# An Authentication for Non-Adjacent LSRs in the LDP

**Morvan D. Müller, Carlos B. Westphall, Carla M. Westphall**

Network and Management Laboratory (LRG), Post-Graduate Program in Computer Science
Federal University of Santa Catarina (UFSC). Phone: +55.48.3317559, Florianópolis, SC, Brazil.
{morvan, westphal, carla}@lrg.ufsc.br

***Abstract.*** *This work proposes a solution for the LDP (Label Distribution Protocol) from the MPLS architecture, that has as objective authenticate, on an end-to-end basis, the establishment of an LSP (Label Switching Path) between one Ingress LSR (Label Switching Router) and its Egress. The objective is to supply the LDP protocol deficiency, that doesn't have one end-to-end authentication mechanism defined for non-adjacent LSRs. The solution makes use of one authentication mechanism based on asymmetric cryptography (private and public keys), that enables the receiver to verify and authenticate the sender. It provides integrity to the information by the use of a hash and additionally protects against replay attacks by the insertion of a nonce to the LDP messages. We don't have knowledge of a similar solution that is effective to solve this problem.*

***Keywords.*** *LDP, Security, MPLS.*

## 1. Introduction

MPLS *(Multi-protocol Label Switchin*g) like RFC3031 [8] is a framework defined by the IETF *(Internet Engineering Task Forc*e) that provides efficient forwarding and switching of data flows across the network for being a packet switching technique based on labels. In the MPLS architecture the LDP (Label Distribution Protocol) is responsible for the distribution of these labels and the establishment of logical ways called LSPs (Label Switched Paths), which are created through LSRs (Label Switched Routers) linked between itself. A weakness in the LDP security can compromise the entire MPLS environment, because the distribution of labels realized by the LDP determines who can participate or not of a MPLS domain through the created LSPs.

The authentication defined for the LDP in the RFC3036 [2] based on the TCP/MD5 option [3], is restricted to adjacent LSRs, because depends on a TCP connection between the involved LSRs. In the case of LSPs between non-adjacent LSRs, in normal conditions, during the establishment of the first LSP, a TCP connection doesn't exist, end-to-end, between these LSRs. So the solution from RFC3036 doesn't deal with efficient way, situations where two LSRs intend to authenticate mutually end-to-end, during the establishment of a new LSP.

This work proposes an end-to-end authentication solution for the LDP in order to overcome this weakness of the protocol, making possible the establishment of LSPs between two non-adjacent LSRs in a safety way. The solution was planned for environments where LSPs crosses external multi-domain environments, not trustworthy between itself , and for this reason need a way to authenticate the endpoints of the LSP during its establishment.

As verified by of related works [5], [12] and [13], currently is unknown a similar solution that effectively attend the proposal of authenticate the establishment of LSPs between non-adjacent LSRs, in a end-to-end basis, in the LDP protocol. To validate this solution, we implement the defined authentication in Linux.

### 1.1 Related Works

[5] describes an end-to-end authentication proposal for the LDP protocol, which has been suggest as a draft (currently expired) for the IETF. The authors haven't made practical experiments as implementations or simulations of the suggested solution. After a deep analysis of this proposal we conclude that it presents an architectural error, fact recognized by its authors, by considering that when sending a LDP message requesting a LSP for some FEC, the source LSR (ingress) knows which will be the destination LSR (egress) that goes to process the request. In the most of the cases it isn't true in the standard form of operation of the LDP protocol. So, the application of this solution is drastically reduced and can only be applied to a minority of cases on the LDP, when the Ingress LSR knows before requesting the LSP who will be the Egress LSR for some FEC.

[12] approaches the security of the MPLS and raises problematic of the authentication during the establishment of LSPs between non-adjacent LSRs in the LDP.

[14] describes a solution that depends on the trustworthiness of the LSPs created between non-adjacent LSRs.

[13] compares the suggested S-BGP architecture with other security solutions defined for the BGP (Border Gateway Protocol), that includes the authentication solution for the LDP based on the TCP/MD5 Option [3] defined in RFC3036.

### 1.2 Organization of the Work

This work is organized in four sections. Section 2 describes the end-to-end authentication solution for the LDP. Section 3 describes the mechanisms to provide the authentication, presents the justifications to the chosen

method of authentication, recommends algorithms for cipher and hash functions and describes the implementation in Linux. For end, section 4 presents the conclusions of this work.

# 2. The End-to-End Authentication Solution for the LDP

## 2.1 Introduction

The proposed model of authentication defines mechanisms to the LDP that make possible to carry the authentication fields through the intermediate LSRs transparently end-to-end, allowing of this form that the endpoints of the LSP are authenticated. The solution makes use of an authentication mechanism based on public-key cryptography (public and private keys) attached to the LDP messages that makes possible to the receiver LSR verifies and authenticates the originator of the messages. It provides integrity protection to the information through a hash mechanism and additionally protects against reply attacks through the insertion of a nonce in the LDP messages. The solution doesn't provide confidentiality based on the fact that information carried in LDP messages, that can be compared with information carried by IP routing protocols like BGP, OSPF and others, doesn't have confidential nature.

As requisite, the solution demands that the LDP operate in "Ordered" control mode. Regarding to the distribution modes of the LDP, "On-Demand" and "Unsolicited", both are compatible with the proposed solution, which can additionally be applied to the CR-LDP (Constrained-Based Routing Protocol) [9] by the fact that it is based on the LDP.

## 2.2 TLVs and Types Defined to the LDP by this Authentication Solution

There where defined two new TLVs (Type-Length-Value) to the LDP to provide the authentication solution, "Hash TLV" and "Nonce TLV". Too, a new "Status Code" type with the value "Authentication Failed" for the Status TLV of the LDP, used in LDP Notification messages to announce that a Label Mapping or Label Request failed on the authentication.

LDP messages involved in the authentication process are: LABEL REQUEST, LABEL MAPPING and LDP NOTIFICATION. These three types of messages give conditions to the LDP to request and send labels for the establishment of LSPs and to notify fails about these operations.

### 2.2.1 Hash TLV

It was defined a new TLV to carry an encrypted hash value.

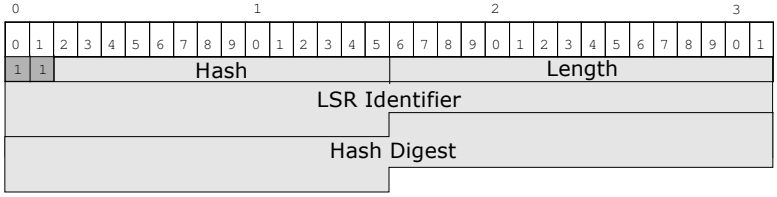


Figure 1. Hash TLV

**U-bit and F-bit:** (1 bit each one). Both set to "1" indicates to the LDP that it should ignore and forward this TLV if it isn't recognized.

**Hash:** (14 bits). This field defines the type of the TLV, "Hash TLV ".

**Length:** (2 bytes). This field indicates the total size in bytes of the following fields:

**LSR Identifier:** (6 bytes). This field identifies the LSR that originated the LDP message and is composed by the LSR-ID (Identification of the LSR) and the "Label Space" in use.

**Hash Digest:** (20 bytes). This field contains a hash value generated from a LDP message, encrypted with the senders LSR private key. In the size definition of this field was considered "SHA-1/160 bits" for the hash function and "Elliptical Curves" for the public-key cipher function, as discussed in section 3.6.

### 2.2.2 Nonce TLV

It was defined a new TLV to carry a nonce a value.

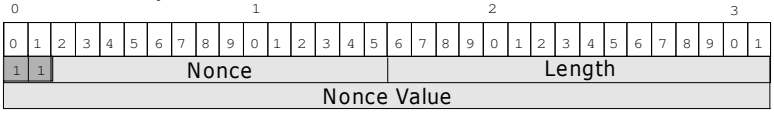


Figure 2. Nonce TLV

**U-bit and F-bit:** follow the same description as 2.2.1.

**Nonce:** (14 bits). This field defines the type of the TLV, "Nonce TLV ".

**Length:** (2 bytes). This field indicates the size in bytes of the field "Nonce Value".

**Nonce Value:** (8 bytes). This field stores a nonce value used to detect reply attacks.

### 2.2.3 Codification and Processing of the Authentication TLVs

The authentication TLVs must be inserted, when LSRs send LDP messages, and processed, when LSRs receiving LDP messages, in LABEL MAPPING, LABEL REQUEST and LDP NOTIFICATION messages, only if the LSR is the EGRESS or INGRESS for the FEC(s) on the LDP message. On the INTERMEDIATES LSRs the authentication TLVs should only be bypassed to the next hop LSR in the way, keeping the same order like in the original message.

## 2.3 Establishment of an LSP between Non-Adjacent LSRs

Aiming a better understanding of the application of this proposal this section explains how the LDP protocol operates to establish an LSP between two non-adjacent LSRs, LERA and LERB (Figure 3). LDP is operating in "On Demand" distribution mode (LSRs only distributes labels if its LDP pairs request it) and in "Ordered" control mode (LSRs only generates input/output labels if they have a correspondent label for the FEC in its label table). From its IP routing table LERA knows the prefix of IP addresses (10.1.0.0/8) that is "behinds" LERB and desires to create a LSP for this FEC. In this MPLS domain, LERA and LERB are respectively INGRESS and EGRESS of the LSP for the FEC 10.1.0.0/8 and NON-ADJACENT LSRs between itself. LERA and LSR1, LSR1 and LSRN, LSRN and LERB are ADJACENT LSRs between itself.

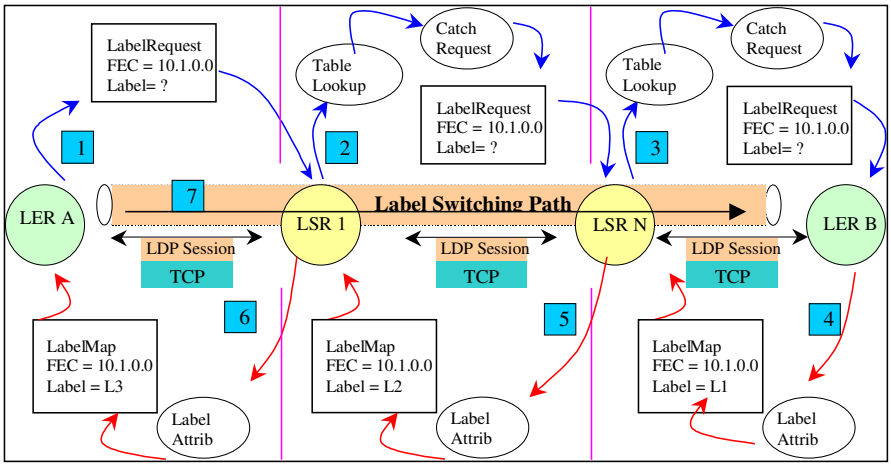


Figure 3. Establishment of an LSP between non-adjacent LSRs

When the LDP is initiated on the LSRs of the domain, in two phases first are established TCP connections and after LDP sessions above these TCP connections between the ADJACENT LSRs, which becomes LDP pairs and begins the "active phase of the LDP" to realize operations with MPLS labels. After this phase LERA can initiate the request of the LSP for the FEC 10.1.0.0/8:

a) in the step 1 (above TCP/LDP session between LERA and the LSR1) LERA sends a LABEL REQUEST for the LSR1 (next-hop for the route 10.1.0.0/8 learned via its IP table) requesting a label for the FEC 10.1.0.0/8;

b) In the step 2, LSR1 make a lookup in its label table (LIB-Label Information Base) and perceives that it does not have a label for this FEC. Then LSR1 modifies the status of the received request to "pending" and encode and sends a new LDP LABEL REQUEST message to its adjacent pair (LSRN) requesting a label for the FEC 10.1.0.0/8 (above TCP/LDP session between LSR1 and LSRN). The same occurs between the LSRN and the LERB (above TCP/LDP session between LSRN and LERB) in step 3.

c) In the step 4, LERB recognizes the FEC 10.1.0.0/8 and perceives that it is the EGRESS for the LSP for this FEC. Then it generates a label (L1) for the FEC 10.1.0.0/8 and sends a LABEL MAPPING for the LSRN informing this label (above TCP/LDP session between LERB and LSRN).

d) In the step 5, the LSRN inserts the received label (L1) into its LIB and perceives that it have a pending request for this FEC originated by LSR1. Then it generates and sends a label (L2) for the LSR1 through a LABEL MAPPING (above TCP/LDP session between the LSRN and the LSR1).

e) In the step 6, between the LSR1 and LERA (label L3), occurs the same, like step 5.

f) In the step 7, when LERA receives the message (with L3) from LSR1 it perceives that itself is the INGRESS LSR for this FEC and at this point the LSP is established between LERA and LERB. Now the FEC 10.1.0.0/8 is mapped to a LSP and all packages with destination to addresses with prefix 10.1.0.0/8 will be routed through MPLS, using labels, and not more through IP, at the network layer.

What the current form of authentication defined for the LDP in the RFC3036, based on the TCP/MD5 Option offers is that, for example, LERB can authenticate the label (L1) in relation to its adjacent pair (LSRN), i.e., only allows to authenticate an adjacent LSR, because depends on a direct TCP/LDP session between them. Between

LERA and LERB at the moment of the establishment of the first LSP doesn't exists a TCP/LDP session end-to-end established, thus it the solution from RFC3036 doesn't apply for this case.

## 2.4 Proposed Authentication Model

Considering the same scenario illustrated in Figure 3 where LERA desires to establish a LSP to a FEC 10.1.0.0/8, Figure 4 illustrates the scenario where LERB (egress) positively authenticate the LDP Label Request sent by LERA (ingress) and returns an authenticated LDP Label Mapping to LERA, using the end-to-end authentication solution. Remember that LDP is operating in "On Demand" distribution and "Ordered" control mode on the LSRs of the environment.

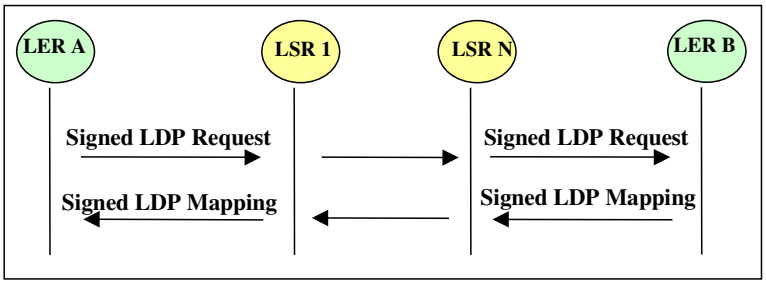


Figure 4. Diagram of the end-to-end authentication

To request the LSP, using the end-to-end authentication solution LERA executes the following steps:

a) codifies a LDP LABEL REQUEST message requesting a label for the FEC 10.1.0.0/8, whose destination route is obtained from its IP routing information;

b) generates a nonce value (for example one sequence number or a timestamp), codifies the fields of the Nonce TLV and attach it at the end of the LDP message;

c) codifies the Hash TLV based on the content of the LDP message. Into the "LSR Identifier" field LERA inserts its LSR-ID and its "Label Space" in use. Into the "Hash Digest" field the input depends on the type of the LDP message. For LABEL REQUEST and MAPPING messages the input will be composed by a byte stream as described in Figure 5:

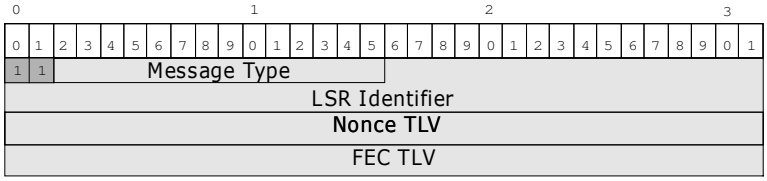


Figure 5. Data input for LDP LABEL REQUEST and LABEL MAPPING messages.

For LDP NOTIFICATION messages the input will be a byte stream as described in Figure 6:

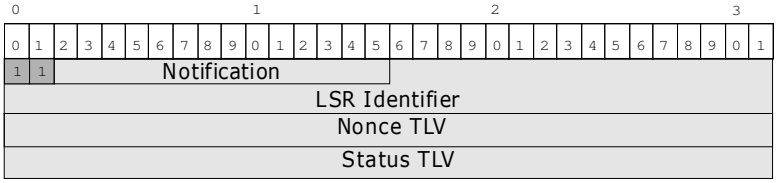


Figure 6. Data input for LDP NOTIFICATION messages.

As LERA is sending a LABEL REQUEST, it forms a byte stream as described in Figure 5 and apply a hash function (this proposal considers the "SHA-1/160 bits" algorithm) to these data. Over the result of the hash function it applies a public-key encryption (this proposal considers the "Elliptic Curves" algorithm) using its (LERA) private key for the encryption. The result of these operations forms the value of the "Hash Digest" field. Then LERA attaches the Hash TLV at the end of the LDP message and the message is ready to be sent.

d) LERA sends the LDP message to the next hop (LSR1) discovered via its IP routing; The INTERMEDIATES LSRs in the way cannot answer the request, so they bypasses the authentication fields transparently until they arrive some LSR that can answer the request, in this case LERB, because it is the Egress LSR for the requested FEC.

When LERB (egress) at the other endpoint receives this LDP REQUEST, it executes the following steps to process the received message:

a) identifies that LERA it is the sender (by the "LSR Identifier" fied of the Hash TLV);

b) verifies in its local configuration if this LSR (LERA) is authorized to establish LSPs. In positive case it selects the public key of LERA (public keys of the authorized LSRs are manually informed in the local configuration of the LSRs);

c) decrypt the "Hash Digest" field from the received Hash TLV using the public key of the sender (LERA). If it have success, means that the sender is AUTHENTIC;

d) in the same form as LERA (Figure 5), LERB generates a Hash based on the received message and compares it with the value of the "Hash Digest" field of the received Hash TLV, so it can verify if this message is UNCHANGED (integrity);

e) LERB generates a local meaning nonce and applies some nonce verification mechanism based on the nonce received from LERA in the "Nonce Value" field of the Nonce TLV. On example of a very simple nonce mechanism is, considering that LERA and LERB clocks are synchronized, LERB can generate a local timestamp and compare it with a received timestamp in order to verify the amount of time that elapsed between the sending and the receiving of the message. If this interval is higher that a threshold time value the message should be discarded.

If the authentication runs successfully, LERB generates a LDP LABEL MAPPING message, codifying it in the same way as LERA, generates a MPLS label to the requested FEC and sends the message back to LERA. If the authentication fails, a LDP NOTIFICATION message with the status code “Authentication Failed” will be sent in back to report the authentication fail (this notification optionally can also be authenticated, i.e., include the authentication TLVs).

To codify the reply message (LABEL MAPPING), LERB executes the same steps as LERA: generates a hash of the message, encrypt it with its (LERB) private key and sends the message back. On the INTERMEDIATES LSRs the message is bypassed until it reaches LERA. LERA then detects that itself is the INGRESS LSR for the FEC of the message (10.1.0.0/8) and precedes the authentication of the LERB. For this it verifies if the LERB is authorized to establish LSPs, verifies the authenticity of the sender, the integrity of the received message, and based on the result of the authentication it executes or not the establishment of the LSP with LERB.

Observes in the illustrated scenarios (Figures 3 and 4), that when LERA requests the LSP it only known the FEC (10.1.0.0/8) for which intends to create a LSP and the IP address of the next hop LSR for this FEC (LSR1), what he got from its IP routing table. It doesn’t know who will be the EGRESS LSR of the MPLS domain for this FEC. Observes also that LABEL REQUEST and LABEL MAPPING messages, that create the LSP, are exchanged using IP routing (network layer). Only after the establishment of the LSP that subsequent packets will be routed by MPLS through the labels generated by the LDP.

# 3. Mechanisms Adopted to Provide the Authentication Solution

## 3.1 Key Distribution

As the authentication solution is based on public-key cryptography, each involved LSR need to generate/know its own pair of keys (public and private) and both LSR entities on the endpoints of the LSP (Ingress and Egress) must know the public key of the LSR at the other endpoint. We suggest two alternatives to distribute these public keys in the environment.

### 3.1.1 Manual Key Distribution

Inform the public keys of the authorized LSRs in the local configuration of each LSR. In the implementation of the prototype in Linux we use this solution for key distribution.

### 3.1.2 Key Distribution using Digital Certification

Inform the public-keys of the LSRs authorized to establish LSPs manually in the local configuration of each LSR raises a common problemc in distributed environments - key distribution. In this context the management of the solution is one of the more compromised aspects. One way to resolve this problem is the use of digital certification that besides supplying an automated solution for key distribution in the environment supplies a higher level of security. Routers (LSRs) have little storage capabilities, generally reduced to a "Flash ROM" memory, but as a digital certificate have in average 1 Kbyte of size, store some certificates in the Flash ROM will not be a restriction.

The proposed solution is:

a) choose a list of Root Certification Authorities (CA's) (we call it "Root-CAs") to emit the certificates for the solution. It can be independent CA's (Verisign, Certsign, etc.) or local solutions like the software "OpenCA" (http://www.openca.org) that is free. Only certificates emitted by "Root-CAs" are valid in the environment.

b) all the LSRs involved in the authentication process must have a certificate emitted by a "Root-CA" and must have stored in its Flash ROM the certificates of the "Root-CAs". Intermediate Registration Authorities (RA) for the "Root-CAs" can also emit certificates, however for it LSRs needs too store the RA certificate plus

the certificates of all higher levels RA's until the respective "Root-CA". The necessary certificates for each LSR can be installed in its setup process. Because the storage limitation in the LSRs is recommended doesn't use an extensive list of "Root-CAs" or much RA's emitting certificates. The standard expiration time of a "Root-CA" certificate (which is auto-signed) is something around 10 years.

c) the sender of LABEL MAPPING, LABEL REQUEST or LDP NOTIFICATION messages must send its own digital certificate with the LDP message to identify itself to the other LSRs. For this purpose a new "Certificate TLV" to store the sender's digital certificate needs to be defined for the LDP, as follow:

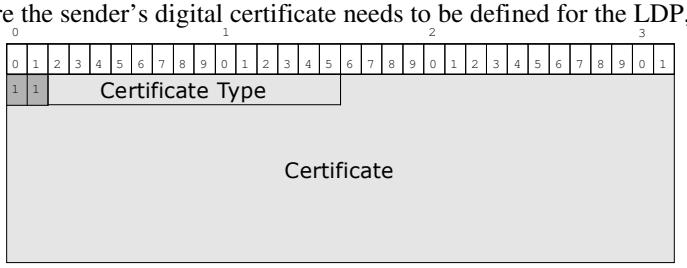


Figure 7. TLV Certificate

**U-bit** and **F-bit:** follow the same description as 2.2.1.

**Certificate Type:** (2 bytes). This field informs the type of the carried certificate. A new certificate extension (*object.id*) will have to be defined, with the finality to authenticate end-to-end the endpoints of a LSP inside the LDP. Details of this definition will not be approached in this document; the objective is give a general vision as how digital certification could be used to distribute the public keys in the solution environment.

**Certificate**: This field stores the digital certificate of the sender, its size depends on the type of carried certificate, but it will have approximately 1 Kbyte.

d) at the receiver (Egress or Ingress LSR for the FEC of the LDP message) to process the authentication, the LSR can validate the senders certificate received in the "Certificate TLV" cause it have locally the certificate of "Root-CA", which is auto-signed, who emits the senders certificate (only certificates emitted by the "Root-CAs" are trusted in the environment). So the receiver LSR can trust and use the public-key informed in the certificate received from the sender to decrypt the "Hash Digest" field of the "Hash TLV" which was encrypted with the senders private key (see section 2.4).

e) After trust the senders certificate the receiver LSR need to verify into its local access control list, using the "LSR Identifier" field from the "Hash TLV" (see section 2.4), to verify if this sender LSR is authorized to establish LSPs.

The function of the "Root-CAs" certificates stored in each LSR Flash ROM is only validate if the certificate presented by the sender LSR is valid and trustworthy. One negative aspect of the use of digital certification to distribute the keys is that it generates a considerable addition of overhead to LDP protocol.

## 3.2 Integrity of the Messages

The sender codifies the LDP message and generates a hash based on it. This hash value is inserted into the "Hash Digest" field of the Hash TLV. The receiver too generates a hash based on the received message and compares with the hash value received from the sender. The hash value can't be modified during the communication because it is encrypted with the private key of the sender.

## 3.3 Origin Authentication

The sender encrypts the "Hash Digest" field using its private key and put its LSR-ID into the "LSR Identifier" field of the Hash TLV. Based on this "LSR Identifier" field the receiver of the message can identify and select the public key of the sender and decrypt the received hash value. If the decryption runs successfully, it will prove that the sender is authentic.

## 3.4 Protection against Reply Attacks

The mechanism to protect against replay attacks is provided by a nonce value, with ever incremental nature, that is inserted by the sender into the "Nonce Value" field of the Nonce TLV in the LDP messages. As the Nonce TLV is included in the hash of the message ("Hash Digest" field of the Hash TLV), it cannot be modified during the communication, and thanks to its incremental nature Nonce TLVs already used in previous communications cannot be replayed. There is many well tested nonce mechanisms [16], like sequence number increments, timestamp based and others. Define the better nonce mechanism to this environment solution will be opened for future works. In the implementation of the prototype in Linux we used a timestamp based nonce and synchronized the clocks of the endpoints LSRs with a NTP (Network Time Protocol) server.

## 3.5 Justifications to the Chosen Model of Authentication

In a first moment, the presented solution seems to have two negative aspects:

a) it uses public-key cryptography to provide the authentication. Generally authentication solutions are based on MAC (Message Authentication Code) that demands minus processing cause uses only a hash function and a secret symmetrical key (that can be a fixed value or a negotiated session key);

b) it doesn't supply mechanisms to negotiate hash and public-key cryptographic algorithms.

The form of operation of the LDP protocol is the justification for these questionings. In the establishment of the first LSP between two non-adjacent LSRs, that is the focus of the presented solution, when the LSR requests a LSP for some FEC, it doesn't know who will be the egress LSR for this FEC. It only knows, via IP routing, who is the next-hop LSR for this FEC. Thus to create the LSP, LDP make a hop-by-hop routing of the request and at the same time generates MPLS labels on each hop until discover the egress LSR. The egress answers with a LDP Label Mapping that is too routed hop-by-hop, being the LSP established only when this Mapping message reaches the ingress LSR, who effectively starts the request of the LSP. Only from this point in ahead, that the next packets are routed via MPLS and not more via IP. So, to create an LSP the LDP exchanges at maximum two messages between the ingress and the egress.

Beyond the flexibility, one of the justifications of the negotiation of session keys and algorithms it's the fact that these will be used in each package later transmitted in a communication session. In the LDP environment, for performance reasons, additional exchanges of messages specifically for negotiation of session keys and algorithms are not justified, because these will be routed hop-by-hop via IP, and after the establishment of the LSP they will not more be used, we trust the MPLS security supplied by the LSP.

Public-key cryptography was chosen because it resolves the problematic of the last receiver (egress) didn't be known at the moment of the LSP request. Because public-key cryptography doesn't demand additional exchanges of messages to authenticate the final receiver, what would be demanded in solutions using MAC, it turns possible a trustworthy authentication with only one exchange of LDP messages between the Ingress LSR and its Egress. As the authentication TLVs are only encrypted/decrypted at the LSP endpoints (Ingress and Egress LSRs), not at the Intermediates LSRs, and is necessary at maximum two exchanges of messages between the endpoints to create the LSP, the use of public-key cryptography is acceptable related to the performance.

## 3.6 Public-Key and Hash Algorithms Recommended

### 3.6.1 Hash Function Algorithms

For Hash functions we suggest as the most appropriate SHA-1 (Secure Hash Algorithm) with digest of 160 bits [7]. The algorithm generates one string with 20 bytes of size. SHA-1 is a standard widely used and considered acceptable related to the security standards. An alternative for performance increase, not significant, is the use of the MD5. Because the hash value is encrypted during the transmission, the security supplied to the hash function isn't much demanded. For future implementations we suggest an analysis of the SHA-256 and SHA-512 [15] algorithms.

### 3.6.2 Public-key Algorithms

For the public-key cryptography functions we suggested the "Elliptical Curves" algorithm [7] as the most appropriate for this solution. The objective is to generate the minimum overhead possible to the LDP protocol. The "Elliptical Curves" algorithm is fast and works with small blocks, so it's favorable to the solution environment. Considering that the input will be a string with 20 bytes of size, result of the SHA-1/160 hash function applying, the result of the cryptography using "Elliptical Curves" algorithm will be one string with the same size, i.e., 20 bytes. The main advantage is that this algorithm doesn't add overhead.

## 3.7 Implementation of the Prototype in Linux

We implemented an prototype of the end-to-end authentication solution presented in this work, using an opened source code project that implements LDP and MPLS in Linux. The selected project was "*MPLS for Linux*" (http://sourceforge.net/projects/mpls-linux/) [6], which is under responsibility to the "source forge" group (http://sourceforge.net). This project is subdivided in two main modules, MPLS-LINUX, that implement the MPLS into the Linux kernel, and LDP-PORTABLE, that implements LDP functionalities. Into the LDP-PORTABLE module we add the code necessary to implement the functionalities of the end-to-end authentication solution presented in this work. The used programming language was "C ANSI" (compiler GCC) and we remained the original program interface of the LDP-PORTABLE module. Tools versions used in the implementation of the prototype had been: Linux RedHat 7.2, Linux Kernel 2.4.19, LDP-PORTABLE version 0.200 (http://prdownloads.sourceforge.net/mpls-linux/ldp-portable-0.200.tar.gz?download), MPLS-LINUX

version 1.170 (http://prdownloads.sourceforge.net/mpls-linux/mpls-linux-1.170.tar.gz?download) and ZEBRA version 0.96 (http://www.zebra.org). Details about the implementation are described in [1].

## 4. Conclusions

We conclude that the end-to-end authentication solution for the LDP presented in this work brought increments that supply weaknesses and deficiencies of the current specification of this protocol, especially about the environment security. An end-to-end authentication form, to make possible a mutual authentication between the endpoints LSRs (Ingress and Egress) during the establishment of a LSP, has fundamental importance to the security of the LDP protocol, especially in MPLS multi-domain environments where the domains are not trustworthy between their selves. It could be verified through the implementation of the prototype in Linux where this objective was readily reached. A classic example of multi-domain interactions is the provisioning of VPNs based in BGP/MPLS when VPNs crosses many VPN providers, described in [10] and [11]. The VPN services are provided based on agreements between the VPN providers.

The solution presented in this work has a generic and widely application scope into the LDP, i.e., it can be applied to all situations of establishment of LSPs between LSRs in the LDP, including LSPs between adjacent LSRs, even its main applicability is between non-adjacent LSRs. Although it hasn't been prove by a practical experiment, like an implementation, the authentication solution presented in this work can perfectly be applied to the CR-LDP protocol, because it has its conception based on the LDP.

As future woks we suggest an evaluation about provide confidentiality to the information carried by the LDP protocol. There are divergent opinions in this direction, so it would be justifiable a deepened study.

## 5. References


[1] MÜLLER, M. "End-to-End Authentication Solution for LDP (Label Distribution Protocol)". Master Degree Thesis, Federal University of Santa Catarina, Post-Graduate Program in Computer Science, Florianópolis, Brazil, Dez. 2002.

[2] L. Andersson, P. Doolan, N. Feldman, et al. "LDP Specification". RFC 3036, http://www.ietf.org/rfc/rfc3036.txt, Jan. 2001.

[3] A. Heffernan. "Protection of BGP Sessions via the TCP MD5 Signature Option". RFC 2385, http://www.ietf.org/rfc/rfc2385.txt, Aug . 1998.

[4] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option". Draft-ietf-idr-rfc2385bis-00.txt, http://www.ietf.org/internet-drafts/draft-ietf-idr-rfc2385bis-01.txt , Mar. 2002. (Work in progress).

[5] J. De Clercq, O. Paridaens, Y. T'Joenset, P. Schrijver. "End-to-end authentication for LDP". Draft-schrijvp-mpls-ldp-end-to-end-auth-03.txt (Expired), Feb. 2001. (We contact with jeremy.de_clercq@alcatel.be, copy of the draft at http://orion.lrg.ufsc.br/~morvan/draft-schrijvp-mpls-ldp-end-to-end-auth-03.txt).

[6] J. LEU; et. al. "Project: MPLS for Linux". http://sourceforge.net/projects/mpls-linux. Fev. 2002 (work in progress).

[7] W. STALLINGS. "Cryptography and Network Security: Principles and Pratice". Ed. Prentice-Hall, Inc., 2ª ed., New Jersey, 1999.

[8] E. Rosen A. Viswanathan, R. Callon. "Multiprotocol Label Switching Architecture". RFC 3031, http://www.ietf.org/rfc/rfc3031.txt, Jan. 2001.

[9] B. JAMOUSSI, et al. "Constraint-Based LSP Setup using LDP". RFC 3212, http://www.ietf.org/rfc/rfc3212.txt, Jan. 2002.

[10] E. Rosen, Y. Rekhter. "BGP/MPLS VPNs". RFC 2547, http://www.ietf.org/rfc/rfc2547.txt, Mar. 1999.

[11] E. Rosen, Y. Rekhter. "BGP/MPLS VPNs". Draft-ietf-ppvpn-rfc2547bis-01, http://www.ietf.org/internet-drafts/draft-ietf-ppvpn-rfc2547bis-04.txt, Jan. 2002.

[12] G. BUDA, D. CHOI, et. al. "Security Standards for the Global Information Grid". IEEE, 0-7803-7272-1/01, 2001.

[13] S. KENT, C. LYNN, K. SEO. "Secure Border Gateway Protocol (S-BGP)". IEEE Journal on Selected Areas In Communications, VOL. 18, NO. 4, Apr. 2000.

[14] Chun-Te WU; Huey-Ing LIU; et. al. "Supporting Virtual Private Dial-Up Services over Label Switching Based Networks". IEEE/IFIP Network Operation and Management Symposium, Hawai, Mai. 2000.

[15] NIST (*National Institute for Standards and Technology).* "Descriptions of SHA-256, SHA-384, and SHA-512". http://csrc.nist.gov/cryptval/shs/sha256-384-512.pdf, Oct. 2000.

[16] M. ABADI; R. NEEDHAM. "Prudent Engineering. Practice for Cryptographic Protocols". IEEE Transactions on Software Engineering, v. 22, n. 1, p. 6-15, 1996.