Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 10, 2014

A. Farrel
Juniper Networks

S. Farrell
Trinity College, Dublin

February 10, 2014

Opportunistic Encryption in MPLS Networks

draft-farrelll-mpls-opportunistic-encrypt-02.txt

Abstract

This document describes a way to apply opportunistic encryption between adjacent nodes on an MPLS Label Switched Path (LSP) or between end points of an LSP. It explains how keys may be exchanged to enable the encryption, and indicates how key identifiers are exchanged in encrypted MPLS packets. Finally, this document describes the applicability of opportunistic encryption in MPLS networks with an indication of the level of improved security as well as the continued vulnerabilities.

This document does not describe security for MPLS control plane protocols.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction

MPLS is an established data plane protocol in the Internet. It is found in the majority of core service provider networks and most end-to-end traffic in the Internet will be carried over MPLS at some point in its path. The MPLS data plane is defined by [RFC3031] and [RFC3032].

Data security (i.e., confidentiality) in MPLS has previously relied on just two features:

- Physical isolation of MPLS networks has been used to ensure that interception of MPLS traffic was not possible.

- Higher-layer protocol security (such as IPsec [RFC4302], [RFC4303]) has been used whenever a particular flow has determined that security was desirable.

These features have a number of significant vulnerabilities:

- Networks are increasingly easily compromised physically such that "taps" may be inserted in links between routers.

- Routers may be compromised either in their entirety or through the management/control plane (or misconfiguration). This may result in packets being diverted to transit inspection points on their way to their destination.

- The increased support for point-to-multipoint (P2MP) MPLS means that routers can easily be configured (or misconfigured) to make a copy of data and to send it to an additional destination.

- End-to-end payload security may be hard to manage and operate and is not turned on by default by many users. While this form of security is desirable, the network should also improve the security of data transfer that it offers.

This document describes a mechanism for opportunistic encryption of the MPLS data plane. It shows what part of an MPLS packet may be encrypted and provides a way to indicate that the packet is encrypted as well as to carry a key identifier with each packet.

MPLS opportunistic encryption can be achieved between adjacent Label Switching Routers (LSRs) on an MPLS Label Switched Path (LSP), and also between end points of an LSP.

This document also provides a mechanism for keys to be exchanged to facilitate encryption. Finally, this document describes the

applicability of opportunistic encryption in MPLS networks with an indication of the level of improved security as well as the continued vulnerabilities.

This document does not describe security for MPLS control plane protocols.

Please note that a discussion of the applicability of MPLS Opportunistic Encryption is provided in Section 5.

2. Principles of Opportunistic Encryption

[[Editor note - the introductory material in Sections 2.1 to 2.3 here will likely be mostly or fully replaced with a reference to a more generic OE draft that is in the process of being written. That may lead to some terminology changes, but shouldn't impact on functionality.]]

2.1 Why Do We Need Opportunistic Encryption?

To introduce this discussion we start from a basic view of how encryption is used in IETF protocols.

Say we have two protocol entities, Alice and Bob, and they would like some message "M" sent from Alice to Bob to have confidentiality. Then Alice needs to send M encrypted with algorithm "E" under some some symmetric (e.g., AES) key, "k". Thus Alice wants to send Bob "E(k,M)", but since she and Bob don't yet have such a shared secret they need to agree on the key, "k".

In many IETF protocols, such as TLS (as commonly used) or S/MIME (CMS) or PGP, Alice simply invents a random key "k" and then encrypts that under Bob's public key "Pub-b" and sends Bob E(Pub-b,k) and E(k,M) together. (There are lots of other details and other options for how this can be handled, but we ignore those for now.) In such cases, before Alice can send "E(k,M)", she needs to first get Bob's public key and she needs to be certain that it really is Bob's public key and not Charlie's. That knowledge requires some long-term key management, which is often done using a Public Key Infrastructure (PKI) so that Alice actually stores the public key (Pub-ca) of a Certification Authority (CA), and Bob gets his public key (Pub-b) "certified" by the CA, which means the CA creates a digitally signed data structure "Cert(Pub-ca,Pub-b)". The crucial thing is that Alice, Bob, and a CA need to co-ordinate before Alice and Bob can agree on a key "k", and that process imposes a key-management burden.

Doing such key management is clearly quite possible, since TLS and IPsec and other well-deployed technologies depend on it. But, in

the case of HTTP/TLS on the public web, we see that only roughly 30% of web sites actually take on this burden, even though the software required is ubiquitous and, at least for 2nd level DNS domains in .com for example, there are CAs who offer free domain-validated certificates. While, some of the 70% who don't set up certificates might not actually want confidentiality, there are certainly some who would and arguably many that would benefit from confidentiality, if it just happened out of the box, without an administrator having to do anything. And there are also arguably many other protocols where the same is true.

Opportunistic encryption (OE) offers a mechanisms to achieve encryption between Alice and Bob without resorting to key-management through CAs and without relying on manual configuration of keys.

2.2 Opportunistic Encryption at 10,000ft

Instead of the "key transport" mechanisms described in Section 2.1, opportunistic encryption uses "key agreement". With key agreement, both Alice and Bob contribute to calculating "k" (instead of the the mechanism where Alice invents "k" and safely transports it to Bob encrypted with Bob's public key as "E(Pub-b,k)").

Assume that Alice and Bob are using some protocol where they can

exchange a few messages in order to agree on the key "k" to use. With a Diffie-Hellman key agreement ("D-H") both Alice and Bob have public and private values, where the private value can be randomly generated, perhaps even once per message "M". They swap the public values, and can then, thanks to the "magic" of Diffie-Hellman, derive a key "k" that nobody else can know.

In this way Alice sends Bob "Pub-a" and Bob sends Alice "Pub-b" and at that point both of them can safely calculate a shared secret "k" from those values. And after that Alice can send Bob "E(k,M)".

From here on, we change the terminology slightly and refer to Alice as the initiator, with private key "i" and Bob as the recipient, with private key "r" so that our notation is closer to that used in IPsec's IKE, on which we model our use of OE.

The "magic" of D-H works as follows. Let "p" be well-known large prime number that we use for all modular arithmetic (meaning that "a^b" is actually "(a^b) mod p"), and let "g" be another well-known value (called a generator for the group determined by "p"). Also let Alice and Bob's private values be "i" and "r" respectively. Now, if Alice sends Bob "g^i" as her public value, and Bob similarly sends Alice "g^r" then both of them can easily calculate "g^(i*r)" or "g^ir" but nobody else can, since calculating

"x" when only given "g^x" is a computationally hard problem for any "x". Once both Alice and Bob have the value "g^ir" in hand, they can easily derive a value "k" from that using any of a number of well-known key derivation functions (KDF).

As you can see from the above, Alice and Bob do not need to pre-arrange anything other than "g" and "p", and those can be public values, that are used by everyone everywhere (or at least by all participants in a particular deployment). Yet, Alice and Bob have managed to derive a common value for a key "k" that they can use to encrypt (and decrypt) "M".

This kind of opportunistic encryption provides strong confidentiality and can be built into any protocol that allows Alice and Bob to occasionally exchange public values.

There are also additional advantages to key agreement when compared to key transport. The most important of those is that with key

agreement we can easily ensure that k has a property called Perfect Forward Secrecy (PFS). That means that an attacker has to separately attack each key k. In contrast, if we use the key transport approach, then an attacker who somehow accesses Bob's private key "Priv-b" can record lots of traffic and later go back and decrypt all the "E(Pub-b,k)" values that all Alice's ever sent to Bob. With key agreement as described, since both Alice and Bob contribute to the value k, and since Alice and Bob will typically periodically generate new private values i and r (perhaps even for every single M), compromise of one party is far less catastrophic, and an attacker who gets access to one private value gets far less benefit.

2.3 What about a Man-in-the-Middle?

But OE is not resilient to Man-in-the-Middle (MITM) attacks. The problem is that Alice does not know that it was really Bob's public value that she received; it could have been Charlie's public value sent by Charlie. And Charlie could also send Bob his public value pretending to be Alice. Now Charlie can share a key with Alice and a key with Bob so that Charlie can sit between Alice and Bob decrypting what he gets from Alice and then re-encrypting it to send to Bob. Neither Alice nor Bob can tell that Charlie is present as a "Man-in-the-Middle" and both Alice and Bob think they are safely exchanging encrypted messages.

A MITM attack like that is bad and making a protocol proof against such attacks comes at the cost of the key-management burden described in Section 2.1. Most IETF protocols to date require that such MITM attacks not be feasible.

However, despite its vulnerability to MITM attacks, OE still has value in some circumstances. This value arises because of the difficulty of inserting a MITM actor, and the cost of processing for the MITM in the case of a very large number of OE relationships. In particular, where the choice is between no encryption (as has been the case for MPLS to date) and OE, it is clear that using OE offers better (although not the best) security.

Consider the case where an attacker taps a link on the path between Alice and Bob. In this case, the attacker can capture every packet between the two parties, and if there is no encryption, can read every message. Furthermore, consider that the attacker could tap a

fiber in the core of the network and so capture every packet between a large number of Alices and their corresponding Bobs. In these cases, Charlie can operate as a "passive MITM" since all he has to do is watch the packets.

With OE in use, Charlie is forced to be an "active MITM". That is he must engage in the D-H exchange between each pair of Alices and Bobs, and he must must decrypt and encrypt each packet he wants to inspect. This imposes a higher cost and is especially burdensome if he is attempting to do it in parallel for lots of Alice/Bob pairs using lots of different keys and communication sessions.

Furthermore, when D-H is in use for OE, management tools can be used to detect the presence of Charlie as a MITM. This is because Charlie has to agree one key "kA" with Alice, and a different key "kB" with Bob. As far as we know, Charlie cannot arrange that kA equals kB because both sides contribute to the key value in the D-H key agreement. That means that if Alice and Bob can check with each other what value of "k" they are using and the values do not match, then they know that Charlie is present. What is more, Alice and Bob can make this check on the value of "k" for any of the "E(k,M)" they ever exchanged.

Thus, in the case of a fiber tap where many Alice/Bob pairs are being monitored, it only takes one Alice and Bob to detect the MITM attack for all Alice/Bob pairs to be alerted to the problem. In such cases the cost of detection for Charlie may be even greater than the cost of performing the MITM attack.

Hence we conclude that OE can have considerable value when used in MPLS networks.

2.4 OE in MPLS Overview

[[Editor Note - the details here are suitable for an early revision draft. We might change to ECDH later, or to use another KDF, or symmetric cipher mode. All that is for discussion.]]

The basic requirement for MPLS OE is that we want to provide a way for two MPLS nodes to do an OE key exchange and to derive a session key from that to use in MPLS packet encryption.

To do that we use a Diffie-Hellman key exchange as outlined in Section 2.2. We model this on IKE [RFC5996] using essentially the same parameters. We feed the shared Diffie-Hellman value, which is g^ir, into a standard key derivation function (KDF) that also takes as input the LSP identifier (LSP ID) together with the sending and receiving LSR IDs - where the the sending LSR is the point of encryption and the receiving LSR is the point of decryption such that the pair of LSRs define the Security Association (SA). These additional inputs are used to ensure that we end up with different keys on an LSP even if the same g^i and g^r values are re-used. The KDF to be used here is as defined in [RFC5869].

D-H values used for MPLS OE MUST be of at least 2048-bits. Implementations of MPLS OE MUST support the 2048-bit modular exponentiation (MODP) group from Section 3 of [RFC3526] and SHOULD support the larger MODP groups from [RFC3526].

This document also defines the mechanism used to derive an identifier for a key (the key-id) from the shared Diffie-Hellman value, which is also based on the KDF output. The key will be used with a symmetric encryption algorithm, such as AEAD_AES_GCM_128 (the default, following [RFC5116]).

As with any symmetric block cipher, one should not use the same key for too long. The nonce defined for MPLS OE keys is derived using a 96 bit counter incremented by one for each encrypted packet. It is critical for security that nonce values MUST NOT be re-used with a given key. (This is an inherent issue with how AES-GCM or any counter mode achieves high performance.)

Accordingly, implementations are RECOMMENDED to change keys at least every 2^32 packets, and MUST change keys before encrypting 2^64 packets. For an LSP running over a fully-busy 100Gbe interface, we might assume that means roughly 160 million packets per second, or roughly 2^44 packets per day. The 2^64 limit therefore means changing keys daily in the busiest cases of some of the largest current links capacities.

To support key change, this document defines a way for two LSRs using a key on an LSP to agree a new key and to switch over to using that key when desired. That means that implementations MUST be able to handle at least two keys (old and new) for a given LSP. Once a new key has been agreed then it should be used for sending packets; once encrypted data packets protected with the new key have been successfully received, the old key should be discarded. Section 4 describes how two LSRs agree keys, and to agree a new key, two LSRs simply run the same key agreement exchange, but this time protected with the old session key as described in Section 4.3. This process can, of course, be repeated any number of times for the same LSP. It is RECOMMENDED that the key on an LSP be changed at least once every day or every 10^6 packets whichever is sooner.

[[Editor Note: These values need considered in the light of latest cryptology advice, but also understanding that this is "best-effort" OE.]]

In the event of a key agreement exchange or decryption failure, an alarm MUST be raised to the operator. Default (i.e., node-wide) and per-LSP behavior SHOULD be configurable in this case: actions may include reverting to non-encrypted traffic, re-attempting key exchange, or tearing down the LSP. Note that a simple attack on OE is to tamper with key agreement exchange messages or encrypted packets so that OE fails. Such attacks may be intended to cause the LSP to operate without encryption, so an operator should consider this when setting the behavior in this case.

Section 7.1 also discusses a mechanism that allows a pair of LSRs using OE on an LSP to detect that a MITM attack has happened. For this, we simply define a function of the shared secret, which can be logged and later compared. Note that logging a sample of these "witness" values will likely be sufficient to detect pervasive MITM attacks. As with the key-id, we base this on the same KDF output.

An additional discussion of the applicability of MPLS OE is found in Section 5.

3. MPLS Packet Encryption

MPLS packets may be individually encrypted according to the mechanisms described in this section.

When an MPLS packet is encrypted, this is indicated by the insertion of a new special purpose label [ID.ietf-mpls-special-purpose-labels] in the label stack. This is referred to as the Opportunistic Encryption Label (OEL). The format of the OEL is described in Section 3.1.

The OEL MUST have the bottom of stack bit (the S bit) set and MUST be followed by a pseudowire control word [RFC4385]. The format of the control word is described in Section 3.2.

The remainder of the MPLS packet is encrypted and cannot be parsed without decryption. Implementations MUST support the AEAD_AES_GCM_128 encryption algorithm, as specified in Section 5.1 of [RFC5116], which is the default as described in Section 4.2 of this document.

Note that it is critical that a new nonce is used for every encryption. The nonce is an implicit packet counter. The initial nonce value is derived from the HKDF output at key agreement time and the counter is incremented by one for each packet encrypted on the sending side and by one for each packet successfully decrypted on the receiver side.

Although the nonce is not transmitted with the packets, a 16-bit counter carried in the control Word indicates the nonce value modulo 65536. This feature allows a receiving node to quickly spot that a packet has been dropped and resynch its own counter in order to be able to continue to decrypt received packets. In the event that the counter cannot be resynchronized or that more than 65536 packet are lost in one batch the receiver will encounter a decryption error. In this case the receiver may report a general decryption error or may attempt to resynchronize by advancing its own counter in units of 65536 according to the modulo value in the received packet. Note that incrementing the counter in order to test for decryption failure does generate a potential DoS if, e.g., an attacker decrements the nonce-mod-65536 value. Implementations that do such tests SHOULD maintain a small maximum window size beyond which they will cease attempting to decrypt. It could be that throwing an error might be the more effective response if the packet loss rates are expected to be low enough.

It should also be noted that the output from encryption will be 16 octets longer than the input.

The bottom of stack bit is set in the OEL to stop implementations continuing to search down the label stack (which is encrypted) and attempting to use the data as though it was a valid label stack. The control word is needed because many implementations that find the bottom of stack expect the next bytes to be a control word or protocol indicator.

The position of the OEL and control word depend on whether hop-by-hop

or end-to-end encryption is being applied.

Figure 1 illustrates the format of an example MPLS packet before and after hop-by-hop opportunistic encryption. The left hand part of the figure shows a normal MPLS packet with a label stack and payload. The bottom label in the stack has the S bit set. The payload is the data carried by the MPLS packet (such as IP) and may be prefixed by a control word.

The right hand part of Figure 1 shows the same packet after it has been encrypted. The top of stack is the OEL with the S bit set. The OEL is followed by a control word. Everything that follows the control word is the entire original MPLS packet encrypted.

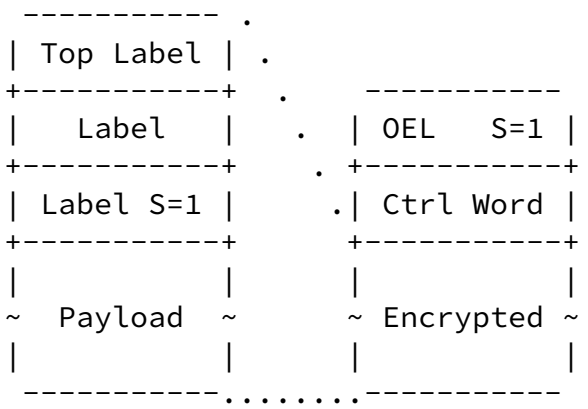

Figure 1 : The Use of the OEL for Hop-by-Hop Opportunistic Encryption

Figure 2 illustrates the format of an example MPLS packet before and after end-to-end opportunistic encryption. The left hand part of the figure shows a normal MPLS packet with a label stack and payload. The bottom label in the stack has the S bit set and the payload may be prefixed by a control word. The right hand part of the figure shows how the top two labels (or however many labels are needed for end-to-end delivery) remain at the top of the label stack. Then follows the OEL with S bit set, and a control word. The remainder of the packet is encrypted and contains the rest of the label stack and the payload.

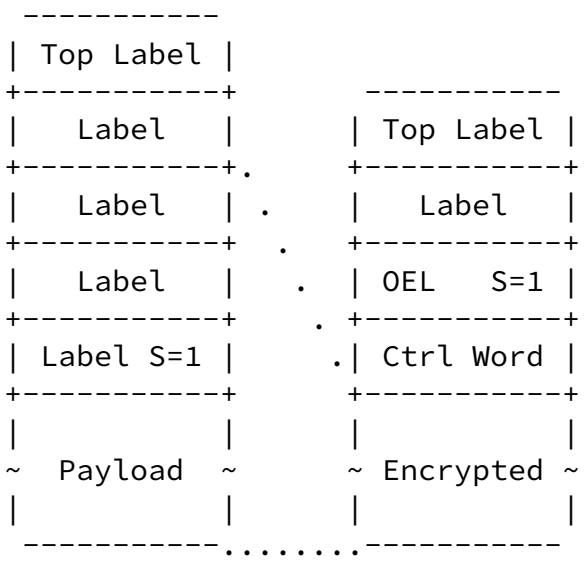

Figure 2 : The Use of the OEL for End-to-End Opportunistic Encryption

3.1. Opportunistic Encryption Label

The Opportunistic Encryption Label (OEL) is a normal label stack entry carrying a special purpose label with value TBD1 to be assigned by IANA. The format of the label stack entry is defined in [RFC3032] and shown in Figure 3.

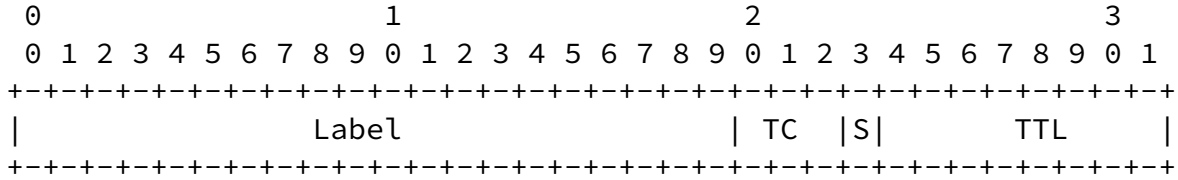

Figure 3 : Format of the OEL Label Stack Entry

Label: Set to TBD1 to indicate this is an OEL
TC: SHOULD be copied from the TC of the first encrypted label.
S: MUST be set to one.
TTL: SHOULD be set to zero to prevent encrypted packets being accidentally forwarded beyond the point of intended decryption.

The sending LSR MAY choose different values for the TTL and TC fields if it is known that the OEL will not be exposed as the top label at any point along the LSP (for example, by penultimate hop popping).

3.2. Control Word

The control word is inserted after the OEL as described in Section 3. The S bit set to one in the OEL and the presence of the control word helps protect against transit nodes that may perform hashing or inspection of the label stack and payload packet headers when forwarding MPLS packets (for example, to enable ECMP). The control word indicates that the payload is not a protocol that can be meaningfully hashed or inspected.

The format of the control word is defined in [RFC4385] and shown in Figure 4.

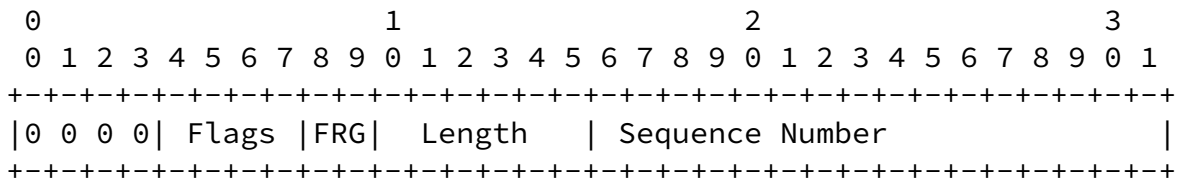

Figure 4: Control Word for Opportunistically Encrypted MPLS

Flags: The Flags field is treated as a four-bit number. It contains the key-id that identifies the algorithm and key as established through configuration or dynamic key exchange as described in Section 4.
FRG: Must be sent as 0, and ignored on receipt. Fragmentation is not used.
Length: MUST be sent as 0, and ignored on receipt.

Sequence Number: This field contains the packet counter (nonce) for the encryption algorithm and key currently in use modulo 65536. It can be used by a receiver to quickly check that the value of the nonce being used for decryption is likely to be correct.

3.3. Considerations for ECMP

As previously stated, the S bit set in the OEL and the presence of the control word prevent implementations from attempting to use the encrypted MPLS packet and its payload to determine a hash value for uses such as ECMP. However, the resultant label stack shown in Figure 2 will probably not provide sufficient entropy for ECMP purposes.

In order to increase the entropy, an implementation that inserts an OEL and OEL MAY also insert an Entropy Label Indicator (ELI) and Entropy Label (EL) as defined in [RFC6790] ELI and EL are positioned in the label stack before the OEL as shown in Figure 5. The setting of the fields in the ELI and EL label stack entries are as described

in [RFC6790].

The ELI and EL will normally occur immediately before the OEL, but they MAY be placed higher up the label stack.

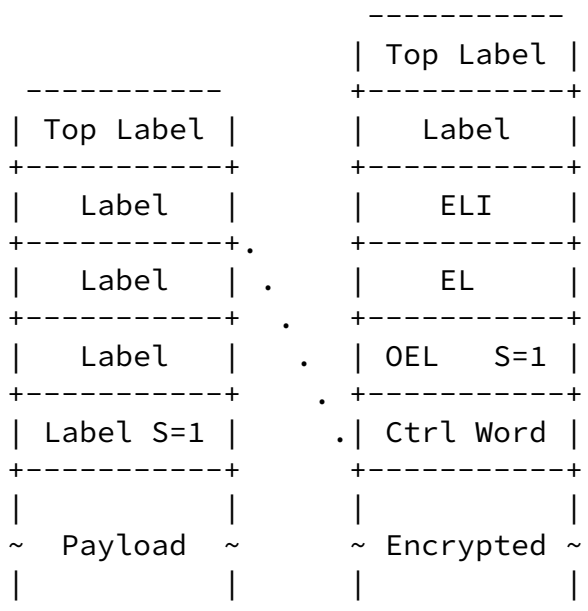

Figure 5 : The Use of ELI and EL with OEL

3.4. Backward Compatibility

Keys and encryption algorithms may be configured manually or exchanged dynamically as described in Section 4. These mechanisms provide a preliminary way to protect against a sender encrypting data that the receiver cannot decrypt, however, misconfiguration may lead to a sender using the OEL when the receiver does not support opportunistic encryption.

When a node finds an unknown label at the top of the label stack it must discard the packet as described in [RFC3031]. Therefore, when a receiver discovers the OEL and does not support opportunistic encryption it will discard the packet. The net result is that when a sender uses opportunistic encryption in error, all packets that it sends on the LSP will be discarded by the receiver. Note that in this discussion, "the receiver" may be the next hop if single hop encryption is used, or may be the end of the LSP if end-to-end encryption is used.

Transit nodes that are not actively participating in the encryption will not inspect the OEL except potentially as part of an ECMP hash, and it should be noted that the use of Special Purpose Labels in

hashing is strongly discouraged. This means that transit nodes will not encounter the OEL during normal packet processing and will not discard packets.

3.5. MTU Considerations

Adding the OEL and Control Word as described above will reduce the available data size by 8 octets. Furthermore, as described in in Section 3, the output of the encryption algorithm is 16 octets longer than the input. Therefore, the use of MPLS OE reduces the available MTU by 24 octets.

When end-to-end OE is in use this can be considered by the ingress LSR, however, when single-hop OE is in use the participating LSRs need to advertise this reduction in link MTU so that the packets do not overflow. MPLS packets MUST NOT be fragmented as a result of

OE.

3.6. Recursive OE

The use of OEL and control word described in Section 3 may be applied recursively. That is, the payload of an encrypted MPLS packet may, itself be an encrypted MPLS packet. This may be particularly useful in the case where an MPLS VPN has native MPLS traffic.

There are no special considerations except to note that encryption and decryption processing may be burdensome if an LSP and its payload LSP have OE applied at the same LSR. Additionally, it should be noted that, as described in Section 3.6, each recursive encryption reduces the MTU by 24 octets.

4. Key Exchange For Opportunistic Encryption in MPLS

For encryption to be useful both ends of an encrypted session must know which algorithm is in use and which key to use. The mechanism described in Section 3 provides a way to indicate an index into a table of algorithms and keys that can be used to decrypt an encrypted MPLS packet.

It is possible that this table has been manually configured or set up using a key exchange protocol such as Internet Key Exchange version 2 (IKEv2) [RFC5996]. However, such a process implies a stable security association between encrypter and decrypter of MPLS packets. Such a stable association is not consistent with the concept of opportunistic encryption.

This section provides a mechanism for adjacent MPLS LSRs, or for a pair of LSRs at opposite ends of an MPLS LSP, to dynamically

exchange keys and algorithm identifiers so that encryption may be applied opportunistically.

The mechanism uses message exchange in the MPLS Generic Associated Channel (G-ACh) [RFC5586]. This channel is in-band with an LSP and may be used to carry messages between neighbors or between the end points of the LSP. A type field within the common message header, the Associated Channel Header (ACH), is used to indicate the type of message carried.

Nodes that receive G-ACh messages and do not understand them, or nodes that understand the G-ACh but do not recognize the ACH message type drop the packets as described in [RFC5586].

4.1. Associated Channel for Key Exchange

The Associated Channel Type value TBD2 indicates that the packet contains a Key Exchange Protocol message as defined in Section 4.2.

Implementations that do not support key exchange in this manner will discard received packets with this Associated Channel Type as described in [RFC5586]. This will result in no dynamic key exchange, but other key definitions are still supported (such as manual configuration) and may be used to construct a table of algorithms and keys that can be used to achieve MPLS encryption using the mechanisms described in Section 3.

Note that TBD2 indicates the use of Diffie-Hellman key exchange. If ECDH is added later a new value will be required.

[[Editor Note. An alternative to this is to embed the type of key exchange within the key exchange messages.]]

4.2. Key Exchange Protocol

[[Editor note: This key exchange protocol is bidirectional, yet LSPs are usually unidirectional. That means we need to establish a channel for the return messages (similar to that in LSP Ping) or use a different approach to Diffie-Hellman.]]

A session key is to be established between an initiator (Alice) and a recipient (Bob). The D-H public value for Alice is g^i and for Bob, g^r. The shared Diffie-Hellman value is g^ir.

g^ir is represented as a string of octets in big endian order padded with zeros if necessary to make it the length of the modulus.

Both g^i and g^r will be 2048 bits long, if the Diffie-Hellman modulus is 2048 bits long.

The key exchange payload is modelled on that from Section 3.4 of

[RFC5996], and is shown in Figure 6.

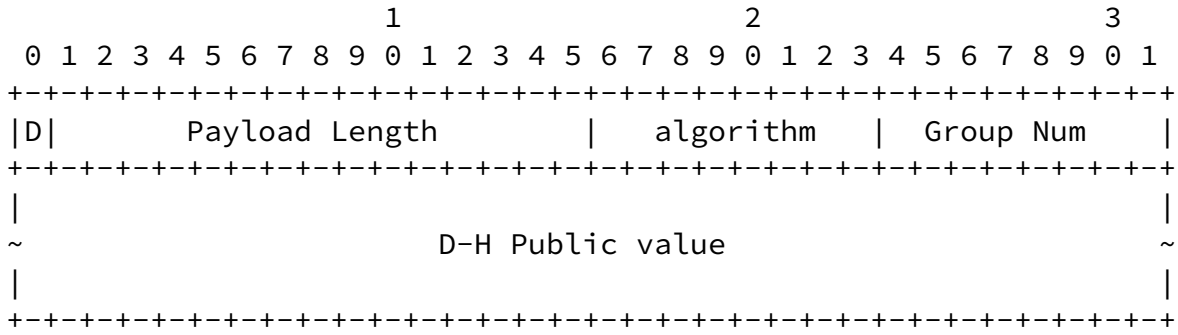

Figure 6 - Key Exchange Message

The flag D denotes the direction of the message, '0' indicates a message from initiator (Alice) to recipient. '1' indicates the reverse direction.

The Payload Length field contains the number of octets following the payload length field (including the group number). This is 15 bits wide.

The algorithm field is a one octet field that specifies both the KDF to use and the symmetric algorithm to be used for data packet encryption. A registry for values of this field is defined in Section 8.3. The value 0 is used to indicate the default KDF and symmetric encryption mode.

The Diffie-Hellman Group Num is from [RFC3526], so the group number for 2048 MODP is decimal 14. Note that this is a one octet field, but is two octets in the [RFC5996] equivalent. This is not an issue because there are only 30 MODP groups defined at present and new groups are not added frequently.

The D-H public value will contain g^i or g^r depending on the direction (i.e., the setting of the D flag) and is in big endian order.

The length of the Diffie-Hellman public value for MODP groups MUST be equal to the length of the prime modulus over which the exponentiation was performed, prepending zero bits to the value if necessary.

Once both sides have derived g^ir they need to feed that and the other inputs described in Section 2.4 into the KDF indicated by the algorithm field. With the default algorithm (value zero), the KDF to be used is HKDF as specified in [RFC5869].

The parameters for the use of HKDF are:

Hash: SHA-256

Salt: Not used. [[Editor Note: maybe we should?]]

Skip: Do not skip.

Info: The catenation of a fixed string indicating use of MPLS OE, with the value "MPLS-OE", the first 32 bits of the key exchange message, with the D flag set to 0, plus the LSP ID and the sender and receiver LSR IDs in that order. That is:

MPLS-OE||0||payloadLen||alg||group Num||LSP-ID||i-LSR-ID||r-LSR-ID

L: The output length in bits is 272.

The fixed string "MPLS-OE" is used as an input here to prevent potential cross-protocol attacks. Those might otherwise be possible if this mechanism were to be copied in other protocols. (If copying this mechanism for any reason, then a different fixed string value should be used.)

LSP-ID is a unique identifier shared between the initiator and receiver (Alice and Bob) that uniquely identifies the LSP.

[[Editor note: This identifier is only needed if the scope of the key is per LSP, but that seems a better scope given the need to rotate the key after a certain number of packets have been transmitted.

Currently the LSP-ID is known along the LSP and at the two end points if RSVP-TE is used for signaling, or potentially if the LSP is manually configured. It is not so clear in LSPs established using LDP. Probably, however, we can use the FEC as defined for RFC 4379 and its extensions.]]

i-LSR-ID and r-LSR-ID are the LSR-IDs of the initiator and receiver respectively, where an LSR-ID is the 32 bit, globally unique identifier of the LSR as described in [RFC5036] and [RFC4990].

The default encryption algorithm, AEAD_AES_GCM_128, specified in

Section 3, requires a 128 bit session key.

The 272-bit HKDF output is the catenation of the session key, the key-id, the witness value and the high-order 16 bits of the initial nonce value in that order. That is the session key is the leftmost 128 bits of the HKDF output. The key-id is the next 4 bits, the witness value is the next 124 bits and the last 16 bits are the 16 most significant bits of the initial nonce value. The low order 64 bits of the initial nonce value are set to zero before the first call to the AES-GCM encryption function. The key-id is carried in encrypted packets as described in Section 3.2.

[[Editor note - It is assumed that a 4 bit key-id is adequate in a system where, for any one LSP there is one active key and one new or replaced key. There might also be more than one algorithm, and it is possible that new keys need to be pipelined if roll-over is frequent.]]

[[Editor note - we might want to consider deriving the witness value from a separate invocation of the KDF that does not depend on the LSP-specific inputs. The benefit from that would be that the same MITM-detection infrastructure could be used for many protocols. However, that would require standardizing a generic D-H MITM-detection protocol, or at least formats, in order to be useful. We also need to consider what additional information needs to be logged with the witness value so that comparisons can easily be made at scale but without creating new privacy-invasive meta-data. (That last is not much of an issue for MPLS-OE, but could be elsewhere.) At present we do not intend to go for the generic MITM-detection approach, but it is worth considering.]]

4.3. Protecting the Key Exchange Protocol Messages

As described in Section 2.4, once one key exchange has been successfully completed, further key exchanges should be protected using a previous key. This is simply achieved since key exchange messages are, themselves, carried in MPLS packets on the LSP and may be subject to encryption exactly as any other packet.

5. Applicability of MPLS Opportunistic Encryption

MPLS OE provides another tool in the security and privacy toolkit. It is not a panacea and does not solve (nor is it intended to solve)

all security or privacy problems. In particular, the use of MPLS OE does not protect user-data end-to-end that might be better secured using encryption at the IP layer or at higher layers.

As noted throughout this document, the intention of OE in MPLS is to

allow one LSR to enable encryption between itself and its neighbor, or between itself and the other end of an LSP, in a dynamic and un-planned way. This can have benefits in a number of scenarios where the network that generates MPLS traffic transmits it over another network (for example, carrier's carrier, or some deployments of enterprise network). Additionally, the use of MPLS OE might allow a service provider to offer a secure edge-to-edge service for a variety of applications ranging from VPNs through pseudowires and where the payload traffic might not always be IP. Lastly, in some non-traditional carriers the user data belongs to the operator or is the direct responsibility of the operator (for example, in data centers, or in large-scale private networks).

As with all security mechanisms, there is a trade-off between a number of factors. On one side is the completeness of the security of the user-data, and on the other side is the complexity of configuring and managing the necessary security associations. Furthermore, while mechanisms closer to the end-user than MPLS OE (for example, TLS and IPsec in tunnel mode) provide better security for user-data by virtue of not transmitting the data across any network hops without it being encrypted, such mechanisms often expose more metadata for inspection by snoopers within the network.

Additionally, while a variety of per-link encryption mechanisms exist and could be used to guard against attacks such as fiber taps, those approaches do not protect against subverted nodes (i.e., routers) on the path since, by definition, per-link encryption does not protect packets once they come off the link. MPLS OE in the end-to-end LSP mode protects packets on the links and as they cross transit routers.

Nevertheless, it is not the purpose of this document to recommend the use of MPLS OE to the exclusion of all other encryption techniques. As already mentioned, MPLS OE is offered as another tool in the tool kit and users as well as network operators are strongly advised to consider using a variety of tools to achieve the level of security and privacy that they desire.

Note that, in order that OE can be used, one end of a peering (neighbor or LSP end) must decide to attempt OE and the other end must support it. This can be determined by the message exchanges described in Section 4.2 since if one peer does not send a key exchange message then encryption will not be used, and if the other peer does not respond then it is unwilling or unable to decrypt messages.

MPLS OE should be applicable to all forms of MPLS. That is, it should be possible to use it in RSVP-TE systems, in LDP systems, and in MPLS-TP systems (by which we mean those that have manually configured

LSPs). Equally, it should work for point-to-point (P2P) and multipoint-to-point (MP2P) uses of MPLS because there is a simple relationship between the sender (encrypter) and the receiver (decrypter) in both cases. In the MP2P case, the sender's identity can be extracted from the key identifier and there are considered to be enough key identifiers to allow an arbitrary number of senders on the LSP. There will, however, be the need for the receiver to hold OE state (keys, packet counters) for each sender which may be a significant amount of data for an MP2P LSP (although no more than if the same LSP were replaced by multiple P2P LSPs). Additionally, it should be noted that not only will each sender on an MP2P LSP have a different key, but each may separately decide whether to encrypt data or not.

At this time it is not certain whether MPLS OE can be applied to a point-to-multipoint (P2MP) or a multipoint-to-multipoint LSP in its entirety because packet replication cannot handle the necessary key conversions for each receiver. However, MPLS OE can certainly be applied to individual hops on these LSPs. Further work is needed to determine whether non-branching multi-hop segments of P2MP and MP2P LSPs can also be protected using MPLS OE.

6. Security Considerations

6.1. Security Improvements

See section 2.1.

6.2. Continued Vulnerabilities

The mechanisms described in this document do not provide protection

against certain types of MITM attacks. For example, the key exchange protocol in Section 4.2 will not detect if key exchange messages or their responses are intercepted and discarded such that the initiating peer believes that encryption is not supported. Similarly, those messages may be tampered with such that a receiver cannot determine the correct table index to algorithm and key mapping when an encrypted packet is received. Furthermore, the OEL in an MPLS packet is not protected and may be overwritten such that a receiver is unable to decrypt the packet.

See Section 7.1 for a discussion of how active MITM attacks can be detected.

6.3. New Security Considerations

If a pair of LSRs do not do the key exchange before sending any data packets on the LSP then those first packets will not be protected by

OE and hence will be available to a monitor.

If a MITM can prevent the OE key exchange from completing, e.g. via deleting messages or changing bits in messages, and if the LSRs continue to send data regardless then those data packets will be available to a monitor. See Section 2.4 and Section 7 for a description of how alarms should be raised in these circumstances.

7. Manageability Considerations

As described in Section 2.4 node-wide and per-LSP behavior SHOULD be configurable to describe the action where key agreement exchange or packet decryption fails. In any case, such events MUST trigger alarms to the operator.

7.1. MITM Detection

Section 2.4 introduces the concept of a function of the shared secret that can be compared by two LSRs that are using OE to see whether they are victims of an active MITM attack.

Section 4.2 describes how a witness value is derived for the default KDF, HKDF.

The participating LSRs can simply log this value plus the LSP

and LSR IDs from time to time and a management application can compare the values. If they are different for the same LSP ID, then an active MITM attack has taken place.

It needs to be carefully noted that the management channel used to log or otherwise compare the witness values from the two LSRs MUST be secure. It is likely that routers use relatively high security management channels for configuration and other management operations.

[[Editor note - please see the note in 4.2 about generic MITM-detection. Changes there could affect what needs to be done here.]]

8. IANA Considerations

8.1. Opportunistic Encryption Label Indicator

IANA maintains a registry called "Multiprotocol Label Switching Architecture (MPLS) Label Values" with a single sub-registry called "Label Values". This registry is to be renamed "Special Purpose MPLS Label Values according to [ID.ietf-mpls-special-purpose-labels].

IANA is requested to assign a value from this registry as follows:

Value	Description	Reference
TBD1	Opportunistic Encryption Label (OEL)	[This.ID]

The value 8 is suggested.

[RFC Editor is requested to replace the string "TBD1" with the value assigned by IANA throughout this document, and to remove this note.]

8.2. Pseudowire Associated Channel Types

IANA maintains a registry called "Pseudowire Name Spaces (PWE3)" with a sub-registry called "Pseudowire Associated Channel Types". IANA is requested to assign a value as follows:

Value	Description	Reference
TBD2	Opportunistic Key Exchange Protocol for MPLS	[This.ID]

The value 19 is suggested.

[RFC Editor is requested to replace the string "TBD2" with the values assigned by IANA throughout this document, and to remove this note.]

8.3. Key Derivation Functions and Symmetric Algorithms

IANA is requested to create a new MPLS registry called the "MPLS Opportunistic Encryption Algorithms Registry". New values are to be assigned through "IETF Review" as defined in [RFC5226].

The available range is 0 - 255.

IANA is requested to record the following information and create an initial entry as follows:

Value	Key Derivation Function	Symmetric Algorithm	Reference
0	HKDF	AEAD_AES_GCM_128	[This.I-D]
1-255	Unassigned		

9. Acknowledgements

Many thanks to Alia Atlas for detailed discussion of the implications and mechanisms of MPLS opportunistic encryption. Thanks also to Ron Bonica for encouraging this work, to Sean Turner and Stewart Bryant for early review, and to Jeff Haas and Ross Callon for discussions. Thanks to Andy Malis and Danny McPherson for advice about the use of the Control Word.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[ID.ietf-mpls-special-purpose-labels] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special Purpose MPLS Labels" draft-ietf-mpls-special-purpose-labels, work in progress.

[RFC3526] Kivinen, T., and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC5116] D. McGrew, "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.

10.2. Informative References

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4990] Shiomoto, K., Papneja, R., and R. Rabbat, "Use of Addresses in Generalized Multiprotocol Label Switching (GMPLS) Networks", RFC 4990, September 2007.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

Authors' Addresses

Adrian Farrel
Juniper Networks

EMail: adrian@olddog.co.uk

Stephen Farrell
Trinity College Dublin
Dublin, 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie