighbors that bogus LSU was received. A distributed diagnosis protocol is invoked to identify the mischievous router(s); we will discuss how link attacks are handled later. Based on the diagnosis result, the mischievous routers are logically removed from the network to restore its operational status.

3.1. Assumptions

We consider a network of routers that use a link state routing protocol. We use a graph G to represent the network, with vertices representing routers and edges representing communication links. If two routers share a link, we call them *neighbors*. A router that correctly executes the routing protocol is called a *good router*; otherwise, it is called a *bad router*. A bad router may be caused by software/hardware faults, misconfiguration, or malicious attacks. A failed/compromised link is called a *bad link*; otherwise, it is called a *good link*. An LSU includes several fields: originating router id, sequence number, age, and link state data. The sequence number field is used to provide an ordering among LSU. With a protected sequence number field, replay and reordering attacks can be countered. The age field is used to support freshness; stale LSU can be prevented from propagating in the network. A router increments the age field of an LSU before forwarding it to its neighbors. Because the age of a LSU needs to be modified by intermediate routers, the age field is excluded in LSU authentication computation. We make the following assumptions:

1. The network remains connected after the removal of bad routers and bad links.

2. There exists a secure public-key distribution protocol. Perlman [16, 17] and Murphy and Badger [14] proposed security protocols for distributing the public keys of routers. OLSV assumes that every router knows the public keys of all routers.

3. Every router has a local clock and the maximum clock skew between any two good routers is bounded by ϵ. That is, the difference between the clocks of any two routers is less than or equal to ϵ at any time. Moreover, we assume that the ratio of clock rates between the fastest clock and the slowest clock among good routers is bounded by α.

4. The total delay—propagation, queueing, and processing delays—for sending a packet using flooding is bounded by δ.

5. There are no adjacent bad routers. This assumption is used to simplify the description of OLSV. We will discuss how this assumption can be removed in Section 4.

6. There exists a one-way hash function. Examples of proposed one-way hash functions are MD5 and SHA. We use H to denote a one-way hash function. Given a random quantity y, it is computationally infeasible to find x such that $y = H(x)$. Moreover, for a random quantity x, it is computationally infeasible to find an $x' \neq x$ such that $H(x) = H(x')$.

7. There exists a good random number generator.

8. A secure digital signature scheme is used. Digital signatures can be generated using a cryptographic hash function and a public-key cipher such as MD5 and RSA. We denote the digital signature of a message m signed using p's private key by $S_p(m)$. Without knowing p's private key, it is computationally infeasible to generate $S_p(m')$ for a new message m'.

9. A secure MAC scheme, which includes a MAC generator *MACG*, is used. Tsudik's [23] keyed-hash scheme and HMAC [7] are examples of MAC schemes. Moreover, they are significantly less expensive than digital signature schemes such as RSA/MD5. We use $MACG_k(m)$ to denote the MAC generated by *MACG* using a key k on a message m. Without knowing k, it is computationally infeasible to generate $MACG_k(m')$ for a new message m'.

3.2. Protocol Overview

Our OLSV protocol is sub-divided into three parts, namely sender, receiver, and recovery. Every router runs a sender process, a receiver process, and a recovery process. The sender process generates keys and uses them to generate MAC for LSU. Those LSU and the associated MAC are then flooded to other routers as in existing link state routing protocols. The keys are released to other routers at designated times. Section 3.3 details the sender process. The receiver process optimistically accepts LSU (as if they were authenticated) and uses them to compute the local routing table. When the corresponding keys arrive, the receiver process verifies the authenticity of the LSU received. Section 3.4 details the receiver process. When the receiver process detects mischievous LSU, the recovery process is invoked. A recovery process is responsible for diagnosis and reconfiguration. Diagnosis is used to locate misbehaving routers. Based on the diagnosis results, reconfiguration is used to logically disconnect those misbehaving routers from the network to restore its operational status. Section 3.5 details the recovery process. The recovery process is designed to counter router attacks. To counter active link attacks, neighboring routers use a MAC scheme to authenticate LSU forwarded between them[7]. Because a router usually has few neighbors, a secret key can be configured or established using a key-exchange protocol for each neighboring router pair and many existing efficient MAC schemes are applicable to authenticate LSU sent between neighboring routers. For the sake of clarity, we omit this LSU authentication between neighboring routers in the subsequent description of our protocol.

3.3. Sender Process

The sender process generates a random quantity r and constructs a hash chain of length ℓ using r and a one-way hash function H.

Subsequently, the sender process composes a *key-chain anchor* (KCA) message that contains the router id, the current time T, and $H^\ell(r)$ and signs it with the private key of the router. Then the signed KCA message $(id, T, H^\ell(r), S_{id}(id, T, H^\ell(r)))$ is distributed to other routers via flooding.

The quantities $H^{\ell-i}(r)$, where $1 \leq i < \ell$, are used as keys to generate MAC for LSU. A *hash-chained key* (HCK) message $(id, i, H^{\ell-i}(r))$ is released to other routers at time $T + i\Delta$, where Δ is the duration of the time intervals between consecutive key releases. In fact, the sender process only needs to release a HCK if the corresponding $H^{\ell-i}(r)$ is used to generate a MAC.

To make OLSV secure, $H^{\ell-i}(r)$ is used to generate MAC for LSU only before time $T + i\Delta - \tau$, where τ is a value that we will derive later. When the sender process wants to send an LSU at time t, where $T + (i-1)\Delta - \tau \leq t < T + i\Delta - \tau$, it uses $H^{\ell-i}(r)$ as the key to generate the MAC. The signed LSU message $(LSU, i, MACG_{H^{\ell-i}(r)}(LSU, i))$ is then flooded to other routers. Figure 1 depicts the chronological order of the actions performed by the sender process.

3.4. Receiver Process

When the receiver process receives a KCA with a digital signature $S_{id}(id, T, H^\ell(r))$, it verifies the authenticity of the KCA using the public key of router id. A verified KCA with T reasonably close to the current clock value of the router is accepted and stored.

The receiver process optimistically accepts a signed LSU $(LSU,\ i,\ MACG_{H^{\ell-i}(r)}(LSU, i))$ if the receiving time is less than $T + i\Delta - \epsilon$. (Note that the router id in LSU can be used to determine the corresponding T.)

When a HCK message (id, i, k) is received, the authenticity of the HCK is verified by applying the hash function

[7] As we will see, our scheme will still work even if we do not perform additional LSU authentication between neighboring routers. Specifically, a link failure may be viewed as a router failure in OLSV. The routers incident on a failed link will detect the failure and cease the neighbor relationship. Consequently, the failed link will not be used. However, using a MAC scheme to authenticate LSU sent between neighboring routers can prevent link attacks, and in some cases does not hurt the connectivity of the network.

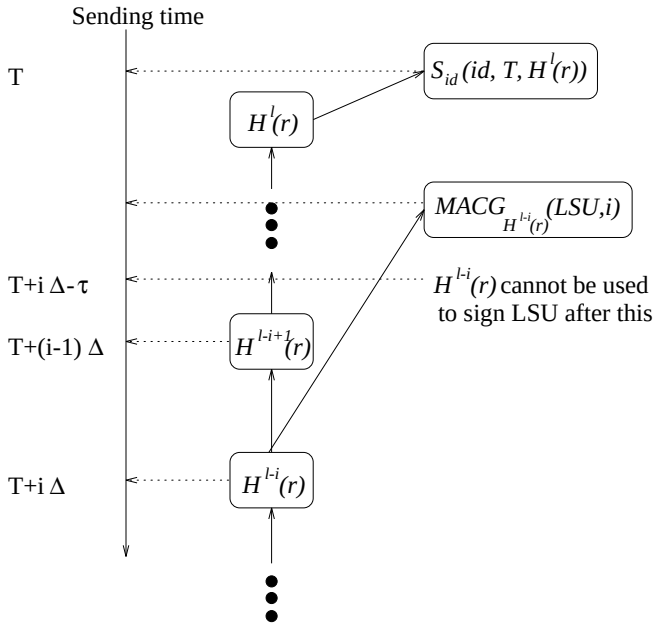

Figure 1. The Sender Process.

H to k one or more times. Like Hauser et al.'s scheme, the verification process is more efficient if the last verified HCK is stored and used. For example, if the last verified HCK message $(id, i-1, H^{\ell-i+1}(r))$ is stored, then verifying the HCK message (id, i, k) only consists of computing $H(k)$ and comparing $H(k)$ with $H^{\ell-i+1}(r)$. Otherwise, it takes i applications of H to verify the HCK if the KCA (or $H^{\ell}(r)$) is used. A verified HCK message $(id, i, H^{\ell-i}(r))$ is then used to verify the authenticity of LSU. For a signed LSU message (LSU, i, mac), $H^{\ell-i}(r)$ is used to generate the MAC of (LSU, i) and the resulting value is compared to mac. If bogus LSU are detected, the recovery process is invoked.

Theorem 1 *If we set $\tau \geq 2\epsilon + \alpha\delta$, then an adversary cannot generate a bogus LSU whose originating router id corresponds to a good router and have the bogus LSU accepted by good routers without being detected. Moreover, a good router always accepts a signed LSU generated by another good router.*

Proof: Recall that the originating router id releases the HCK message $(id, i, H^{\ell-i}(r))$ at time $T + i\Delta$. At that time, the clock values of good routers are at least $T + i\Delta - \epsilon$. By requiring good routers not to accept LSU signed with $H^{\ell-i}(r)$ after $T + i\Delta - \epsilon$, the key $H^{\ell-i}(r)$ will not be useful to an adversary to generate bogus LSU by the time the adversary receives the HCK message. Note that r is a random quantity and $H^{\ell-j}(r)$, where $i < j < \ell$, are released after the time $H^{\ell-i}(r)$ is released. Moreover, knowing $H^{\ell-k}(r)$, where $1 \leq k < i$, is not useful to determine $H^{\ell-i}(r)$. Thus the adversary cannot determine $H^{\ell-i}(r)$ in time to generate a bogus LSU and have it accepted by a good router without being detected. When router id's local time is $T + i\Delta - 2\epsilon$, the clock values of good routers are at most $T + i\Delta - \epsilon$. Thus having router id to send the LSU signed with $H^{\ell-i}(r)$ before $T + i\Delta - (2\epsilon + \alpha\delta)$, all good routers will accept the LSU. □

3.5. Recovery Process

When bogus LSU are detected by the receiver process, a *bad routing update advertisement* (BRUA) message $(BLSU, i, pred)$ is constructed, where $BLSU$ is a "selected" bogus LSU detected using the key $H^{\ell-i}(r)$, and *pred* is the id of the neighboring router from which $BLSU$ is received. If multiple bogus LSU are detected using $H^{\ell-i}(r)$, the one with the smallest sequence number is selected. Moreover, if more than one bogus LSU are associated with the smallest sequence number, the one with the largest checksum is selected. The BRUA is signed with the router's private key and the signed BRUA is then flooded to other routers.

Then the recovery process waits for $2\alpha\delta$ time units to ensure BRUA from other routers can reach itself. The $2\alpha\delta$ time delay covers the time for the corresponding HCK and the time for the BRUA to reach every router. We assume that it takes the same amount of time for every router to use the HCK to verify LSU and to construct and sign a BRUA. Otherwise, the waiting time should be increased accordingly to include the LSU verification time and the BRUA signing time.

Based on the BRUA received, the recovery process constructs a *bad routing update propagation* (BRUP) graph. Each BRUP corresponds to one bogus LSU. In the case where BRUA corresponding to multiple bogus LSU are received, we use the same arbitration rules—choosing the bogus LSU with the smallest sequence number and using the checksums to break ties—to select one. A BRUA $(BLSU, i, p)$ sent by q is represented by a labeled edge $q \xrightarrow{a} p$ in the BRUP graph, where a is the age of BLSU when q receives it from p.

Then the recovery process performs a depth-first search on the BRUP graph. The search starts with the node that has the largest id and has an outgoing edge. Moreover, if the node has incoming edges, the age value associated with the outgoing edge is larger than those of its incoming edges. Because each node has at most one outgoing edge and the age values are used to cope with loops, the procedure for finding the starting node is well-defined. The search continues until one of the following is encountered: (1) An edge $q \xrightarrow{a} p$ and p does not have an outgoing edge; (2) A path segment

$q \xrightarrow{a_2} p \xrightarrow{a_1} o$, where $a2 \leq a1$; (3) A node visited previously is reached. For the first two cases, (p, q) is recorded. The search procedure is repeated starting with an unvisited node in the BRUP graph.

In case (1), q claims that p sent q the bogus LSU. Moreover, p does not claim that it received the bogus LSU from a neighbor. Because link attacks are prevented using a MAC scheme, we can infer that p or q is a bad router. In case (2), q claims that p sent q the bogus LSU with age a_2. Moreover, p claims that o sent p the bogus LSU with age a_1. Because p should increment the age field before forwarding the LSU to q, a_2 should be strictly larger than a_1. Thus either p lies or q wrongly accuses p being a bad router. By a case analysis, one can show that those two cases are sufficient to cover all scenarios in which a bad router sends out the bogus LSU.

Once bad routers are located, the routers respond by reconfiguring the network to logically remove the bad routers. Specifically, when the diagnosis described above reveals that p or q is a bad router, the neighbor relationship between p and q will be ceased by the good router. (Because we assume that no bad routers are adjacent, p or q is a good router.) As a result, if a bad router keeps sending out bogus LSU to its neighbors, the bad router will eventually be disconnected from the network.

3.6. An Example

Figure 2 depicts a network that has six routers R_i, where $1 \leq i \leq 6$. Consider that R_1 floods a signed LSU $\mathcal{L}$ to other routers. We assume that R_2 is a bad router. Instead of forwarding $\mathcal{L}$, R_2 forwards a modified LSU $\mathcal{L}'$ to its neighbors R_3 and R_5. Without knowing $\mathcal{L}'$ is a bogus LSU, R_3 in turn forwards $\mathcal{L}'$ to R_6. When the key used to verify $\mathcal{L}$ is disseminated by R_1, the bogus LSU $\mathcal{L}'$ will be detected. R_6 will send out a BRUA indicating $\mathcal{L}'$ was received from R_3. R_3 and R_5 will send out a BRUA indicating $\mathcal{L}'$ was received from R_2. Then a BRUP graph as shown in Figure 2 will be constructed. Performing a DFS on the BRUP graph outputs the following results: R_2 or R_3 is a bad router, and R_2 or R_5 is a bad router. Subsequently, R_3 and R_5 will cease their neighbor relationship with R_2. Note that based on the BRUP graph, R_1 cannot determine whether R_2 is a bad router or its neighbors are bad routers. Thus R_1 does not disconnect itself from R_2. However, if R_2 continues to misbehave, R_2 may eventually be completely disconnected from the network.

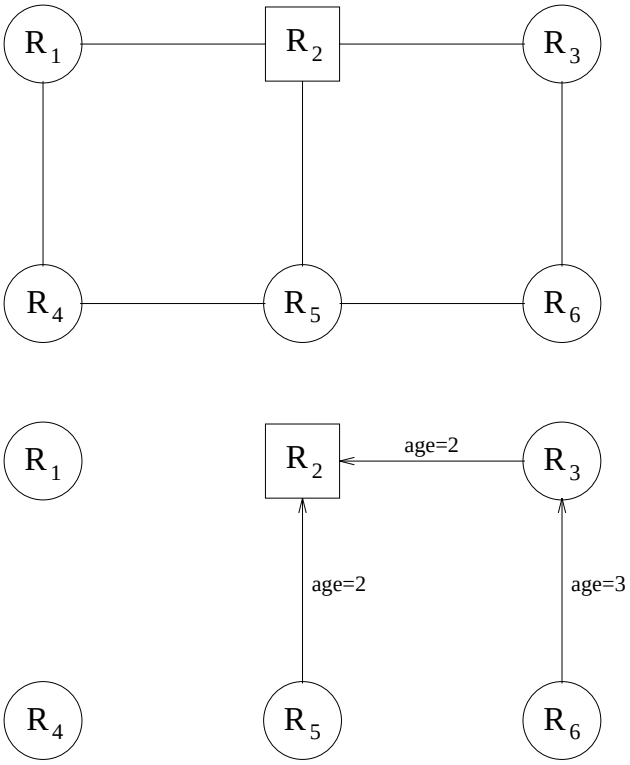

Figure 2. A Network and an Associated BRUP Graph.

3.7. Cost Analysis

The major costs of our OLSV scheme are in the time to generate and to verify digital signatures for KCA and BRUA, the time to generate and to verify MAC for LSU, the time to perform DFS on BRUP graphs, the storage required to

store hash-chained keys, the storage required to store LSU that are yet to be verified, and the network bandwidth for sending HCK, signed LSU, and BRUA.

Time: Because a KCA is generated and verified once for every ℓ LSU, the amortized cost is very small. Our experiments performed on a SPARC-5 running SunOS 4.1.3 and using the RSAREF2 library show the following results: (1) Generating and verifying a signature for a 16-byte block (i.e., the same size as an MD5 digest) using RSA with the 512-bit key size takes 0.38 second and 0.033 second respectively. (2) Generating an MD5 digest for a 512-byte block and a 16-byte block takes 240 microseconds and 52 microseconds respectively. The time to generate an MD5 digest roughly corresponds to the times to generate and to verify a keyed-MD5 MAC. Thus using our scheme may be orders of magnitude less expensive than using a digital signature scheme for LSU authentication. Our main goal is to minimize the performance impact of LSU authentication when the network operates normally, which occurs most of the time. We argue that the costs of recovery (i.e., signing and verifying BRUA and performing DFS on BRUP graphs), performed only when the network is under attack, are tolerable.

Storage: Every router needs to store a hash chain[8] of length ℓ. Note that MD5 produces a 128-bit digest and SHA produces a 160-bit digest. Thus one can choose a large ℓ in practice. On the receiver process side, only the last verified HCK from each router needs to be stored. Thus the storage cost for the hash-chained keys may be tolerable. Because a router needs to store the LSU that are yet to be verified, a potential denial of service vulnerability exists. To prevent a bad router from sending out a lot of bogus LSU to consume all memory of a router, one may impose a limit on how frequently a router may forward LSU originated by a particular router. In fact, modern link state routing protocols such as OSPF impose a minimum time between distinct originations of any particular link state advertisement. Another technique is to store conflicting LSU only—two or more LSU conflict with each other if they have different link state data but the same originating router id and the same sequence number. When a router receives (bogus) LSU whose originating router is the router itself and the sequence number is larger[9] than the router's current sequence number, the router floods an LSU with the same sequence number and the current link state data so other routers can detect a conflict. Note that only conflicting LSU that have the smallest sequence number are needed to be stored. Moreover, among those LSU that have the smallest sequence number, only the two LSU that have the largest checksums are needed. (We need to store two to ensure that the bogus LSU with the largest checksum is stored; the authentic LSU may have the largest checksum.) As a result, a router only needs to keep an LSU for $2\alpha\delta$ time units (to ensure the LSU from the originating router can reach itself) instead of $\tau + \Delta + \alpha\delta$ time units to detect bogus LSU.

Network Bandwidth: The recurring extra network traffic are from HCK messages and two fields in signed LSU (i.e., an index and a MAC). An HCK and those two fields in a signed LSU are about the size of a message digest. A BRUA is about the size of a signed LSU. Again, a BRUA is sent only when the network is under attack. Thus the extra network bandwidth needed in our scheme should be insignificant.

4. Discussion

Although both Hauser et al.'s scheme and our OLSV scheme use hash chains as a tool, they differ in how hash chains are used. In Hauser et al.'s scheme, hash chain entries are used as signatures. In our OLSV scheme, hash chain entries are used as keys for generating and verifying MAC. Our OLSV scheme has several advantages over Hauser et al.'s scheme. First, OLSV can efficiently handle multiple-valued link states. The need for multiple-valued link states arises when link costs depend on traffic load, and when a border router advertises (summarized) link costs for destinations that reside in other autonomous systems or in other areas within the same autonomous system. Second, OLSV can be used to handle very frequent link state changes. In OLSV, a hash-chained key can be used to generate and to verify MAC for multiple LSU. In Hauser et al.'s scheme, consecutive link state changes are at least Δ time units apart. Moreover, Hauser et al.'s scheme may not be able to use a small Δ (c.f. Section 2). Third, OLSV does not require ϵ to be smaller than a certain value. (Hauser et al.'s scheme requires $\epsilon < 3\Delta$.) However, we note that there is a tradeoff between the tightness of clock synchronization and the time to recover. It is because a signed LSU may be released $\tau + \Delta$ time units before the time the corresponding hash-chained key is released. A future work item is to reduce the recovery time of OLSV, especially when the routers are very loosely synchronized.

In Section 3, we assume that there are no adjacent bad routers in the network. We note that the recovery protocol can be extended to cope with adjacent bad routers. After lo-

[8] Note that Hauser et al.'s scheme requires a router to maintain two hash chains for each link incident on the router. Our scheme only requires one hash chain per router.

[9] Sending bogus LSU with old sequence numbers is not an effective attack because those LSU will not be used.

cating a suspicious router pair by performing DFS in a BRUP graph, we know that at least one of them are mischievous. If none of them ceases the neighbor relationship, one can conclude that both of them are bad routers. The neighbors of those two routers should disconnect themselves from those two routers in the next round. By repeating this procedure, a good router that is adjacent to those bad routers will be able to determine which neighbors are bad routers and cease its neighbor relationship with them.

Some techniques presented in OLSV are useful in other contexts. The basic idea of optimistic verification is applicable in general to applications in which a subject needs to authenticate data to many other subjects efficiently. For example, it can be used to reduce the costs of Smith's and Garcia-Luna-Aceves's [21] scheme on securing BGP and Smith's, Murthy's and Garcia-Luna-Aceves's [22] scheme on securing distance vector routing protocols. If we remove "optimistic verification" from OLSV, we have an efficient LSU authentication scheme that is applicable to a network with tightly synchronized routers[10]. Specifically, after a router receives a signed LSU, it will wait until the corresponding HCK arrives. The authenticity of the HCK is then verified. Verified HCK are used to verify the authenticity of LSU. Only verified LSU are used to update the routing table. In this case, we would not need to perform the recovery portion of the protocol.

Independently, Wu et al. [24] proposed an intrusion detection approach to secure link state routing protocols. Their main idea is that a router generates a session key and uses it to sign k LSU. After the session key is used k times, the originating router will sign the session key using its private key and send the signed session key to other routers. If bogus LSU are detected, bad routers are identified using a statistical analysis technique. Even though our OLSV scheme and Wu et al.'s scheme use a similar approach, our techniques and protocols are quite different from theirs. Among other things, our OLSV may be able to detect attacks sooner than Wu et al.'s scheme. Because generating and verifying digital signatures are expensive, k must be reasonably large. Thus the time between a LSU is sent and the corresponding session key is released may be quite long, especially when link state changes are infrequent. Our OLSV, on the other hand, does not rely on batch verification for cost reduction. A hash-chained key can be released every Δ time units and Δ may be chosen to be quite small. Thus OLSV may be able to detect bogus LSU and initiate the recovery process sooner.

[10] Tight synchronization among routers may be achieved using a secure network time protocol.

5. Acknowledgments

This research is supported by DARPA under grant ARMY/DAAH 04-96-1-0207. We would like to thank Phil Rogaway and Karl Levitt for helpful discussions. We thank Karl Levitt, Kirk Bradley, Bridget Bradley, and the anonymous referees for their useful comments on this paper.

References

[1] S. Cheung and K. Levitt. Protecting routing infrastructures from denial of service using cooperative intrusion detection. In *Proceedings of the New Security Paradigms Workshop*, September 1997.

[2] D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, February 1987.

[3] G. Finn. Reducing the vulnerability of dynamic computer networks. Research Report RR-88-201, ISI, University of Southern California, June 1988.

[4] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the cost of security in link-state routing. In *Proceedings of the Symposium on Network and Distributed System Security (SNDSS '97)*, pages 93–99, February 1997.

[5] K. Ilgun, R. Kemmerer, and P. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[6] Joint Technical Committee ISO/IEC JTC 1 *Information Technology*. Intermediate system to intermediate system intra-domain routeing information exchange protocol for use in conjunction with the connectionless-mode network service (ISO 8473), ISO/IEC 10589, April 1992.

[7] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, IETF, February 1997.

[8] B. Kumar and J. Crowcroft. Integrating security in inter-domain routing protocols. *Computer Communication Review*, 23(5):36–51, October 1993.

[9] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[10] T. Lunt. A survey of intrusion detection techniques. *Computer and Security*, 12(4):405–418, June 1993.

[11] J. McQuillan, I. Richer, and E. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[12] J. Moy. OSPF version 2. RFC 2178, IETF, July 1997.

[13] B. Mukherjee, L. Heberlein, and K. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, May-June 1994.

[14] S. Murphy and M. Badger. Digital signature protection of the OSPF routing protocol. In *Proceedings of the Symposium on Network and Distributed System Security (SNDSS '96)*, pages 93–102, February 1996.

[15] NIST (National Institute of Standards and Technology). Secure hash standard. FIPS Pub. 180, NIST, May 1993.

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

[17] R. Perlman. *Interconnections: Bridges and Routers*. Addison-Wesley, 1992.

[18] R. Rivest. The MD5 message-digest algorithm. RFC 1321, IETF, April 1992.

[19] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[20] K. Sirois and S. Kent. Securing the Nimrod routing architecture. In *Proceedings of the Symposium on Network and Distributed System Security (SNDSS '97)*, pages 74–84, February 1997.

[21] B. Smith and J. Garcia-Luna-Aceves. Securing the border gateway routing protocol. In *Proceedings of Global Internet 1996*, November 1996.

[22] B. Smith, S. Murthy, and J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *Proceedings of the Symposium on Network and Distributed System Security (SNDSS '97)*, pages 85–92, February 1997.

[23] G. Tsudik. Message authentication with one-way hash functions. In *Proceedings of IEEE INFOCOM '92*, pages 2055–2059, May 1992.

[24] S. Wu, F. Wang, B. Vetter, W. Rance Cleaveland II, Y. Jou, F. Gong, and C. Sargor. Intrusion detection for link-state routing protocols. Presentation in the 1997 IEEE Symposium on Security and Privacy. Further information is available at http://shang.csc.ncsu.edu, May 1997.