Enhanced security strategies for MPLS signaling

Francesco Palmieri and Ugo Fiore
Federico II University, Napoli, Italy
Email: {Francesco.Palmieri, Ugo.Fiore}@unina.it

***Abstract* — In an increasingly hostile environment, the need for security in network infrastructure is stronger than ever, especially for Multi-Protocol Label Switching (MPLS), widely used to provide most of the new-generation network infrastructure-level services in the Internet. Unfortunately, the MPLS control plane lacks scalable verification for the authenticity and legitimacy of signaling messages and communication between peer routers is subject to active and passive forgery, hijacking and wiretapping activities. In this paper, we propose a robust framework for MPLS-based network survivability against security threats. The security of MPLS control plane protocols can be greatly enhanced by requiring digital signature of all the signaling messages, in accordance with a common security paradigm valid for all the protocols. Our design goals include integrity safeguarding, protection against replay attacks, and gradual deployment, with routers not supporting authentication breaking the trust chain but operating undisturbed under any other respect.**

***Index Terms* — MPLS, strong authentication, integrity, label distribution, signaling**

I. INTRODUCTION

The concept of security is traditionally connected to the exigency of protecting sensitive data from unauthorized access, but nowadays security is frequently approached from a different perspective. With the growing use of the Internet infrastructure for commercial applications, the demand for evolutionary network services has rapidly increased. Multi-Protocol Label Switching (MPLS) is one of the emerging paradigms in Internet, and seems to be the cornerstone for developing more and more network infrastructure-level services. Almost all the new MPLS-based network services, such as Traffic Engineering, Differentiated Services QoS and layer-2/layer-3 VPN facilities need complex, reliable control and signaling protocols for information exchange between the nodes participating in an MPLS domain. In spite of this, there are many security issues that the current MPLS protocol framework does not address. The lack of strong protection based on cryptographic techniques as part of the original network control and signaling protocol design actually allows malicious users to exploit the network in numerous ways by intercepting basic control information in the MPLS domain and sending fake data to subvert the network behavior. Thus, it is possible that an attacker can wiretap on the transmission link to inject anything he wants to hijack the network routers and then breach the overall infrastructure as desired. The above is an indication of the fact that bringing down a network infrastructure without causing any physical damage to the network entities, is quite conceivable. As a consequence, the need for security in MPLS network infrastructure is stronger than ever. The main requirements are data origin authentication and data integrity for all the transactions involved in the MPLS label exchange process. Furthermore, the importance has also emerged of making sufficiently robust the protocols transporting vital information for the network behavior . Our first goal is to identify the infrastructure-level vulnerabilities of existing MPLS domains and then develop the necessary security enhancements to be deployed for a survivable and secure MPLS framework. Accordingly, we propose a general method, implementable as a common practice or framework, to fight against these attacks based on digital signature: the originator of an MPLS control plane message, such as any label distribution or signaling protocol handler, signs the message it sends, and the corresponding recipients verify the signature so that the authenticity and integrity of the message can be protected. Our assessment shows that the framework is capable of preventing many currently unchecked security threats against the MPLS control plane without unduly overloading network elements. Even if we don't address protection from denial of service threats such as overwhelming the routers by generating spurious excess traffic, our framework may also provide an early warning of the presence of a compromised MPLS core router.

II. RELATED WORK

The idea of signing vital protocol information is not new. Several efforts in this direction have been done for routing protocols. Foremost, of course, there is the design that R. Perlman reported in her book [1] for signing link state information and for distribution of the public keys used in signing. However MPLS security is still an almost totally open issue. Behringer [2] and Senevirathne et al. [3] discuss two approaches to securing MPLS and analyzed the security of the MPLS framework. Senevirathne proposes an encryption approach using a modified version of IPsec working on single MPLS

Based on "Securing the MPLS control plane", by Francesco Palmieri and Ugo Fiore which appeared in the Proceedings of the International Conference on High Performance Computing and Communications (HPCC 2005), Sorrento, Italy, September 2005, LNCS 3726/2005 pp. 511-523, Springer Verlag.

packets that adds significant processing delay in label forwarding nodes and overhead in packet size, and is clearly unacceptable in actual MPLS production networks. Behringer [2] analyzed MPLS security from the VPN service side making the assumption that the core MPLS network is trusted and provided in a secure manner. We make no such assumption in our work. We assume that the MPLS nodes themselves are secure, but the physical links connecting them are not – thus we will protect all the exchange of control plane messages between them using our enhanced security framework. A simple authentication scheme has been proposed for RSVP-TE [20]. However, that scheme does not cover LDP. Recently, Fang [4] suggested a set of common practices structured together in a security framework, where signaling nodes can authenticate each other by means of existing digest-based techniques. The opportunity of using stronger authentication methods is also cited but not analyzed in depth. Here, we are proposing a unified authentication framework based on X.509 certificates together with existing protocol enhancements and a hop-by-hop trust chain, whose scope is the entire MPLS control plane. We are unaware of other research efforts aimed at the same goal.

III. THE MPLS FRAMEWORK: BASIC CONCEPTS

This section briefly introduces some of the basic concepts that will be useful to better explain the MPLS framework, by presenting its architectural building blocks, ideology and the theory behind it.

A. The MPLS paradigm

MPLS is a packet forwarding technique being standardized by IETF [5]. MPLS uses labels to make forwarding decisions at the network node level, in contrast to the traditional destination-based hop-by-hop forwarding in IP networks. In MPLS, the space of all possible forwarding options in a network domain is partitioned into *"Forwarding Equivalence Classes" (FECs)*. For example, all the packets destined for a given egress may belong to the same FEC. The packets are labeled at the ingress depending on the FEC they belong to. Each of the intermediate nodes uses the label of incoming packet to determine its next hop, and also performs "label swapping," i.e., replaces the incoming label with the new outgoing label that identifies the respective FEC for the downstream node. The MPLS encapsulation envelope is depicted in fig. 1 below.

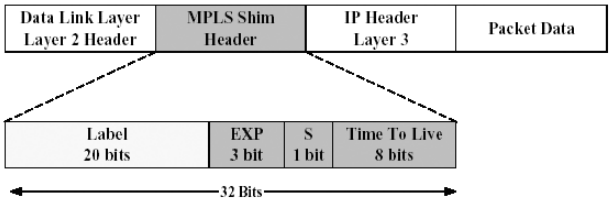

Figure 1. MPLS shim header.

The label swapping maps corresponding to the FEC are established using standard MPLS signaling protocols that will be discussed in the following paragraph. Such a label-based forwarding technique reduces the processing overhead required for routing at the intermediate nodes, thereby improving their packet forwarding performance. Also, the label-merging procedure used by MPLS creates multipoint-to-point packet forwarding trees in contrast to a routing mesh in conventional network based on a similar paradigm such as ATM networks. This reduces considerably the size of forwarding table at the intermediate nodes, thereby improving their scalability.

B. MPLS Signaling

In MPLS, packets are forwarded based on a label rather than the IP address, as in traditional routing. At each hop along the forwarding path, an MPLS node uses the label to make forwarding decisions for a packet. Remember that MPLS is a label switching or swapping scheme. Each MPLS node maintains a table of label information; MPLS nodes use this table to look up the label on a packet coming into the node so they know which label to apply to the packet before forwarding it on to the next MPLS node. Each MPLS node removes the incoming label and appends an outgoing label. Like TCP, MPLS deals with streams of data, and a key aspect of MPLS is that packets with the same label are forwarded the same way. For packets forwarded in a hop-by-hop fashion, for example, this means that all packets with the same label are forwarded to the same next hop. In this way, a *Label Switched Path (LSP)* is established. LSPs are basically a concatenation of one or more label switched hops. They are extended through a network as each MPLS node swaps the incoming label for the outgoing label assigned to the next hop for that data stream. In essence, an LSP is established when each MPLS node along the path between the initial (ingress) MPLS node and the final (egress) MPLS node has a binding between an incoming label and an outgoing label. On the one hand, forwarding with labels requires relatively simple functions – looking up a label in a table, swapping labels, and perhaps checking a *Time-to-Live field (TTL)*. On the other hand, a control component is needed for critical functions, such as creating the bindings between labels and routes, distributing label information to MPLS nodes so they know which label to apply to a given stream of data, and withdrawing labels. MPLS has a clear separation between data forwarding and the control mechanisms used for routing, label management, and other control functions. That is, packets are forwarded based on label switching, regardless of the underlying control mechanism. As a result, a variety of control mechanisms can be used with MPLS; signaling is a key aspect of these control mechanisms. Specifically, MPLS supports a variety of signaling mechanisms for label distribution and LSP set up. To date, the mechanisms the IETF has defined for basic label distribution is the *Label Distribution Protocol (LDP)* [6]. The IETF has also defined enhancements to LDP for *Constraint Route signaling (CR-LDP)* [7] and extensions to basic *RSVP for Traffic Engineering (RSVP-TE)* [8]. Both of these signaling protocols build on existing protocols to allow explicit set-up and traffic engineering of LSPs in an MPLS network.

C. Label distribution: the LDP protocol

The LDP protocol comes into play primarily for best-effort forwarding and is the signaling protocol designed for setting up LSPs on a hop-by-hop basis. LDP encompasses two major functions. First, it is a mechanism by which MPLS nodes can exchange information about label mappings. Second, it defines a set of procedures and messages that allow MPLS nodes to establish LSPs through a network that match hop-by-hop routed paths. It is this second function, more so than the first, that clearly delineates LDP as a signaling protocol. In addition, LDP is also the basis for the modern MPLS pseudo-wire services, providing transparent edge-to-edge transport of Ethernet, ATM, frame relay, and TDM, by encapsulating them into LSP tunnels traversing the MPLS core. Like all modern signaling protocols, LDP is message based. LDP defines a number of messages, including session messages that are used to establish, maintain, and terminate sessions between MPLS nodes. It also uses advertisement messages in order to create, change, and delete label mappings. In addition, it has notification messages to provide advisory and error information. All of these messages are transmitted over TCP connections. The only messages that LDP sends as UDP packets are discovery messages, used by MPLS nodes to announce their presence on the network. A taxonomy of all the LDP message types together with the generic message structure is reported in fig. 2 below.

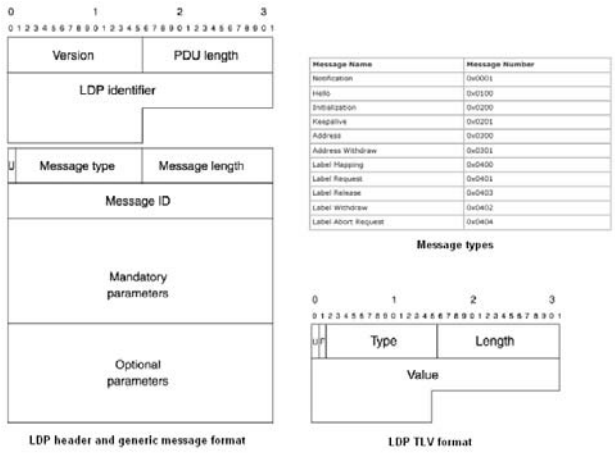

Figure 2. LDP message layout.

An LDP packet always starts with a Protocol Data Unit (PDU) header comprised of the following fields:

- *Version*: Currently, the only version defined is 1.
- *PDU Length*: The length of the PDU, including any data carried in the packet after the PDU header.
- *LDP Identifier*: An LDP identifier is a six-octet string used to identify a particular label space. The first four octets of this LDP identifier are the LSR-ID. The value of the next two octets depends on the label space. If this PDU pertains to global label space, these two octets are both 0. Otherwise, if it is per-interface label space, these two octets are a unique number assigned by the PDU originator.

The LDP PDU header is followed by one or more LDP messages, as shown in the above figure. Here, the *Unknown (U)* bit determines the action to take if an unknown message type is received. The *Message Length* reports the length in bytes of the variable set of fields that follows (*Message ID+Mandatory Parameters+Optional Parameters*). The Message ID is sometimes used to associate some messages with others. For example, a message sent in response to another message uses the Message ID of the message to which it's responding. The Mandatory Parameters and Optional Parameters depend on the types of message sent and are often built with *TLV (Type/Length/Value)* triplets, a common way of encoding arbitrary amounts of data in a packet envelope. The generic TLV format used inside an LDP message is also shown in fig. 2. The *U* bit in the TLV header, if set to 1, means the receiving router should ignore it if it doesn't understand this message. The *Forward (F)* bit is relevant only if the U bit is set to 1 and the LDP message containing the unknown TLV is to be forwarded. Because our model is based on a hop-by-hop authentication, we will assume this bit to be 0. The Type field indicates the type of data carried in the Value portion of the TLV. The Length field indicates the length of the data carried in the Value portion of the TLV. The actual values in the Type, Length, and Value fields depends on the Message Type and what exactly LDP is trying to accomplish with a particular message. As we noted above, LDP establishes LSPs on a hop-by-hop basis. That is, each MPLS node independently selects the next hop for a given stream of data based on the information in its label information table and its routing table. However, the path that an LSP follows can also be explicitly defined based on some special handling requirement or constraint, such as the need for a path with a given QoS or for paths that allow the network operator to balance traffic across a network. Network operators may want to use explicit routing for a variety of reasons. For example, explicit routing can be used for traffic engineering, enabling network operators to select paths for traffic with an eye toward balancing the load on links, routers, and switches across a network. Network operators also may want to apply resource reservations to explicitly routed paths to ensure that a particular traffic type, such as voice, gets the handling it needs as it traverses the network. Specific signaling protocols have been defined to establish explicitly routed LSPs. Let's examine them more closely.

D. Signaling for explicit routing: RSVP-TE and CR-LDP

The IETF has defined two signaling schemes for establishing explicitly routed or constraint-based LSPs in an MPLS environment. One relies on the use of RSVP, while the other approach is based on extensions to LDP. Both schemes allow for the setup of explicitly routed LSPs and for signaling of a set of parameters, such as bandwidth constraints, relating to those LSPs. RSVP is a soft-state protocol with its own protocol type (46), commonly used to reserve resources throughout a network, that needs to periodically refresh its reservations by re-signaling them. Its basic functionalities are related to four macro-activities: path setup, teardown and

maintenance and error signaling. The RSVP packet format is pretty straightforward, as can be seen from fig. 3 below. Every message is composed of a common *header*, followed by one or more *objects* which are TLV-encoded fields having specific purposes. The number of objects in a message depends on exactly what the message is trying to accomplish as defined in the *Message Type* field. The most important message types are Path (0x01), Resv (0x02), PathErr (0x03), ResvErr (0x04), PathTear (0x05), ResvTear (0x06), ResvConf (0x07), ResvTearConf (0x0a) and Hello (0x14).

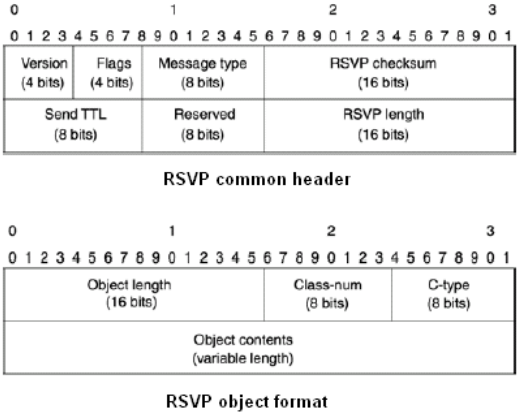

Figure 3. RSVP message layout.

Twenty-three different object classes are defined. Each class has its own C-Type number space. The C-Type numbers are unique within a class. For example, the SESSION class has four defined C-Types: IPv4, IPv6, LSP_TUNNEL_IPv4, and LSP_TUNNEL_IPv6. The numbers assigned to these C-Types are 1, 2, 7, and 8. LABEL_REQUEST has three C-Types defined: Without Label Range, With ATM Label Range, and With Frame Relay Label Range. The numbers assigned to these C-Types are 1, 2, and 3. You can see that a C-Type of 1 is not enough to uniquely identify a message's contents; you need to look at both the class and C-Type numbers.

In [8], the IETF has defined extensions to RSVP that allow it to be used to establish explicitly routed LSPs. Since support for traffic engineering was a key requirement that drove many of the new features in RSVP, this enhanced version of the protocol is referred to as RSVP-TE. It provides procedures for using RSVP messages to allocate and bind labels, to distribute these label bindings, and to establish LSPs. RSVP-TE uses many of the same message types defined for RSVP, but with newly-defined objects. For example, labels are distributed using a special LABEL object in RESV messages. Similarly, labels are bound to specific LSP tunnels through the use of the LABEL_REQUEST object in PATH messages. An EXPLICIT-ROUTE object in PATH messages is used to set up explicit routes, and so on. RSVP-TE encompasses a range of MPLS-related functions, including the capability to establish LSP tunnels with or without QoS reservations, reroute around an established LSP tunnel, and preempt an established LSP tunnel under administrative policy control. Because RSVP was originally designed to signal QoS requirements in an IP network, RSVP-TE is easily used to support QoS-based traffic forwarding. The RSVP-TE PATH message includes a field that encompasses the QoS parameters for bandwidth, burst limits, delay, jitter, etc. for each link along the requested LSP.

The second signaling protocol that the IETF has defined for explicit routing is based on extensions to LDP, ant so inherits all the LDP message structure and behavior. The IETF has defined these extensions in an Internet draft titled “Constraint-Based LSP Setup Using LDP”, and refers to the signaling protocol as CR-LDP (for constraint-based routing). Functionally, CR-LDP is quite similar to RSVP-TE: It allows for the specification of traffic parameters relating to an explicitly routed LSP and supports the preemption of existing LSPs to accommodate the resource requirements of a new LSP. As with RSVP-TE, CR-LDP relies on the definition of new objects that are carried in LDP messages. As in the RSVP-TE PATH message, the CR-LDP LABEL REQUEST message has fields defined for the explicit route and specific bandwidth parameter requests for each link along the requested LSP. It is through signaling mechanisms that establish LSPs that MPLS achieves its ability to bring “connection orientedness” to IP. And because LDP, RSVP-TE, and CR-LDP are message-based signaling mechanisms, it’s easy to see how they are extended to support additional constraints or parameters in establishing LSPs.

IV. THE MOST COMMON VULNERABILITIES AND THREATS

Broadly speaking, threats to MPLS control and signaling protocols come mainly from two sources, external and internal. External threats come from outside intruders who are non-participants in the protocol. Internal threats come from compromised protocol participants such as regular MPLS nodes belonging to the domain. From another point of view the above threats may be originated by passive or active attacks to the MPLS nodes. Passive attacks are those in which the attacker does not actively participate in bringing the network down. An attacker just eavesdrops the network traffic and determines the nodes which are pivotal for the proper operation of the network. He can propagate this information to one or more accomplices who can in turn use it to launch direct active attacks to network functionality. To perform an active attack the attacker must be able to inject arbitrary packets into the network. The goal may be to subvert the network behavior by attracting or redirecting packets destined to other nodes for further analysis or fraudulent handling of the available network resources or just to disable the network or parts of it. A major difference in comparison with passive attacks is that an active attack can sometimes be detected. This makes active attacks a less inviting, but necessary, option for most attackers. In particular, active *Denial of Service (DoS)* attacks can be mounted specifically against the control plane mechanisms the service provider uses to provide various services, or against the general infrastructure of the service provider (e.g. internal MPLS

core routers or shared aspects of provider edge routers). Such DoS attacks can be accomplished through an MPLS signaling storm, resulting in high CPU utilization and possibly leading to control plane resource starvation. Some ability to prevent such control-plane DoS attacks may be provided by rate-limiting signaling and routing messages that can be sent by a peer provider according to a traffic profile and by guarding against malformed packets. Clearly, most of the attacks become easier to realize and hence more likely as the shared infrastructure by means of which MPLS service is provided expands from a single provider to multiple cooperating ones to the global Internet. More specifically, an outside intruder could attack an MPLS domain in various ways:

- *Label Spoofing*: Similar to IP spoofing attacks, where an attacker fakes the source IP address of a packet, it is also theoretically possible to spoof the label of an MPLS packet to force passing of packets through disjoint VPNs, and break the VPN isolation mechanism by inserting packets with labels that belong to LSPs of others' VPNs. There are two places from where the attackers can launch such attacks. The first place is from the outside. Since packets from outside should not have MPLS labels, the MPLS edge routers should drop incoming packets with MPLS labels. The second place is from compromised nodes within the MPLS network.
- *Breaking the neighbor relationship*: This is a typical attack on *availability* in which an intelligent filter placed by an intruder on a communication link between two nodes could modify or change information in the label binding updates (advertisements or notifications) or even intercept traffic belonging to any data session. For example, if LDP *Hello* or *Keepalive* message (UDP port 646) are filtered out, then the neighbor relationship is not established or terminated.
- *Replay attack*: An intruder could passively collect label advertisement information. Later, the intruder could retransmit "obsolete" or fake information messages attacking the *integrity* of the MPLS domain. If obsolete information is accepted and disseminated, a normal node could make incorrect label binding decisions and send wrong information to its LDP neighbors, and consequently partially or totally disrupt the overall network behavior.
- *Masquerading*: During the neighbor acquisition process, an outside intruder could masquerade as a nonexistent or existing node by attaching itself to communication link and illegally joining in the LDP protocol domain by compromising the neighborship negotiation system. The threat of masquerading, which is typically an attack on *authenticity*, is almost the same as that of a compromised internal node.
- *LSP spoofing or DoS attacks*: An unauthorized network element can establish an LDP session by sending LDP *Hello* and *LDP* Init messages, leading to the potential setup of an LSP, as well as accompanying LDP state table consumption. Similarly, also RSVP can be exploited by setting up a very large number of unauthorized LSPs. That's worse, even without successfully establishing LSPs, an attacker can launch a DoS attack in the form of a storm of RSVP *Hello* or *Path* or LDP *Hello* messages and/or LDP *TCP Syn* messages, leading to high CPU utilization on the target router. It has been demonstrated that unprotected MPLS routers can be effectively disabled by both types of DoS attacks. These attacks may even be combined, by using the unauthorized LSPs to transport additional messages across routers where they might otherwise be filtered out. Signaling attacks can be launched against routers adjacent to the attacker, or against non-adjacent routers within the MPLS domain, if there is no effective mechanism to filter them out.
- *Passive Listening and traffic analysis*: The intruder could passively gather exposed label binding or LSP setup information, or gain router access information such as remote terminal or SNMP access credentials. Such an attack, on information *confidentiality*, cannot directly affect the operation of the MPLS domain, but the obtained information can be used successively for hostile activity. Sensitive router access, label bindings or LSP status information should be protected and their confidentiality should be the responsibility of the remote access, label distribution or signaling protocol.

If an internal MPLS node has been subverted, all information inside the node is exposed and at risk. The label information base (LIB) or the label forwarding information base (LFIB), can be directly manipulated via system commands or control interfaces to disrupt the network layer decisions and cause incorrect forwarding behavior in the network. By seizing the control of an MPLS node, an intruder could add a route entry to FIB or a label binding entry into LFIB which will divert data traffic to a particular destination. An intruder could also randomly modify the above information to make the router advertise wrong labels or tear down existing LSPs or in the worst case creating some for of traffic blackholing (dropping because of a wrong label/prefix mapping has been advertised), which is a kind of DoS attack. Clearly such attacks may have serious consequences on the network infrastructure and on the end-to-end communications. The compromised node problem has not received much attention to date, for several reasons. First, there are usually much fewer MPLS routers in a network administrative domain than hosts, and they are usually under tight control and monitoring of network administrators. Therefore, the MPLS nodes have a larger defense perimeter than the ordinary hosts. In other words, the trust of them from humans is high. Second, control plane signaling is distributed and cooperative in nature (i.e. routers in an MPLS domain must coordinate their actions and cooperate to meet their protocol requirement) and thus there is a tradition of trust in all these protocols.

MPLS networks must provide at least the same level of protection against all these forms of attack. There are two fundamental ways to protect the network: firstly,

hardening protocols that could be abused, secondly making the network itself as inaccessible as possible. This can be achieved by a combination of packet filtering and address hiding. In more detail, the MPLS network's internal structure information which may be useful to attackers must be totally hidden from outsiders. For example, DoS attacks against core routers will be hard if attackers do not know the targeted router's address. Thus, MPLS networks must not reveal the internal structure to customers. However, the MPLS core can be compromised by attacking the exposed edge routers. Since MPLS internal structure can be easily hidden from the outside, attackers do not know the address of any router in the core thus cannot attack them directly. To protect the edge routers whose addresses are known to the outside, we can use Access Control List (ACL) to accept any communication only from customers' routers. On the other side hardening MPLS control protocols is a more complex matter and to accomplish such a task several extensions have to be done to their protocol structure, by introducing strong authentication, no-replay and cryptography facilities, as will be detailed in the following section.

V. HARDENING MPLS SIGNALING PROTOCOLS

Generally speaking, making MPLS signaling protocol sufficiently secure requires that important label distribution and LSP setup information will be strongly authenticated between neighboring routers, since almost all the actually known attacks take advantage of the lack of authentication, integrity and confidentiality services. Here are the definitions of these services in the context of secure communication protocols:

- Authentication services are primarily concerned in providing assurances about the identity of an entity. In our context, when a router sends out a message, the identity of the originator of the information should be strongly validated.
- Integrity services ensure that the data being transmitted is consistent with the data being received.
- No-repudiation services provide irrefutable evidence that a certain event took place.
- Confidentiality service provides privacy of the protocol message, which uses encryption to prevent others from knowing what the message contains.

Accordingly, we need to introduce some new security features into the above protocols. These security features can be broadly divided into two families, namely encryption schemes and authentication schemes. Encryption schemes provide data confidentiality, which keeps the content of information only available to those who are authorized to have it [9]. Authentication schemes assure no-repudiation and data integrity, which protects messages against forgery or unauthorized alteration. Note that confidentiality does not imply data integrity. In this work we will not address confidentiality, since relevant information still travels in the clear. The essential security mechanism that have to be implemented into the MPLS control protocols are described below.

A. Introducing sequence information

Sequence information, which can be sequence number or timestamp, has to be added to each update to prevent the replay of old information. The message sender generates packets with monotonically increasing sequence numbers. In turn, the receiver drops messages that have a lower sequence number than the previously received packet from the same source. Duplicate messages with old sequence information will be dropped. The primary challenge posed by this requirement on long-lived MPLS control information is how to prevent sequence information from wrapping around. A sequence information must be valid for the life of a given MPLS router id. The primary advantage of sequence numbers compared to timestamps is their significantly longer life, because they will only increase when messages are sent. A sequence number can be relatively small and still provide reasonable assurance of not cycling. If timestamps were to be used, one would have to make sure that clocks were properly synchronized, e.g. via NTP. The main benefit of a timestamp would be the ease of administration provided by the well defined external reference for use in resetting lost state. The life of even a small timestamp, while not as dramatic as for sequence numbers, is still significant; assuming a granularity as small as one second, a four octet timestamp still has a life of over 130 years, much longer than any reasonable lifetime of a router. Anyway, in order to be able to check the validity of any type of sequence number, both parties must handshake and agree on some common initial information. Thus, whenever a router first establishes an LDP or RSVP relationship with a neighbor, e.g. at reboot, it must generate an initial sequence number (ISN) for that neighbor, and communicate it to the neighbor, for example transported in an LDP or RSVP HELLO message. The sequence number can then be incremented as usual, and it can be transmitted to the neighbor with each message. Another approach would be that the ingress LSR were to associate a nonce value (analogously to what is described in [10], [11], [12]) to each LDP request. That nonce would be then transmitted to the LDP neighbor, who would include it, unaltered, in the reply. Such a mechanism provides a way of matching requests and replies but offers a relatively weak protection against replay attacks, since an attacker could reuse a previously captured packet to force a long-lived LSP to close. If the nonce were to be associated to a single LSP at the time of the initial setup, with any operation involving that LSP deemed valid only if it carried that particular nonce, the nonce in the first LABEL REQUEST message should be trusted, still leaving room for attacks. We believe that a solution to this problem would be the use of symmetric encryption techniques and a trusted mutual key exchange, but this is out of the scope of this paper. We will investigate these issues in a future work.

B. Digitally Signing Update information

To ensure the authenticity and integrity of the information exchanged between different routers, the originating router can digitally sign each update it

generates. In addition, to allow receiving routers to validate the signature, a strong verification against a known key associated to the originating router must be performed on each update. These signatures can be used to validate each vital control information flowing between the network elements in an MPLS domain such as a candidate label or path to a destination change before that label or path is selected for use. Modern standardized digital signature schemas rely on Public Key Cryptography, that is the popular synonym for asymmetric cryptography, i.e. encryption using two matched keys in which a pair of users does not share a single private key. Instead, users have their own key pairs and each key pair consists of a matched private and public key. Moreover, the process of computing one key from another is not possible. Asymmetric cryptography can perform digital signature, thus offering data origin authentication and data integrity, secure transmission or exchange of secret keys and data privacy (encryption and decryption). To accomplish the above functionalities on our protocol messages, each message has to be signed using a private key *d*, to produce an authenticated message *(m;t)*, where *t* is the tag on *m*, and this authenticated message can be verified by the sender's public key *e*. In relation to digital signature schemes, the tag generated with this method can be seen as a message signature and is known as a *Message Authentication Code (MAC)*. A MAC is used by the receiver to verify that the message originated by the presumed sender and has not been tampered with. Hash functions can be used for providing data integrity in digital signature schemes, where the message is typically hashed first, then the hash-value is signed in place of the original message. A hash function *h* has the following properties:

- given an input *x* of arbitrary finite bit length, an output *h(x)* of fixed bit length will be produced,
- given description of h and an input x, it is easy and efficient to compute h(x),
- it is computationally infeasible to find any input that hashes to a specific output,
- given an input *x*, it is computationally infeasible to find an $y \neq x$, where $h(x) = h(y)$ [13].

Examples of public key cryptography standards are DSS (Digital Signature Standard) and RSA (Rivest, Shamir, and Adleman). Of course, any encryption-based technique adds computational burden, end hence execution time, to the process that has to be made more secure, but a correct choice of algorithms and key sizes can make the above tradeoffs more than acceptable.

C. The trust model: key management and distribution

The trust model among the MPLS network elements, the involved providers, and the other parts of the network is a key element in determining the applicability of asymmetric cryptographic techniques for any specific security framework implementation. The only disadvantage of asymmetric cryptography is the fact that the public keys must somehow be introduced and trusted by all the participating parties. In other words, a trustable binding it is needed between the user's identity and his public key. The best solution is given by the third trusted party (TTP) model. In this model a third party – commonly agreed to be trusted by all interested parties – authenticates users to each other. The X.509 ITU-T Recommendation defines a feasible (and widely adopted) TTP model/framework to provide and support data origin authentication and peer entity authentication services, including formats for public-key certificates. The X.509 certificates [14] are cryptographic structures used for binding a user's identity to its public key. This binding is sealed by means of a digital signature performed by the Third party Trust Authority (usually a certification authority or CA) that issues, trusts and digitally signs a digital certificate. Hence, an X.509 CA has the roles of issuing, distributing and revoking, whenever necessary, its public-key certificates. Although certificates tend to be long-lived, they need to be revoked if the authenticity or authority they represent is no longer valid. In the standard approach, the authority who issued certificates generates and signs a certificate revocation list (CRL) that contains the serial numbers of revoked certificates. Regarding issuance of certificates, the CA can decide to delegate the task of user identification to an optional separate entity, a registration authority (RA), that has the responsibility for recording or verifying some or all of the information needed by a CA to issue certificates and CRLs and to perform other certificate management functions. The distribution of certificates can be achieved in at least three ways: the users can directly share their certificates with each other, or the certificates can be distributed via HTTP protocol or, better, via LDAP [15] directory service. In what concerns revocation of certificates, the CAs have the duty to publish at certain time intervals the CRLs, the black lists on which the revoked certificates are enlisted, together with the date and reason of revocation. Any party that wishes to validate a certificate may download CRLs from publicly available sources to verify that a given certificate is not revoked. CRLs are updated periodically, but since they are essentially checked off-line some synchronization problem may occur. In contrast, a more timely approach for validating certificate status is the Online Certificate Status Protocol (OCSP) [11] where online responders answer certificate status queries in real time. To ensure sufficient scalability to the above model through clever engineering of its complex components the concept of public key infrastructure (PKI) has been introduced. A PKI is usually defined as a set of cooperating CAs that issue, distribute and revoke certificates according through a hierarchical trusting policy. Typically, a per PKI-authority is established to approve or coordinate certification practice statements, which are security policies under which the components of a PKI operate. This organization reduces the problem of verifying everyone's public key to verifying just one (or a few) public keys. In a typical PKI, an authorized third party issues certificates according to a well defined hierarchal structure. A certificate binds an entity with its public key and is signed with the authorized third party's own

private key. The recipient of a certificate use the authorized third party's public key to authenticate it.

Using digital signature for trusting MPLS control information implicitly assumes that there should exist a Trust Authority or PKI to which the MPLS domain refers. The Trust Authority has its own private/public key pair and all the bindings between the LSRs and their public keys. Each router will be configured with its own pair of private/public key and the public key of the Trust Authority that they use to verify the trustiness of all the information obtained from the authority.

D. The overall security scheme

We propose a digital signature scheme relying on Public Key Cryptography that takes its foundations on the work done by Murphy and Badger in [16] to secure the OSPF protocol, but is significantly modified and extended both in the basic paradigm and in key management/distribution to cope with the very different features and security requirements of the MPLS control protocols. The basic idea of this scheme is to add digital signature to each protocol message, and use an hash function such as message digest (like keyed MD5) to protect all exchanged message data. The originator will sign each message before sending it and the signature will stay within the message body during all the message life. This will protect the message integrity and provide authentication for all the transported data, ensuring that the data really does come from their legitimate source and has not been modified in transit. Obviously, the modification of any of the fields belonging to the protocol messages or signature information will result in a failure in signature verification that implies, on router supporting this enhanced security feature, message discard. Furthermore, in the case where incorrect data is distributed by a faulty router, the signature facility provides a way to trace the problem to its source. MPLS signaling protocols have to be extended to support on each of their vital messages a new optional object containing the above signature and all the useful information on its size, digital signing and hashing algorithm. Eventually this object can be also used for transporting the sequence number information whose necessity has been describer in the previous section.

Routers providing this digital signature facility must behave as normal MPLS routers for all the protocol functionalities and can be straightforwardly mixed with router not implementing this feature. All the signature and integrity-enforcing mechanism has to be implemented, according to the same paradigm, for each signaling protocol (namely LDP and RSVP) as an optional attribute that will be transported in any message, used if recognized and otherwise silently discarded.

When a router receives a signed message, it at first verifies the signature using the known public key of the originator identified by the IP address associated to its LSR-id. If the signature verifies, the router accepts the message for usual processing, otherwise if the router does not knows the public key of the originator, it asks the trust authority for it, performing a standard LDAP query, and if it is available obtains and verifies the certification by using the Trust Authority's known public key and then stores the public key in its memory for further usage. For faster retrieval, public keys should be stored into the router memory by using a hash table indexed by originator LSR identifier where each public key will be kept until it expires or is revoked. If the originator public key does not exists the signed message is immediately discarded, otherwise when the known key is no more valid, to handle change or revocation/reissue cases, a new public key is requested via LDAP and, if available, stored into the router's memory and a re-verification, against the new public key take place. The pseudo-code skeletal representation of the proposed security scheme, as it should be implemented within the MPLS signaling protocols, is reported in fig 4 below.

```
for each message m received from LSR r do
   if Public-Key-Hash-Cache(r) is not null then
   /* Check the expiration info available with the certificate */
      if not expired (Public-Key-Hash-Cache(r)) then
         if Check-Is-Valid(Public-Key-Hash-Cache(r)) then
            if verify-mesg-auth-code(m,
                  Public-Key-Hash-Cache(r)) then Accept(m)
            else Reject(m); goto continue
            end
         end
      end
/* Fetch the certificate for the requester LSR-ID via LDAP */
      temporary-cert = LDAP-retrieve-certificate(r)
      if temporary-cert is not null then
         if verify-mesg-auth-code(m, temporary-cert) then
            Accept(m)
            Public-Key-Hash-Cache(r ) = temporary-cert
            goto continue
         end
      end
   Reject(m)
continue:
end
```

Figure 4. RSVP message layout.

The above digital signature scheme can prevent external attacks to signaling protocols. Since external attackers can not generate correct signature for MPLS control messages, if they intercept and modify them or inject some malicious information into the protocol, they can be easily detected. There are still a few vulnerabilities with this security paradigm, coming from internal attackers. A subverted router is still able to forge protocol information, delete label mapping updates, and disclose LSP status information.

VI. SECURING THE LABEL DISTRIBUTION PROTOCOL

LDP security is still a largely open issue since until now the LDP protocol provides no scalable standard-based mechanism for protecting the authentication, privacy and integrity of label distribution messages. LDP can actually use the TCP MD5 Signature Option to provide for the authenticity and integrity of its TCP session messages but MD5 authentication, as asserted by [17] is now considered to be too weak for this

application. Recent research has in fact shown the MD5 hash algorithm to be vulnerable to collision attacks [18]. However the security requirements of label distribution protocols are essentially identical to those of the protocols which distribute routing information. In fact, with the LDP protocol, each router processes the information received from its neighbors and sends back the aggregate information. The result is that it is very hard to validate the received information since the real originator of the information is obscured. Thus we need to set up a hop-by-hop trust chain in which message authentication and integrity checking has to be performed independently at each traversed node, from origin to destination. This implies a neighborship-based checking with message resigning and re-verification on each transaction. LDP has two different types of neighbors: directly connected neighbors - having a Layer 2 direct connection, hence a single IP hop, between them and non-directly connected neighbors - who are several IP hops away and connected to each other by MPLS traffic engineering tunnels that have LDP enabled on them. Signature verification, performed on each message exchanged on directly and non-directly connected neighbors ensures that the message data cannot be altered in all its fields, including the sequence, (integrity enforcing) and has been signed by the neighbor LSR owning the correct private key (origin authentication). The assurance of source authenticity and integrity relies on two things: first, that the private key is known only to one router, and second that the public key is reliably known, through the trusted authority, to belong to that one router. The direct consequence is a great level of trust in the whole MPLS domain to the prejudice of a slight overhead, if related to message relative frequency, in message processing. In more detail, our enhanced security schema for LDP is based on extending the protocol itself to transport on each of its messages all the security information needed for the strong authentication and integrity requirements presented in the previous sections. Accordingly we propose the introduction of a new optional TLV attribute, named *LDP-Security* (with section title "LDP security extensions") identified by the new type 0x8701 and structured as in fig 5 below:

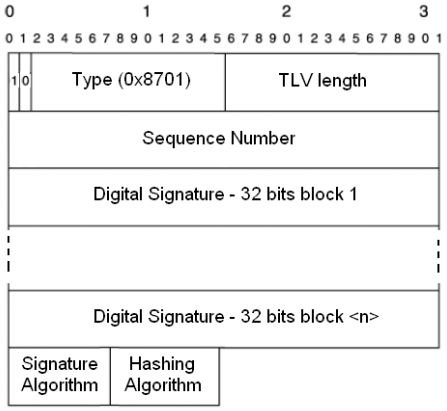

Figure 5. The LDP Security TLV

The encoding scheme is very simple and general and no alignment requirement for the first octet of a TLV. In principle, everything appearing in an LDP PDU could be encoded as a TLV. Our new LDP TLV is encoded, according to the standard LDP scheme, as a 2 octet field that uses 14 bits to specify a Type and 2 bits to specify behavior when an LSR doesn't recognize the Type, followed by a 2 octet Length Field, and by a variable length Value field. The Value field itself may contain TLV encodings. That is, TLVs may be nested. Furthermore, since the value corresponding to our new TLV type has been chosen greater than 0x8000 (the high order "U" bit is set), according to the LDP specification, upon reception of this TLV in the optional parameters section of an LDP message, if it is not recognized by the accepting LDP neighbor as a known LDP type, is silently ignored and the rest of the message is processed as if the unknown TLV did not exist, without any error notification returned to the originator. This allow our scheme to guarantee the full functionality of the label distribution mechanisms also when routers supporting this new TLV to enforce LDP security are mixed with router not implementing any additional security feature. The first 32 bits field contains the message sequence number, calculated as explained in the previous section. The variable length signature follows, built on 32-bit words, followed by two byte codes, respectively identifying the signature algorithm (0x01 for RSA and 0x02 for DSA) and the hashing algorithm (0x01 for MD5 and 0x02 for SHA). This provides great flexibility and extensibility if other asymmetric cryptography or hashing algorithms become available.

Such a TLV can be transported, and hence enhanced security can be provided, on almost all the LDP protocol messages, namely Hello (0x0100), Initialization (0x0200), KeepAlive (0x0201), Address (0x0300), Address Withdraw (0x0301), Label Mapping (0x0400), Label Request (0x0401), Label Withdraw (0x0402), Label Release (0x0403) and Label Abort Request (0x0404), except for the Notification (0x0001) message, whose contents are however only informational, which doesn't allow the transport of optional parameters. This implies that the above security mechanism for LDP will most likely be applicable to the PWE3 [19] control protocol as well, straightforwardly allowing enhanced security for MPLS pseudo-wire services based on it.

Label distribution requires privacy also to address the threat of label spoofing. However, that privacy alone would not protect against label spoofing attacks since data packets carry labels in the clear. Furthermore, label spoofing attacks can be made without knowledge of the FEC bound to a label. To avoid label spoofing attacks, it is necessary to ensure that labeled data packets are labeled by trusted, hence authenticated, LSRs and that the labels placed on the packets are properly learned by the labeling LSRs only if the sending LSRs are trusted. Our hop-by-hop trusting model, providing implicit authentication and checking at each hop and hence natively avoids any label spoofing activity.

VII. Securing The RSVP-TE protocol

To ensure the integrity and authentication of its control messages, RSVP requires the ability to protect its them against corruption and spoofing. Corrupted or spoofed RSVP reservation requests could lead to theft of service by unauthorized parties or to denial of service caused by locking up network resources. RSVP protects against such attacks with a hop-by-hop authentication mechanism using an encrypted hash function [20]. The mechanism, providing protection against forgery or message modification, is supported by the so-called INTEGRITY objects that may appear in any RSVP message. The INTEGRITY object of each RSVP message is tagged with a one-time-use sequence number, allowing the message receiver to identify playbacks and hence to thwart replay attacks. However the proposed mechanism does not afford confidentiality, since messages stay in the clear. Furthermore, RSVP INTEGRITY objects use a keyed cryptographic digest technique, which assumes that RSVP neighbors share a common secret, that is a highly deprecated practice. Reliable and survivable use of an RSVP hop-by-hop authentication mechanism in an MPLS domain requires the availability of a stronger security model based on asymmetric cryptography, providing adequate key management and distribution infrastructure for participating routers. To implement such a strong authentication and integrity mechanism, based on the already explained neighbor based trust chain model implemented by hop-by-hop message signature and verification, as it has been done for LDP, we introduced a new optional RSVP object named STRONG-INTEGRITY defined by a Class value of 0x84 and a C-Type of 1 that can be transported in any RSVP message to be secured, namely Path (0x01), Resv (0x02), PathErr (0x03), ResvErr (0x04), PathTear (0x05), ResvTear (0x06), ResvConf (0x07), ResvTearConf (0x0a) and Hello (0x14). Each sender supporting our enhanced security paradigm must ensure that all RSVP messages sent include a STRONG-INTEGRITY object, with the signature generated using its private key. Receivers, if supporting the above feature, have to verify whether RSVP messages, contain the new object and in this case perform signature verification according to the scheme presented in the previous sections. The Class value has been properly chosen equal to that of the already implemented INTEGRITY class, but with the high order bit set, since according to the RSVP specification when a node receives an object with an unknown class, if the high order bit in the class-num is set the node should ignore the object neither forwarding it nor sending an error message. This allows employing such new objects within an MPLS domain mixed with nodes using implementations that do not recognize them. The object structure is very similar to that used for the new "LDP Security" TLV introduced in the previous section, as it can be seen in fig 6 below. The first 32 bits field contains the usual RSVP object header, and more precisely the 16 bits object length and the two Class-num and C-type octets. Next, we find the 32 bit message sequence number, calculated as explained in the previous section. The variable length signature follows, built on 32-bit words, followed by two byte codes, respectively identifying the signature algorithm (0x01 for RSA and 0x02 for DSA) and the hashing algorithm (0x01 for MD5 and 0x02 for SHA).

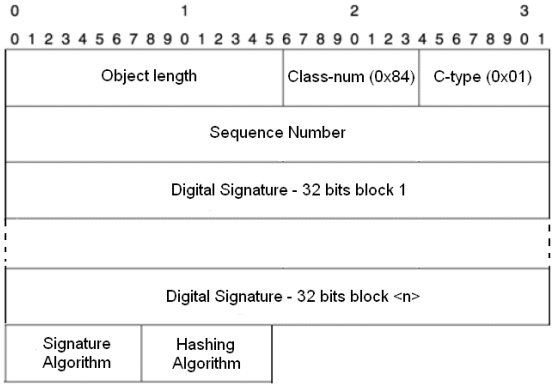

Figure 6. The RSVP STRONG-INTEGRITY object.

VIII. Implementation

To prove the concept of using digital signatures to authenticate control plane messages, we set up a minimal MPLS network using five Intel-based machines with old 800MhZ CPUs (which is a common speed for actual MPLS routers), connected via fast-ethernet interfaces, as shown in fig. 7 below, equipped with the Linux operating system, the mpls-linux v1.935 implementation [21], and, in particular, the ldp-portable v0.800 package, on which most of the implementation efforts take place.

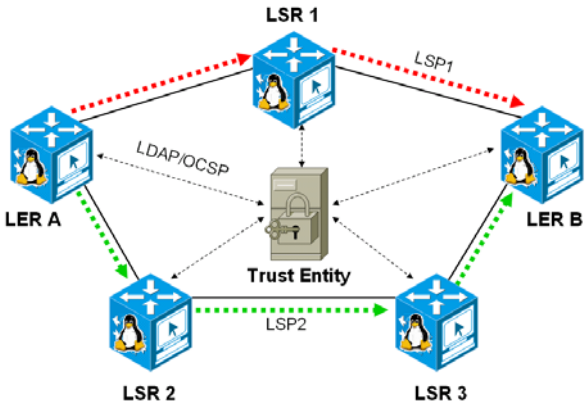

Figure 7. The simple testbed network.

The choice for Linux had several reasons. First, the kernel source code was available, so that we could add or modify the protocol modules needed to implement the MPLS functionalities. Then, the analyzing capabilities are perfect, because the whole open source system is transparent for inspection. Also, debugging and logging in all interesting modules is possible. We only implemented the modification described in the previous sections on the LDP protocol. The GNU Zebra 0.94 [24] routing software provided the necessary underlying routing functions through the supported OSPF implementation. We set up two LSPs from LER A to

LER B, LSP1 and LSP2, passing respectively through LSR1 and LSR2-LSR3 and we sent packets from LER A to LER B by using both the LSPs, checking all the LDP traffic between the nodes. Although this implementation it is only a *"proof of concept"*, it wants to emulate an active MPLS network in which hop-by-hop authentication takes place. For the LDAP and OCSP protocols, OpenLDAP v2.2.15 [22] and OpenSSL v0.9.7e [23] were used. Since the OCSP protocol is faster than LDAP we used it to quickly check if the certificate copy held in a router's cache is still valid. The local CA used for certificate distribution in the MPLS domain was set up based on OpenCA 0.9.2.1 [25], with the OCSP responder daemon 1.0.2, on a Linux server. The certificate cache in each router was implemented using a hash table addressed by the 32-bit LSR-id IP address and containing the corresponding public key certificates and their expiration date. A similar hash table stores the CRLs as they are periodically retrieved from the CAs. For the sake of simplicity, the Linux routers were synchronized via NTP and the sequence number has been implemented as a 32-bit timestamp directly got from the Unix *time()* function. Our simple testbed demonstrated the correct operation of the proposed security framework that does not affect in any way the basic LDP functionalities of the MPLS nodes. The correct behavior of the MPLS LDP messages has been also checked with monitoring/sniffing tools like *TcpDump* and *Ethereal*, and the traffic observations confirmed the impossibility for an internal or external attacker to eavesdrop, intercept, or modify these messages or even inject harmful messages into routing infrastructure. Finally, the LSR 2 Linux-based router has been replaced with a Cisco 3640 router, to test the backward compatibility feature through which routers with or without the above enhanced LDP security support can be mixed in the same MPLS domain. We observed that the LSP2 label switched path was successfully established, obviously without security in the last two hops in the path. The above evidence is sufficient to prove the interoperability and backward compatibility of our schema.

IX. Performance considerations and evaluation

Neither public key cryptography nor access to public key infrastructure comes for free. However, when designing and analyzing the impact of these technologies on large distributed networking systems, it's easy to overlook these implementation details, and the performance impact they can have on the overall protocol behavior. In our performance analysis we used a straightforward set of metrics to unambiguously evaluate the performance impact of the proposed security framework in terms of time and space. For time, we measured the number of cryptographic operations involved, the resulting CPU cycles elapsed and the LDP message propagation time: the time an LSR takes to propagate an LDP *Label Request* message received from a neighbor to another. For space, we measured both the message size and the storage cost in memory of all the used data structures. We also studied the efficacy of strategies for caching some certificates and CRLs.

A. Computational requirements

We first looked at the performance impact of signatures and verifications, then examined the certificate retrieval and validation cost on top of that. All the results, reported in table 1, were obtained by adding monitoring code to the various processing modules used in our security framework. One concern about the use of digital signatures to protect the integrity of control plane messages is that signature producing and verification will clearly affect the performance of the involved routers. The time it takes to produce and verify a signature can vary widely, depending on algorithm, software implementation, key length, and platform. From twenty sample functional tests ran on our testbed, using RSA and DSA algorithms with 1024-bit keys and respectively 128 and 40 bytes signature length, we observed an average verification time lower than 2ms for RSA and slightly greater that 24 ms for DSA. The Signature operation, on the opposite, was faster for DSA (16.5 ms) and significantly slower for RSA (37 ms). Signatures are twice the length of the modulus in DSA, totaling 320 bits. By using a secure storage area, the signer can precompute the message-independent values for future DSA signatures. This technique dramatically speeds up signing time (9.5ms). Anyway, this result clearly implies that the verification overheads are a minor factor compared to signing operations. But, because our model entails hop-by-hop authentication, the number of signature operations will equal the number of verifications and thus in average, considering the whole protocol transactions, both the algorithms exhibit almost the same performance. Consequently, in a real implementation there would be no significant advantage in choosing a scheme (such as RSA) where verification is much simpler to compute than signature. However, since the number of the protocol messages that must be signed (and hence the number of verifications), is very small compared to the overall traffic flowing through the MPLS domain, because signatures are produced at relatively infrequent intervals (baring changes in the MPLS network), and since each signature can be computed at any time during the interval, we do not believe that signing will significantly damper performance. We estimated, however, that cryptography alone increases the cost of processing a *Label Request* message by 170% to 220%, depending on the algorithms and parameters used. It should be taken into account, however, that the topology of our test network is quite simple. Moreover, for each signature to be verified, the verifier needs to validate the certificate of the alleged signer. This usually implies a certificate validation against a CRL or, better, online against a CA through the OCSP protocol. To decrease the number of CRL retrieval and certificate verification operations, one could cache some of the certificates or the entire CRLs in memory. Therefore, memory cost becomes another issue. For certificate retrieval and validation we measured ad compared the average LDAP and OCSP transaction (query/response) times, reported in table 1 below.

TABLE I.
PERFORMANCE MEASUREMENTS RESULTS

Algorithm	RSA	DSA
Verify Time	1,8 ms	24,5 ms
Sign Time	37 ms	16,5 ms
Signature size	128 bits	40 bits
Key size	1024 bits	1024 bits

Protocol	LDAP	OCSP
Network latency	0.4 ms	0.7 s
Connection latency	2.9 ms	-
Processing latency	5 ms	-
Total	8.3 ms	0.7 s

Out of the total LDAP measured response latency of 8.3 ms, about 5 ms come from the processing latency, 36% of which is contributed by back-end processing (entry retrieval from the database), and 64% by front-end processing from the client and server side (building the search filter, matching it against an entry, ASN.1 encoding/decoding of the query/result entry, sending the search result and status). The remaining 3 ms are due to connect and network latency. An efficient local caching strategy such that explained in the previous section can greatly reduce (or nearly zero) the retrieval time by allowing frequently used certificates to be stored in memory. On the other side, certificate validation is a more complex issue. The OCSP transaction time measured for one round is about 0.4–0.7s, the majority of which is from network latency. In fact, in a typical PKI, one or more OCSP responders connect to a certificate database operated by local CAs to provide the status information of the certificates issued by local CAs. Optionally, the responders can set up SSL connections to enhance privacy for the client. The resulting OCSP response is a signed data structure that contains the real-time status of a requested certificate. This introduces latencies, from setting up an SSL connection, from network delays, from real-time signing, and from signature verification. Consequently, checking certificate status online by using OCSP, while providing always fresh information, is intolerably expensive and should be avoided whenever possible. Furthermore, sending sequential OCSP requests is an especially bad idea and to achieve some performance improvements we can think to collect multiple certificates to be validated and send OCSP requests in parallel to multiple OCSP responders throughout the network. A proxy, such as the Certificate Arbitrator Module (CAM) [26] can do the work. Caching the revocation lists, at the expense of more memory on the LSRs will clearly be better. Since each signature verification requires an up-to-date copy of the CRL from the relevant CA, the LSR pays the price of extemporarily fetching and validating fresh CRLs before verifying signatures only if some cached entries are missing or expired. Simply stated, the verifier downloads CRLs periodically, checks certificate status with these local copies, and (when the local copies expire) get fresh CRLs from the appropriate repositories via the LDAP protocol. To evaluate the cost of fetching CRLs, we forced the expiration of a certain fraction of the CRLs stored in the local cache of LSR nodes, and then measured the average retrieval time. In our experiments we evaluated that it costs 0.3–0.7 seconds on average. If we assume that in a typical real-world PKI, CRLs are valid for at least 2 hours, we can see how the overall performance will be greatly improved by the above CRL caching policy. This can be easily observed in Fig. 8 below, where the message propagation times are compared for both the abovementioned validation strategies.

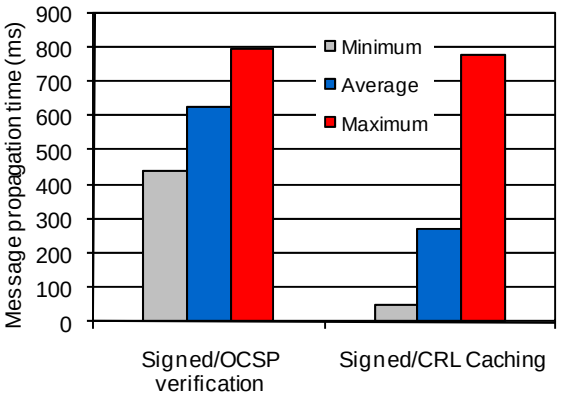

Figure 8. Verification strategy comparison.

Finally, it should be noted the relatively low frequency of LDP messages with respect to payload-carrying MPLS traffic can dilute the overall effect on performance. Several methods could be used to reduce the impact of the required verifications. In all of them, the verification effort could be offloaded from the processor performing the normal router functions. One offload method would be to use PCMCIA or better FPGA cards. These cards have an added benefit in that they provide increased security for the crucial private key. Another offload method would be to employ a separate processor in a multi-processor router architecture. Multi-processor architectures are now available among many router vendors, as mechanisms to separate routing computations from forwarding functions.

B. Storage requirements

The first and simplest factor affecting memory cost is signature length. Clearly, a DSA signature is much shorter than a RSA signature (e.g., 40 bytes vs. 128 bytes) and this directly influences the message size. Anyway, since messages are not usually buffered or stored for a significant time, the above consideration does not perceivably affect the overall memory usage on LSR nodes. The storage requirement for locally caching the public keys is moderate, i.e., one key per LDP neighbor router, plus the keys for any CA involved. Furthermore, the use of a cache will ensure that only the keys pertaining to frequent talkers are kept in memory. In a production environment, the cache size will be a tradeoff between performance considerations and the overall memory requirements of the LSR processes. Note that our algorithm (see fig. 4) only allows a cache entry to be replaced after positive verification. A flooding attack, where the router is swamped with a large number of forged messages with the sole intent of consuming resources, will not have an impact on memory. However,

such an attack will affect the router CPU. Caching CRLs may be, on the other side, significantly more memory-intensive, since real world PKIs often provide huge CRL structures. To handle the above tradeoff, some mixed caching and on-line verification (OCSP) strategy may be introduced, by caching only the CRL entries associated to known LSR certificates and using OCSP to verify new nodes that were not known in the MPLS domain at the periodical CRL retrieval time.

X. CONCLUSIONS

In this paper, we propose a robust framework for MPLS network survivability against most of the common security threats, based on modern standardized digital signature technology, that in our objective should greatly help the network community in making the MPLS control and signaling protocol more secure. We designed and developed a cooperative security model, easily applicable in all the control plane protocols commonly used in the MPLS domain, based on a hop-by-hop trust chain in which message strong authentication and integrity checking has to be performed independently at each traversed node, from origin to destination. This implies a neighborship-based checking with message resigning and re-verification on each transaction that can be apparently seen as a drawback since public key cryptography is very expensive and could slow performance of the MPLS nodes, which should be as fast as possible. The relative frequency of the above control plane messages in a real MPLS-based network, together with the increasing performance of the modern MPLS nodes make the above consideration not a practical problem. On the other side, the introduction of the above mechanisms will require the availability of a complex CA or PKI-based key management and distribution infrastructure for routers participating in a trusted domain. Anyway, our proposed cryptographic system for the protection of MPLS control plane infrastructure proved to be successful in its primary task – ensuring enhanced survivability against malicious service disruption by an intruder with exterior or interior access to an MPLS network.

REFERENCES

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

[2] M. Behringer, *Analysis of the Security of the MPLS Architecture*, IETF Draft <draft-behringer-mpls-security-10.txt>, 2001.

[3] T. Senevirathne, O. Paridaens, *Secure MPLS – Encryption and Authentication of MPLS Payloads*, Internet Draft, IETF Network Working Group, 2001.

[4] L. Fang, M. Behringer et Al., *Security Framework for MPLS and GMPLS Networks*, IETF Draft < draft-fang-mpls-gmpls-security-framework-00.txt>, 2007

[5] E. Rosen, A. Viswanathan, R. Callon, *Multiprotocol Label Switching Architecture*, IETF RFC 3031, 2001

[6] L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, *LDP Specification*, IETF RFC 3036, 2001

[7] B. Jamoussi, L. Andersson, R. Callon, et al. *Constraint-Based LSP Setup using LDP*, IETF RFC 3212, 2002

[8] D. Awduche, L. Berger et al., *RSVP-TE: Extensions to RSVP for LSP Tunnels*, IETF RFC 3209, 2001

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, New York, 1997.

[10] D. Maughan, M. Schertler, M. Schneider, J. Turner, *Internet Security Association and Key Management Protocol (ISAKMP)*, IETF RFC 2408, 1998

[11] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, IETF RFC 2560, 1999.

[12] C. Adams, P. Sylvester, M. Zolotarev, R. Zuccherato, *Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols*, IETF RFC3029, 2001.

[13] D. R. Stinson, *Cryptography Theory and Practice*, CRC Press, 1995.

[14] R. Housley, W. Ford, W. Polk, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, IETF RFC 2459, 1999.

[15] M. Wahl, T. Howes, S. Kille, *Lightweight Directory Access Protocol (v3)*, IETF RFC 2251, 1997

[16] S. Murphy and M. Badger, *Digital signature protection of the OSPF routing protocol*, In Proceedings of the SNDSS'96 Symposium, 1996.

[17] A. Heffernan, *Protection of BGP Sessions via the TCP MD5 Signature Option*, IETF RFC 2385, 1998

[18] V. Klima, *Finding MD5 Collisions - a Toy For a Notebook*,(http://cryptography.hyperlink.cz/md5/MD5_collisions.pdf), 2005.

[19] Martini et al., *Pseudowire setup and maintenance using the Label Distribution Protocol (LDP)*, IETF RFC 4447, 2006.

[20] F. Baker, B. Lindell, M. Talwar, *RSVP Cryptographic Authentication*, IETF RFC 2747, 2000

[21] mpls-linux (http://mpls-linux.sourceforge.net/).

[22] OpenLDAP 2.2.15 (http://www.openldap.org/).

[23] OpenSSL 0.9.7e (http://www.openssl.org/).

[24] Zebra 0.94 (http://www.zebra.org/).

[25] OpenCA 0.9.2.1 (http://www.openca.org/)

[26] MitreTek Systems. Certificate Arbitrator Module (http://cam.mitretek.org/cam/).

Francesco Palmieri holds two Computer Science degrees from Salerno University, Italy. Since 1989, he has worked for several international companies on a variety of networking-related projects, concerned with nation-wide communication systems, network management, transport protocols, and IP networking. Since 1997 he leads the network operation centre of the Federico II University, in Napoli, Italy. He has been closely involved with the development of the Internet in Italy in the last years, particularly within the academic and research sector, as a member of the Technical Scientific Committee and of the Computer Emergency Response Team of the Italian NREN GARR. He is an active researcher in the fields of high performance/evolutionary networking and network security.

Ugo Fiore (Italian Physics degree, 1989) has been with Italian National Council for Research at the beginning of his career. He has been working for more than 10 years in the industry, developing software support systems for telco operators. He is currently with the network management/operation centre of the Federico II University, in Napoli, Italy. His research interests focus on optimization techniques and algorithms aiming at improving the performance of high-speed core networks. He is also actively investigating security-related algorithms and protocols.