Proceedings of the 2003 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY June 2003

A Cryptographic Protocol to Protect MPLS Labels

David A. Barlow, *Member, IEEE*, Vasos Vassiliou, *Member, IEEE*, and Henry L. Owen, *Member, IEEE*

Abstract -- We have designed a cryptographic protocol to protect the Multiprotocol Label Switching (MPLS) header used in an Internet Service Provider (ISP) network. This protocol protects the MPLS header primarily against tampering for purposes of hijacking ISP resources. Secondary goals are protection against replay attack and traffic analysis of ISP traffic. The protocol is fast so as to minimize delay introduced into the high-speed MPLS routers. We use the Blowfish encryption algorithm in our encryption system, and smart cards and the Diffie-Hellman protocol in our key distribution system.

Index terms – MPLS, cryptographic, ISP, encryption

I. Introduction

This research develops and evaluates a cryptographic protocol to protect the MPLS header used in an Internet Service Provider (ISP) network. The primary requirement for this protocol is to prevent theft of services from an ISP. To insure that it does not interfere with the quality of service that an ISP provides, our protocol is both fast and simple to implement. The remainder of this paper is organized as background information, design considerations, the encryption system, the key distribution system, and conclusions.

II. Background

MPLS: Multiprotocol Label Switching (MPLS) [1] is a technology proposed by the Internet Engineering Task Force (IETF). It was designed to facilitate several problem areas in the Internet, including routing performance, and is increasingly being adopted by Internet Service Providers in their core networks.

MPLS combines a label-swapping algorithm, similar to that used in ATM, with network layer routing. A label is a short, fixed-length identifier that is used to forward packets, shown in figure 1. In conventional IP forwarding, the router uses a longest-prefix match on the destination IP address to determine where to forward a packet. With MPLS, labels are attached to packets at the ingress point to an MPLS network. Within the network, the labels are used to route the packets, without regard to the original packet header information. These labels can be stacked as a last in first out (LIFO) label stack, enabling MPLS flows to be combined for transport and separated later for distribution.

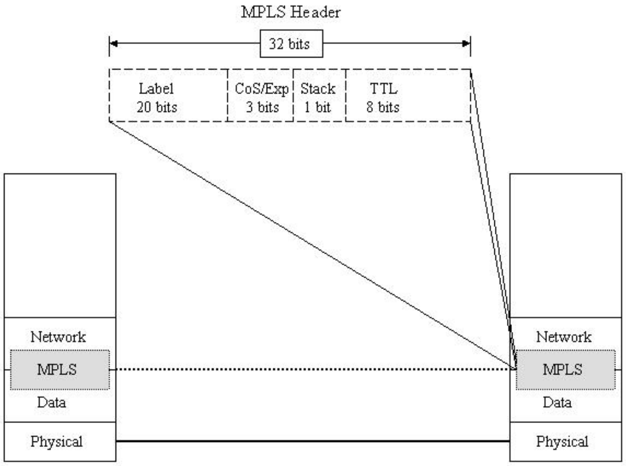

Figure 1 – MPLS Header and Relationship to TCP/IP Network Model

Current proposed protocols for MPLS security, Behringer [2] and Senevirathne et al. [3] discuss two approaches to securing MPLS. Behringer [2] makes 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 only the MPLS nodes themselves are secure. The physical links connecting the nodes are assumed to not be secure – we protect them using our protocol. Senevirathne et al. [3] proposes an encryption approach using a modified version of IPsec. IPsec is defined by the IETF [4], and is an all-purpose encryption protocol that includes key distribution, authentication for the IP header, and authentication and encryption for the IP payload. Senevirathne et al. [3] translate these capabilities to an MPLS environment. Their proposed system does not meet the requirements specified above for our problem for two reasons:

Dr. David Barlow is an Assistant Professor in the Department of Electrical Engineering and Computer Science, USMA, West Point, NY
Dr. Vasos Vassiliou is an Assistant Professor in the Department of Computer Science, Intercollege, Nicosia, Cyprus
Dr. Henry Owen is an Associate Professor in the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA

ISBN 0-7803-7809-1/03/$17.00 © 2003 IEEE

1. It adds at least 128 bits to each MPLS header. This is four times the size of the MPLS header itself. This level of overhead on every packet would probably prove unacceptable to an ISP. It would almost certainly add significantly more processing delay to each packet when compared to a simple encryption scheme.

2. It does not encrypt the MPLS header (but provides authentication). Therefore the header is vulnerable to traffic analysis.

We require fast and inexpensive operation, since MPLS routers are mainly routers without the full capability to do Layer 3 routing operations or are Layer 2 (ATM) switches with some additional capability. For this reason, application layer distribution designs are not applicable in our case. We also require the key exchange algorithm to be aware of the computation burden it imposes on the underlying system and communications performance (quality of service).

Currently, MPLS does not provide header or payload encryption. The only security function employed in MPLS is the use of MD5 [5] to sign and authenticate the control messages sent using TCP. MPLS control messages are transported using IP and do not fall under the scope of this research. They can be secured either by IPsec or any other proprietary method. Integration of the Label Distribution Protocol (LDP) security is an open issue for future study. Nevertheless, MD5 could be used for MPLS header security, since it is already present in the routers' software. MD-5 is particularly suitable in fast re-keying and for the hash or keyed-hash functions that may need to be used.

III. Design Considerations

In our design, we consider the types of attacks we intend to protect from, and operational considerations including speed, span of encryption, and key distribution.

What kinds of attacks are we protecting from? We are protecting the header in an ISP provider's network so that an attacker cannot collect and analyze traffic data, understand route configuration, and eventually create a covert channel. This protocol will protect the links between MPLS routers by protecting the MPLS headers. All MPLS headers will be encrypted with this protocol.
There are four general categories of attacks described by Stallings [6]:

Interruption -- An asset on the system is destroyed or becomes unavailable. This does not have to be a physical asset. This is an attack on availability. We are not protecting from this type of attack with our protocol.

Interception -- An unauthorized party gains access to an asset and can capture data. This is an attack on confidentiality. We are not protecting from this type of attack with our protocol.

Modification -- An attacker modifies the contents of a message. This is an attack on integrity. We are protecting the MPLS header contents with our protocol.

Fabrication -- An attacker inserts counterfeit objects in to the system. This is an attack on authenticity. We are protecting an MPLS network from this type of attack with our protocol.

Our encryption system must be as fast as possible (to minimize processing delay) while meeting the stated security objectives. The protocol should not add bits to the MPLS header or require an additional header to be inserted into each data packet. MPLS routers must be able to recognize a valid, decrypted (received) MPLS header. There will only be a small number of different MPLS headers exchanged between nodes, however there will be potentially hundreds of millions of copies of these same headers exchanged. Therefore, the protocol must not encrypt the same plaintext to the same cipher text. This, along with the fact that each MPLS header is only 32 bits long implies that some kind of stream protocol is necessary. Packets can be lost or damaged in transmission, so the encryption protocol must be self-synchronizing. This further implies that a cipher feedback mode of operation is required.

Each MPLS router in an MPLS network must be able to read and change the label in the MPLS headers it processes. Therefore, a link encryption scheme is necessary. A disadvantage of link encryption is that the MPLS header message is vulnerable at each router. A link encryption device must be present at each end of the link. Each pair of nodes that share a link could have a unique key with a different key for each link. As an alternative, the same key can be used for all links in an administrative domain and different keys can be used at the edges of the domain. This can expose a large number of nodes (belonging to the same domain) if the key is stolen.

ISBN 0-7803-7809-1/03/$17.00 © 2003 IEEE

IV. The Encryption System

A. *Encryption Algorithm*

We propose to use Blowfish [7] as the encryption algorithm. Use of Blowfish has two main advantages over other block ciphers adaptable to our problem:

- It is relatively fast. See table 1 from Counterpane Labs [8].

- It is unpatented and royalty-free.

In order to use Blowfish as our encryption algorithm, we have inserted it into a Cipher Feedback (CFB) mode of operation as described by Stallings [6]. This mode of operation is able to recover from both a bit error, and lost packet problems, producing three bad decrypted MPLS headers for the bit error and two for a missing header (packet).

Algorithm	Clock cycles per round	# of rounds	# of clock cycles per byte encrypted
Blowfish	9	16	18
RC5	12	16	23
DES	18	16	45
IDEA	50	8	50
Triple-DES	18	48	108

Table 1 - Speed Comparisons of Block Ciphers on a Pentium 150

B. *Speeding Processing*

If the Blowfish algorithm was to prove to add too much processing delay to the MPLS routers, we could use a reduced number of rounds to speed the processing. This would likely reduce the security afforded by 16 rounds; however, the published research to date has not been able to break five rounds or more of Blowfish. There is also a trade-off to be made between the time required to break Blowfish (by brute force) and the time between re-keying. If we are willing to re-key more frequently, we can use fewer rounds in our Blowfish encryption and speed the processing at each MPLS router.

C. *Determining if Decrypted MPLS header is Valid*

The method of determining if an MPLS header is valid is to check if the label portion of the header is a valid MPLS label in the context of that particular router. Unfortunately, if we are using a large number of labels there is a small but significant chance that an intruder's MPLS header or a valid header with an error will produce a valid label (though a random one). For example, if we are using 1024 labels there is a one in a thousand chance for a random "encrypted" label to produce a valid decrypted label.

Our proposed solution to this problem is to "steal" bits from the CoS and TTL fields to increase our chances of detecting a bad MPLS header. Using the three bits from the CoS field (which assumes we create separate paths for each class of service and therefore do not need these bits) and two bits from the TTL field would improve our odds of detecting bad MPLS headers. Redoing the example above using the "stolen" bits, we determined that 15 in 1,000,000 bad MPLS headers would decrypt to a valid label.

Changing the expected bit pattern of the unused bits from zero to some other value will not improve the security or ability to detect a valid header, since any value has an equally likely chance of occurring.

D. *Blowfish Setup*

The P and S boxes in the Blowfish algorithm require a large amount of processing to set up with each new key. MPLS routers used by an ISP will in general have a large amount of processing power to cope with the expected large traffic flows. Fortunately, virtually all ISP routers experience both busy periods of high traffic volume, and slow periods with reduced (far below capacity) traffic volume. The Blowfish P and S boxes setup process should be run during times of reduced traffic volume. An alternative to processing the P and S box setup in each router is to do so in a centralized server. This solution has several disadvantages, including providing a single point of failure for the network, and the requirement to transfer the P and S boxes to each router in a secure manner.

V. Vulnerabilities

A. *Denial of Service (DoS) Attack*

Since our protocol operates in CFB mode, an attacker could send junk MPLS packets (that will not decrypt to a valid header) every 3rd packet. This would cause the CFB mode to always produce invalid results and deny service to valid packets. A counter to this type of DoS attack would be to alt-route traffic around a link under a DoS attack using the ability to route individual MPLS paths. However, it would be necessary to track down the source of the injected packets.

B. *Exterior Attack*

A method of bypassing this protocol from the exterior of the network could be implemented at points where the network is connected to other MPLS networks and MPLS is used on those connecting links. In this case, an attacker attempting to pass traffic to a specific destination could inject a series of packets with all possible MPLS headers to map a path(s) through our network. This assumes the attacker could also observe at least one egress point from our network to look for his injected packets. Our protocol provides no protection from this type of service-stealing attack.

VI. The Key Distribution System

A. *Key Distribution Methods*

There are many ways to distribute keys. We discuss each method listed below and conclude with our decision on the method best suited for our system:

1. Direct Physical Exchange

2. Trusted third-party physical exchange

3. Key transmission using a previously used key.

4. Key distribution through a KDC

5. Key transmission using an encrypted connection through a trusted third party.

6. Public Key model

7. Open Key model

For link encryption and if the number of core nodes is not large, the first two methods are not unreasonable. Manual key exchange is the most primitive way of providing a shared key to two parties, but it is still a good way. What is essential in cases like ours (where manual key exchange can be done periodically, but not extremely frequently) is a key exchange protocol that is able to do an automatic periodic refreshment of keys and minimize the risk attached to long-lived master keys.

Key distribution through a KDC has been the basis of many modern key distribution methods and it is always an option. However, we assume that an ISP would be reluctant to use a KDC managed by someone else because it would mean that it would have to rely on a server, out of its control, and which would be a single point of failure. The single point of failure problem is also a reason why the ISP would not be likely to implement the KDC by itself. We expect that the ISP is more interested in providing services that add value to its customer base and bring revenue, rather than invest in a KDC.

Even though using a trusted third party may not be possible in our case, if the two routers have secure connections to a common-centralized server they could use that server to exchange keys. For that method we could design our key establishment protocol based on the framework given by Boyd [9].

Public key cryptography between links is not an option for our system. We do not want to use the limited storage for the public keys of every possible neighbor. In addition, PKI is not fully developed and still not proven in the opinion of Ellison and Schneier [10]. Future improvements in the areas of PKI and router processing ability may enable this scheme.

The Open Key Model bases trust on the existence of a certification authority. Even though it is probably the most scalable model for the Internet, it is not necessarily the best for us because the size and availability of MPLS routers is not changing frequently.

We propose to use SmartCards as a form of manual distribution. We view SmartCards as a good solution for the initial distribution because they remove the danger of a single person having the keys to all routers. They are considered tamper resistant elements and it would be reasonably difficult to get any information out of them. In addition, SmartCards can be used as probabilistic devices, in which case they do not even carry a key but a method for calculating the key. Probabilistic oracle methods are described by Shoup [11].

In addition to the method of key establishment and distribution, we are also interested in the frequency of such a process. As we will see later, there may be a connection between the encryption method and the re-keying frequency if the encryption algorithm proves to be weak or has to be weak because of certain constraints. For an ISP we expect that each router farm or co-location point would be visited at least once a month for maintenance and management purposes. Key distribution can be done during those visits.

In addition to a monthly key change, we propose to frequently create new session keys for each link, utilizing the monthly key to protect the process. Using the same key on a link for a month or more is very risky if the encryption algorithm can be broken during that time. The more the key is used, the more information an adversary has for breaking it. Therefore, by shortening the key's useful life we limit the potential for breaking the key.

By using session keys, we also protect against replay attacks in the case that the attacker has been able to record and/or compromise past communications. Key refreshment (if implemented correctly) should also be able to minimize the dependency between past and future keys and strengthen forward secrecy.

B. *Key exchange, Key distribution, Key agreement*

The first part of our key management protocol is a hybrid transport-exchange and the last part is a pure key exchange.

1. Share/Establish an Initial Secret Key K_0

2. Establish a Session Key – using D-H exponent exchange

3. Encrypt and Authenticate the D-H Exchange

The first part could be done in several ways as follows:

Plain Transfer of K_0:
SmartCard → Router: K_0

Encrypted Transfer of K_o:
SmartCard → Router: $PKE_R(K_0)$

Encrypted Transfer of a Half-Key:
SmartCard → Router: $PKE_R(K_{SC})$

We chose to use the last method. Following this last method, the router has to generate its own half-key and calculate the initial secret key. In order for both machines at the ends of the link to generate the same half key and secret key, we chose to use a function of the old key as the router's half key. This can be a hash function, making it difficult to reproduce the old K_0 and the subsequent keys:

$$K_R = F(Old\text{-}K_0)$$

The secret key is the hash of the concatenation of the two half keys:

$$New\text{-}K_0 = H(K_R, K_{SC})$$

For an initial start where no K_0 has been defined, we should provide a unique IV for each pair to avoid having the same keys at different parts of the same network at the same time.

C. *Session Keys*

The second phase of the key exchange is the establishment of the session keys. We want to be sure that only the router at the other end of the link is communicating with us and that the session key is fresh. This is usually achieved by using one of the following methods:

1. Challenge response method - one of the parties generates a random number and sends it to the other end and the challenge is returned with the key (or whatever the other end's response may be).

2. Timestamps - communication is timestamped. Timestamps are problematic because there are difficulties in ensuring strict clock synchronization.

We chose to use the D-H exchange for this phase because it is the simplest and most proven way. We first agree on a large prime number p and a generator number g and then we perform the exponent exchange. We can send the exponents in the clear and then authenticate them, or we can save a message pair and use a keyed-hash function with K_0 as the key to both encrypt and authenticate, since only the two parties know the secret key.

$$R1 \rightarrow R2 : F_{K0}(g^{xa}, id_A)$$
$$R1 \leftarrow R2 : F_{K0}(g^{xb}, id_B)$$

The two routers can now calculate their session key.

The function used for this exchange can be any function that involves the key in the processing. It can be a keyed-hash function with K_0 as the key or just a transmission of g^{xaK_0}. K_0 is never used to encrypt any data. It is only used in the keyed-hash functions that encrypt and authenticate the D-H exponent exchange. By using the secret key in the function that generates the session key $KS = g^{xaxbK_0}$ we also provide forward secrecy [12].

The proof that the DH method we use is secure against man-in-the-middle attacks is presented by Ateniese et al. [13]. The method is secure because it is difficult for an attacker C to compute either g^{xa} or g^{xb} from g^{xaK_0} or g^{xbK_0}.

We chose to use keyed hash functions because they are bi-one-way functions: they can be one way in either of the variables when the other is fixed [12]. For example, for a fixed K_0 and output value y_0 it should be infeasible to find x in $f(k_0, x) = y_0$. Also, for a fixed x and y it should be infeasible to find K.

VII. Conclusions

Our proposed cryptographic system for the protection of MPLS headers would likely prove successful at its primary task – preventing theft of services by an intruder with interior access to an ISP's MPLS network. Our encryption system uses the (to date) very strong Blowfish algorithm. Without actually implementing and testing this system, we cannot know for sure what delay our encryption protocol would produce in an MPLS router. This delay would be a critical issue for ISP's supporting Quality of Service protocols, like Differentiated Services. Our key distribution system strives to be as simple as possible to administer, yet provide a high level of security for the keys themselves. We believe we have met this goal. However, like the encryption system delay measurement, the proof of any system lies in the implementation and testing.

VIII. References

[1] Rosen, E., Viswanathan, A., and Callon, R. "Multiprotocol Label Switching Architecture," IETF RFC 3031, January 2001.

[2] Behringer, M., "Analysis of the Security of the MPLS Architecture," Internet Draft, IETF Network Working Group, February 2001.

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

[4] Kent, S. and Atkinson, R., "Security Architecture for the Internet Protocol," IETF RFC 2401, November 1998.

[5] Rivest, R., " The MD5 Message-Digest Algorithm," IETF RFC 1321, April 1992.

[6] Stallings, W., Cryptography and Network Security Principles and Practice, 2nd Edition, Prentice Hall, New Jersey, 1999.

[7] Schneier, B., Blowfish symmetric block cipher, 1993.

[8] Counterpane Internet Security Web Site, Copyright Counterpane Internet Security, Inc., 2001.

[9] Boyd C., “A Framework for Design of Key Establishment Protocols,” IEEE ACISP, 1996.

[10] Ellison, C. and Schneier, B., "Ten Risks of PKI: What You’re not BeingTold about Public Key Infrastructure," Computer Security Journal, Volume XVI, November 2000.

[11] Shoup V. and Rubin A., “Session Key Distribution Using Smart Cards,” EUROCRYPT ’96.

[12] Boyd C., “A Class of Flexible and Efficient Key Management Protocols,” IEEE Computer Security Foundation Workshop, 1996.

[13] Ateniese G, Steiner M, and Tsudik G., “Authenticated Group Key Agreement and Friends,” Proceedings of 5th ACM Conference on Computer and Communication Security, San Fransisco, CA, Nov. 1998.