

KARP
Internet-Draft
Intended status: Standards Track
Expires: January 31, 2013

W. Atwood
R. Bangalore Somanatha
Concordia University/CSE
July 30, 2012

Automatic Key and Adjacency Management for Routing Protocols
draft-atwood-karp-akam-rp-02

Abstract

When tightening the security of the core routing infrastructure, two steps are necessary. The first is to secure the routing protocols' packets on the wire. The second is to ensure that the keying material for the routing protocol exchanges is distributed only to the appropriate routers. This document specifies requirements on that distribution and proposes the use of a set of protocols to achieve those requirements.

Status of this Memo

This Internet-Draft is submitted to IETF 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."

This Internet-Draft will expire on January 31, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	4
2. Keying Groups (Key Scopes)	4
2.1. Keying Groups	4
2.2. Key Scopes	5
3. Problem Statement	5
3.1. Security Goals	6
3.2. Non-security Goals	6
4. High Level Design	6
4.1. Global View	7
4.2. Entities in the system	7
4.3. Protocol Operations	9
5. Detailed Design	10
5.1. System Design	10
5.1.1. Communication among the Entities	10
5.1.2. Inner View of a GM	12
5.1.3. Hierarchical Design	13
5.2. Protocol Design	13
5.2.1. Step 1 - Initial Exchanges: GCKS, GM mutual authentication	14
5.2.2. Step 2 - Key Management Message Exchanges between GCKS, GM	15
5.2.3. Step 3 - GM-GM mutual authentication	18
5.2.4. Step 4 - Key Management Message Exchanges between GMs	18
5.2.5. Variations for handling other Keying Groups	21
6. Other Aspects of the Key Management Problem	23
6.1. Key Updates	23
6.2. Regular Key Updates	25
6.2.1. Same key for the entire AD	25
6.2.2. Key per link	25
6.2.3. Key per sending router	26
6.2.4. Key per sending router per interface	26
6.2.5. Key per peer	26
6.3. Router Installation/ Uninstallation	26
6.3.1. Same key for the entire AD	27
6.3.2. Key per link	27
6.3.3. Key per sending router	28
6.3.4. Key per sending router per interface	28
6.3.5. Key per peer	28
6.4. Router Reboots	28
6.5. Scalability	31

6.6. Option to Turn Off Adjacency Management	32
6.7. Incremental Deployment	33
6.8. Smooth Key Rollover	33
6.9. Eliminating Single Point of Failure	34
7. Detailed Packet Formats	34
8. IANA Considerations	34
9. Acknowledgements	34
10. Change History (RFC Editor: Delete Before Publishing)	34
11. Needs Work in Next Draft (RFC Editor: Delete Before Publishing)	35
12. References	35
12.1. Normative References	35
12.2. Informative References	35
Authors' Addresses	37

1. Introduction

Within the Keying and Authentication for Routing Protocols working group, there are several goals:

- o Determining how to update the security of existing routing protocols, and guiding this work;
- o Development of automated mechanisms for management of the keying material.

Within the second goal, there is at this time considerable activity on protocols and procedures for creating shared keys, under the assumption that the end points of the exchanges (the routers) are entitled to enter into the conversation, i.e., that they can prove that they are who they say they are. However, there appears to be no work on ensuring that the end points are entitled to be neighbors.

This document addresses this issue. In particular, it addresses the need to ensure that keying material is distributed only to routers that legitimately form part of the "neighbor set" of a particular speaking router.

1.1. Terminology

Autonomous System ...

Administrative Domain ...

Traffic Encryption Key (TEK) ...

2. Keying Groups (Key Scopes)

2.1. Keying Groups

In an AD, all routers having the same TEK can be referred to as forming a 'keying group'. We can have routers forming a 'keying group' as follows:

A group per AD - This is the most coarsely grained category of keying group where all routers in an AD share the same traffic key. Hence the incoming and outgoing keys for protecting control traffic on all routers are the same. This is the case typically in usage today with manual keying.

- A group per link - Here, all routers sharing a link share the key for that link. The routers could have different keys on their different interfaces, and share them with the other routers connected to those respective links.
- A group per sending router - This category is more finely grained compared to the previous two cases; each router uses a different key to secure its outgoing control traffic.
- A group per sending router per interface - This is the most finely grained category wherein each router has a different key for each of its interfaces, which in turn is different from the keys used by other routers to secure their outgoing traffic.
- A group per peer router - This category is strictly for unicast communication wherein peer routers share keys for their interaction. There is one outgoing key corresponding to each router in every pair of routers. These keys can be established through a unicast key management protocol such as IKE [RFC2409] or IKEv2 [RFC5996].

2.2. Key Scopes

Alternatively, keying groups can be viewed from another perspective. Instead of looking at the granularity of keying from the point of view of the members, we can look at it from the point of view of the keys. This can be referred to as 'key scope'.

The key scopes corresponding to the above categories of keying groups in the same order could be defined as follows:

- Same key for the entire AD - all routers in the domain share the same key.
- Key per link - all routers on a link share the same key.
- Key per sending router - each router has a different key to secure its outgoing control traffic.
- Key per sending router per interface - each router uses different keys for each of its interfaces, which in turn are different from the keys used by the other routers for securing their outgoing traffic.
- Key per peer router - there exist two keys corresponding to every pair of routers.

3. Problem Statement

The overall aim of this document is to specify an overall system for automated key management, which will eliminate the disadvantages of the manual method of key updating. The basic function of this automated system is secure generation of keys and their distribution. The system should also enable key updates at regular intervals so as

to protect against both active intruders and passive intruders who could be eavesdropping the traffic after having gained access to the keys secretly.

Along with these basic goals, a key management system should satisfy an additional set of requirements. These requirements ensure among other things, security, easy deployment, robustness and scalability. We have compiled this set after referring to the KARP Design Guide [RFC6518], the KARP Threats and Requirements Guide [I-D.ietf-karp-threats-reqs] and the PIM-SM "security on the wire" specification [RFC5796].

3.1. Security Goals

1. Peer authentication for unicast and authentication of all members of the group for multicast protocols.
2. Message authentication, which includes data origin authentication and message integrity.
3. Protection of the system from replay attacks.
4. Peer liveness.
5. Secrecy of key management messages.
6. Authorization to ensure that only authorized routers get the keys.
7. Adjacency management, which implies ensuring the legitimacy of neighbor relationships of each router. Also providing an option to turn off adjacency management if required.
8. Ensuring Perfect Forward Security (PFS) and Perfect Backward Security (PBS).
9. Resistance to man-in-the-middle attacks.
10. Resistance to DoS attacks.
11. Usage of strong keys; those that are unpredictable and are of sufficient length.

3.2. Non-security Goals

1. Ability to handle various categories of keying groups depending on the security level required.
2. Possibility for easy and incremental deployment.
3. Smooth key rollover.
4. Robustness across router reboots.
5. Scalable design.
6. Single key management architecture accommodating both unicast and multicast systems.

4. High Level Design

In this section, we propose an architecture for an automated key

management and adjacency management system. In order to build this framework, we have reused parts of some existing proposals and fitted them into their correct places in the overall architecture. We have then extended/ modified them so as to handle the key management issues that they appear to have overlooked.

Our design deals with securing the control traffic of routers within an AD.

4.1. Global View

The main entities in our system are the following:

1. Administrator
2. Policy Server
3. GCKS
4. Standby GCKS
5. GMs

These entities and their functions are explained in the next section.

4.2. Entities in the system

The entities are based on those in GSAKMP. The difference is that the Group Owner in GSAKMP has been replaced by a Policy Server, and the Subordinate GC/KS has been replaced by a Standby GCKS in our design. We have chosen the term 'Policy Server' in order to be consistent with RFC 3740 [RFC3740], and the term 'Standby GCKS' since it is not a subordinate in our design and is a standby that is capable of performing all operations performed by the active GCKS. Our design conforms to the Multicast Group Security Architecture [RFC3740].

The network administrator makes configurations for the Policy Server and the GCKS. Security policies go to the policy server, and configurations related to the AD go to the GCKS.

Policy Server is the entity that manages security policies for the AD. The behavior of the policy server we describe here draws contents from and is very similar to the 'Group Owner' in GSAKMP. The security policies include general policies such as authorization details for the GCKS, access control for the GMs, rekey intervals, as well as other specific policies that may be necessary for the group. These policies are put together into a 'Policy Token' [RFC4535] and sent to the GCKS.

The GCKS is either a router or a server chosen by the administrator as the group controller. It is the entity whose major function is

key management and adjacency management. The GCKS should also ensure that the security policies in the policy token are enforced. This implies that whenever a GM requests keys from the GCKS, the GCKS should enforce access control for the GM according to the terms specified in the policy token. The administrator configures the GCKS with information such as the type of keying group to be enforced for the AD and the adjacencies for each router in the AD corresponding to a particular routing protocol (or a set of similar routing protocols). This last point is due to our proposal that there could be one instance of a GCKS per routing protocol or a set of similar routing protocols. This is in fact necessary because GCKS is the entity that should ensure adjacency management, and adjacencies may be defined differently for different routing protocols. Also, according to [I-D.ietf-karp-ops-model] , "KARP must not permit configuration of an inappropriate key scope". This means that each routing protocol could have a different requirement of key scope and that needs to be satisfied. The GCKS may also generate, distribute and update keys, depending on the type of keying group to be enforced in the AD.

The standby GCKS is an entity that is always kept in sync with the active GCKS, ready to take over at any time should the active fail. This design eliminates the possibility of a single point of failure in a centralized system.

GMs are the group member routers that communicate with each other as well as with the GCKS. When they request keys from the GCKS, they are given the keys along with the policy token. GMs are required to check the rules specified in the policy token to determine if the GCKS is authorized to act in that role. Each GM has a Local Key Server (LKS) [atwo2009:AKM]. It is a key generation and storage entity within the GM. A GM may sometimes be required to generate keys itself depending on the category of keying group being enforced. This kind of design ensures that the architecture is distributed in the sense that key management responsibility is divided between the GCKS and the LKSes.

From the description above, it can be seen that the architecture we propose is a balance between a completely centralized model and a completely distributed one, developed by picking the plus points of both types. It defines the concept of a GCKS, which is a centralized entity, as well as the concept of a LKS, which is distributed as being one entity per router. The design tries to bring in the advantages of both models. A centralized entity is considered necessary mainly to make adjacency management possible. In the absence of a central controller that has information about the adjacencies of each router in the AD, individual routers will not be able to establish the legitimacy of their neighbors. Adjacency

management is especially important since we are dealing with control packets, which are usually exchanged with immediate neighbors. At the same time, loading the centralized entity with multiple responsibilities may lead to its failure. Hence we have a localized entity that can take up some of the functions of the central controller as and when the need arises. This enhances scalability, which is so important in a key management system. Another factor leading to scalability is the presence of the standby GCKS. A centralized system could have the disadvantage of having a single point of failure. Our design tries to eliminate this by defining a standby for the central controller that is always kept in sync with it, ready to take over at any time.

4.3. Protocol Operations

The operations of key management and adjacency management occur at two different levels. To ensure scalability of the system, as many operations as possible need to take place among adjacent routers. However, to ensure overall control, policies need to be set centrally for the entire AD.

We recognize two types of groups, which represent the two levels of operation:

- o a group consisting of the GCKS and all the routers (called group members or GMs);
- o many small groups, each consisting of a set of adjacent routers.

The overall operation proceeds in four steps:

1. Establishment of a secure path between each GM and the GCKS.
2. Exchange of policy information between each GM and the GCKS. This policy information defines the key management approach and parameters and the adjacency management approach and parameters.
3. Establishment of a secure path between pairs of adjacent GMs, where the legitimacy of the adjacency was established in step 2;
4. (if required) Exchange or generation of the shared key (and other security parameters) that will be used to protect the routing protocol packets.

If the key scope corresponds to "same key for the entire AD", then the key management policy in step 2 could be "use this key", where "this key" is the same for all GMs, and is sent as a parameter along with the policy. In this case, the key generation in step 4 is not necessary.

If the key scope corresponds to "key per link", then the key may be mutually determined by the routers on that link, or a "local" GCKS

may be elected and assume the task of generating the key, which will then be distributed on the secure paths established in step 3.

If the key scope corresponds to "key per sending router" or "key per sending router per interface", then the sending router assumes the responsibility for generating and distributing the key(s) that it will use to send its routing protocol traffic. In the first case, each router maintains $(n+1)$ keys, one for each neighbor, for incoming traffic from that neighbor, and one key for outgoing traffic. In the second case, each router maintains $(n+k)$ keys, where "k" is the number of interfaces.

Similarly, if the key scope corresponds to "same key for the entire AD", then the adjacency management policy is probably "accept any router that claims to be your neighbor" or "accept any router that presents a valid router identification string".

For other key scopes, the authentication part of step 3 will have to confirm that a match exists between what is presented by the neighbor router and what is specified in the adjacency management policy information.

If IPsec is to be used to protect the routing protocol packets, negotiation of the Security Parameter Index (SPI) to be used will be done as part of step 4. This has to be mutually negotiated among the users of a particular key, because it cannot be arbitrarily set by any particular member of the group of adjacent routers. (This is in contrast with a two-party Security Association, where the SPI can be safely set by the (single) receiver of the incoming packets.) However, in the case where a single key is being used for the entire AD, the SPI may be dictated by the GCKS

5. Detailed Design

This section provides a detailed description of the automated key and adjacency management system. This is followed by the details of the communication among the various entities of the system.

5.1. System Design

This section provides a detailed description of the architecture, showing also the communication among the different entities.

5.1.1. Communication among the Entities

Figure 1 gives a closer view of the entities in our design as described previously and shows the interactions among them.

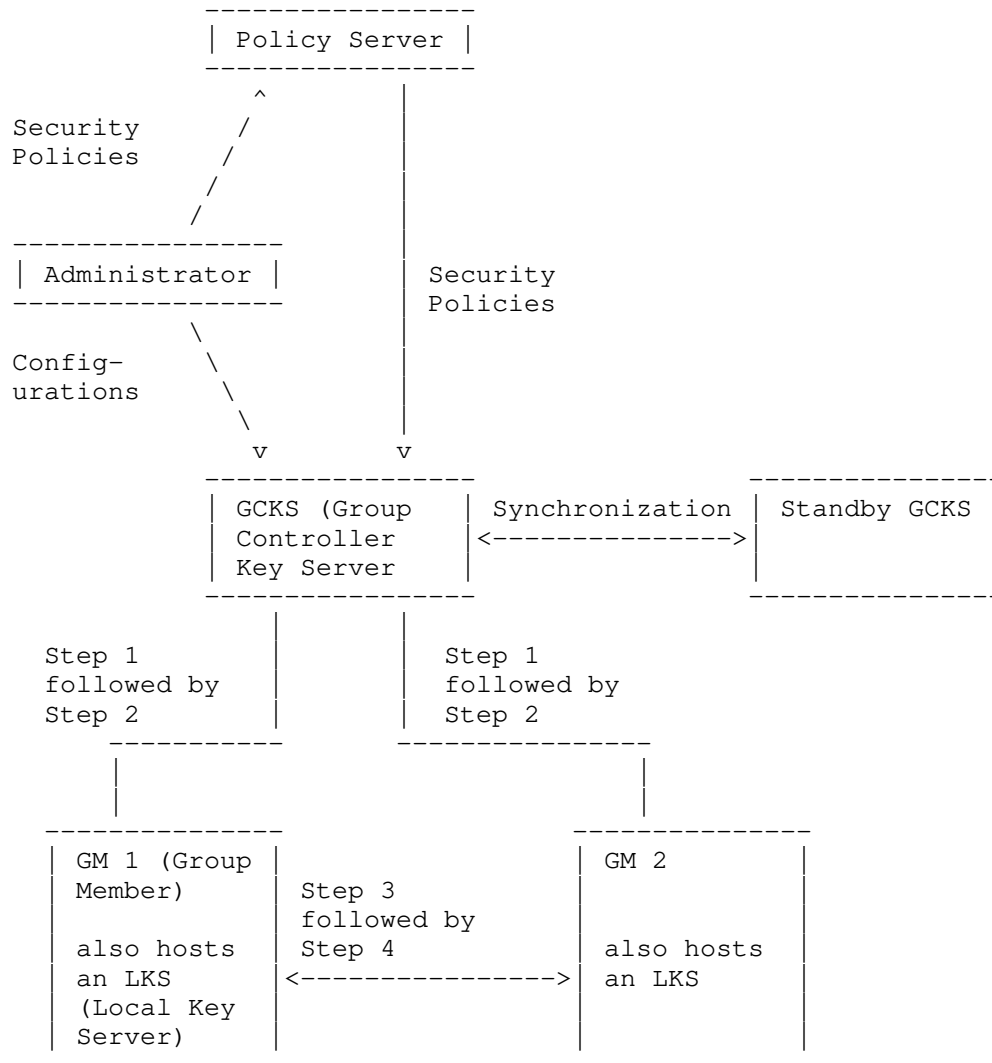


Figure 1: Communication between the entities

Basically there is a centralized GCKS in the system and localized LKS, local to each GM router. The GCKS and the LKS have the ability to generate SA parameters through a KMP, and to store them in a key store. The different scenarios to be considered and the steps of communication are described in this section and the next.

5.1.2. Inner View of a GM

Figure 2 shows an inner view of a GM with interactions among the KMP, a routing protocol and the LKS.

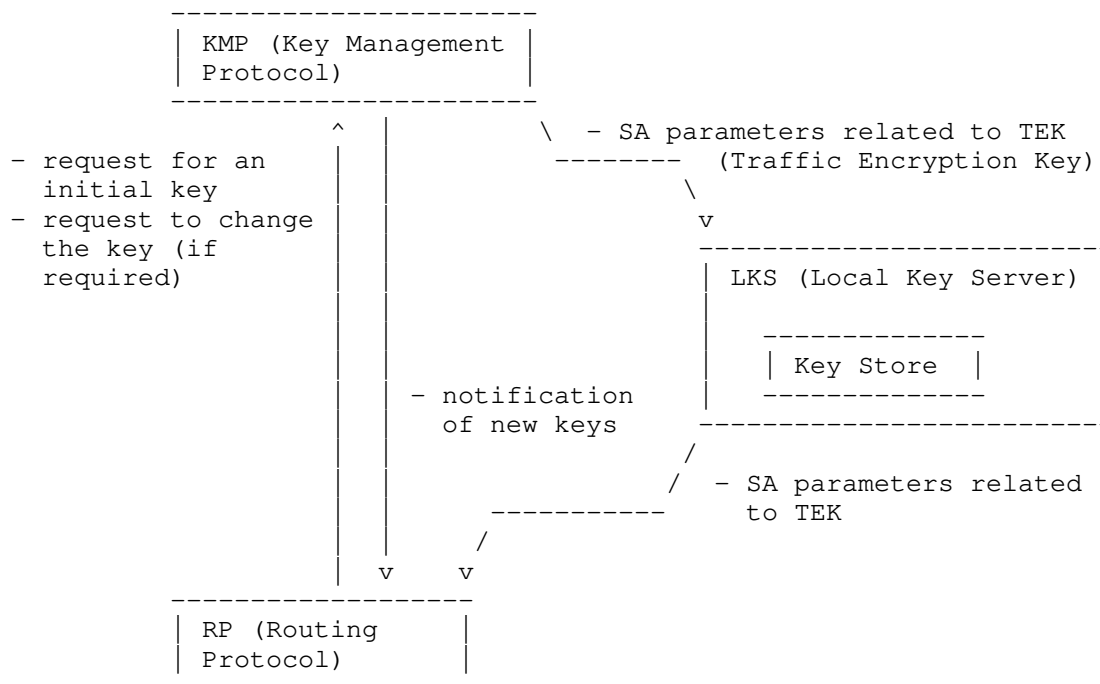


Figure 2: Inside view of a GM

Initially the routing protocol requests keys from the KMP to secure its control traffic. This starts the communication between the GM and the GCKS through the KMP, as shown by the numbered steps in Figure 1. The key generation policy specified by the GCKS is transferred to the GM. Then the keys are generated by the LKS of the GM, and stored into a key store hosted by the LKS. The KMP notifies the routing protocol that new keys are available for its use as shown in Figure 2. The routing protocol then retrieves the keys from the key store. For some categories of keying groups, the LKS is given the keys directly by the GCKS. For others, it may negotiate the keys with its neighbors. These cases are explored in detail in the sections that follow.

The proposed KMP runs between the GCKS and the GMs, and among the GMs themselves. The KMP messages need to be protected, and this can be achieved by running a protocol prior to it to derive keys to protect

it. This is similar to the manner in which GDOI messages are protected by keys generated by a phase 1 protocol such as IKE.

5.1.3. Hierarchical Design

The design we propose is a hierarchical one. There are two kinds of groups that can be formed here (not to be confused with keying groups). The first kind is the one formed by the GCKS with each GM in the AD. The second kind is the one formed among the GMs. The design can be seen as comprised of 5 main steps. The steps together help ensure key and adjacency management in a secure manner.

- Step 1 - Mutual authentication between the GCKS and each GM in the AD.
- Step 2 - Communication between the GCKS and each GM in the AD for secure distribution of policies and keys.
- Step 3 - Inter-GM authentication.
- Step 4 - Communication among the GMs themselves for key distribution.
- Step 5 - The actual transfer of routing protocol control packets using the keys derived through the previous four steps.

Each step is dependant on the previous ones leading to a hierarchy and ensuring modularity of design. Our design concentrates on steps 1 through 4 in order to enable a secure step 5.

The details of each of these steps are explained in the next section.

5.2. Protocol Design

In this section, we give a detailed description of our proposal for a protocol that serves as a solution to the key management problem outlined in Section 3. To summarise, the intention is to develop a protocol for an automated key management system such that all the requirements listed in Section 3 are satisfied.

We have seen the set of entities in the proposed design in Section 4. Now we shall see the exact messages exchanged among them so that the keys required for securing routing protocol control traffic can be generated and distributed to the appropriate routers.

Initially the administrator configures security rules on the Policy Server, and configuration parameters on the GCKS. The security rules have among other things, access control rules related to GMs, and authorization rules related to the GCKS. The configuration parameters include among other things, the key scope information pertaining to the AD and adjacency information corresponding to each router in the AD. If required, the Policy Server generates other

security policies relevant to the group and puts them together into a policy token. This policy token is sent to the GCKS.

Once this is done, steps 1, 2, 3 and 4 as outlined in Section 5.1.3 follow. Step 1 is for GCKS-GM authentication, step 2 is for key and/or policy transfer from the GCKS to each GM, step 3 is for GM-GM authentication, and step 4 is for key exchange between GMs that need to communicate with each other. Steps 2 and 4 have small variations depending on the key scope being enforced for the AD.

Steps 1 and 2 are based on the GDOI GROUPKEY-PULL protocol [RFC6407]. However, step 2 in our case is an extension of GROUPKEY-PULL in the sense that it accommodates various cases of keying groups and adjacency management as well. Steps 3 and 4 have been designed such that GROUPKEY-PULL has been extended to inter-GM communication.

Now we shall look at each of these steps in detail.

5.2.1. Step 1 - Initial Exchanges: GCKS, GM mutual authentication

Initially, when a routing protocol instance wishes to start communication, be it unicast or multicast communication, it informs the same to the KMP instance on the router. This information is communicated by the KMP instance from that router to the KMP instance on the router or server it believes to be the GCKS. At this point, the GCKS needs the identity of the requesting router in order to authenticate it. The requesting router also has to authenticate the GCKS. Any of the ISAKMP group of unicast protocols could be used for step 1 communication between the GCKS and each router that requests keys from it. IKE/ IKEv2 is an example of such a protocol. This protocol provides peer authentication, and parameters for an SA including a key to help provide confidentiality and message integrity for the next step where the actual traffic keys would be generated. We call the key derived in this phase as SKEYID_a (term taken from GDOI). It is assumed that the routers have agreed upon a way to establish their identity during authentication, either through pre-shared keys, asymmetric keys or certificates. If peer authentication is successful, the router becomes a GM.

As already mentioned, GM stands for 'Group Member'. When talking about the GCKS-GM interactions, 'group' typically means the entire set of GMs in the AD. When talking about the GM-GM interactions, 'group' typically means the sending router and some set of its neighbors. This set may include all of its neighbors or only a subset, depending on the key scope in use. For example, when the key scope is per link, a 'group' may refer to all routers sharing a link. This will become evident as we see the GM-GM interactions shortly.

5.2.1.1. Message Exchanges for Step 1

The protocol message exchanges for this step are the standard IKE exchanges since we propose using IKE for this step. We would like to mention at this point that whenever we say IKE, we intend to refer to IKE or IKEv2, unless explicitly stated otherwise.

5.2.2. Step 2 - Key Management Message Exchanges between GCKS, GM

This is the step where the KMP takes over. The goal of the KMP is to provide parameters for an SA to be eventually used by a routing protocol to secure its control traffic.

Messages in this step are secured by the key generated by the step 1 protocol, that is, SKEYID_a. This key helps achieve authentication and confidentiality for step 2. For step 2, we have taken most of the messages from GROUPKEY-PULL protocol of GDOI. However, there are some modifications and important addition of functionality in our case, with the GCKS passing additional information to the GMs. We shall see this in this section.

We shall initially look at the KMP details for one of the finely grained cases of keying groups, namely, the group per sending router. This is a flavor of multicast communication. Soon after this we will see the small variations necessary in order to handle the other categories of keying groups.

In step 2, the (each) GM makes requests from the GCKS through the KMP for SA parameters required to secure its control traffic. In the request to the GCKS, the GM specifies the identity of the routing protocol for which it needs the keys. Although the GCKS corresponding to the routing protocol would have already been selected in step 1, specifying the routing protocol id again here helps to handle the case where the same GCKS may be used for a category of similar routing protocols.

When the GCKS receives this request from the GM, it checks to verify if the GM can be given access to key related information according to the rules in the policy token. If the checks fail, the communication with the GM should not be continued. The exact behavior can be determined from the rules in the policy token. If the checks succeed, the GCKS delivers to the GM the following information:

- o SA policy corresponding to the TEK. This could include the actual SA parameters as well depending on the category of keying group being enforced. The TEK is the traffic key whose scope could be anything among those described under key scopes in Section 2. The SA policy includes policy information about SA parameters. This

could include information pertaining to the algorithms, the TEK, the SPI and other parameters. For the category of keying group being discussed now, that is, the key per sending router, the exact TEK and SA parameters are not delivered by the GCKS to the GM. Only rules pertaining to their generation are handed down. The actual SA parameters are generated by the GM itself soon after step 2 so that the GCKS is not overloaded.

- o A certificate signed with the private key of the GCKS. This is to be used by the GM for authentication purposes when it communicates with neighboring GMs and with the GCKS for any SA updates in future.
- o The policy token information received by the GCKS from the Policy Server. As already mentioned, this includes authorization and access control related information. This is read by the GM in order to authorize the GCKS and verify if it is entitled to perform the role of GCKS.
- o The key scope being enforced in the AD. This configuration is made by the administrator on the GCKS and is pushed to the GM. This is necessary so that the GM knows whether to expect the traffic keys from the GCKS, or whether it needs to generate them itself.
- o The adjacency information, which includes details of all legitimate neighbors on all interfaces of the GM and not only the neighbors online at that point of time. This is in order to avoid a DoS attack on the GCKS that could result if the GMs started querying the GCKS for every router coming up, especially during the boot up sequence, to know if it is a legitimate neighbor. Also, this ensures completeness of information. It even helps eliminate spoofing attacks where a legitimate neighbor may appear on an interface other than the one it was supposed to appear on. The adjacency information is used by the GM to know the set of authorized neighbors with which it should communicate during steps 3 and 4.

5.2.2.1. Message Exchanges for Step 2

The protocol message exchanges for step 2 are shown in Figure 3.

```
GM->GCKS: HDR*, HASH(1), Ni, RP_ID (1)
GCKS->GM: HDR*, HASH(2), Nr, SA, CERT, K_SCOPE, PT, ADJ (2)
GM->GCKS: HDR*, HASH(3) (3)
```

Figure 3: Message exchanges for Step 2

In the message exchanges, HDR is an ISAKMP header payload. It has a message id M-ID. The '*' indicates that the message contents following the header are encrypted. The encryption is done with SKEYID_a. This ensures authentication (since the key is a secret

generated in step 1 and can be possessed only by the GCKS and the GM with which the step 1 has been carried out) as well as secrecy (due to the encryption). Hashes are used for ensuring message integrity and data origin authentication; this will be explained shortly.

In exchange (1), the GM requests SA information from the GCKS to protect its control traffic corresponding to the routing protocol whose id is given by RP_ID. Ni is a nonce used to protect against replay attacks as well as to ensure liveness of the GM.

In exchange (2), the GCKS initially confirms from the rules in the policy token that the GM can be given SA information. It also verifies the freshness of the nonce Ni. If this is successful, the GCKS proceeds to deliver to the GM the following information:

- o SA policy corresponding to the TEK - through the parameter SA
- o A signed certificate - CERT
- o Key Scope - K_SCOPE
- o Policy token - PT
- o Adjacency information - ADJ

The details of these pieces of information have already been explained. Nr is a nonce used for replay protection and to ensure liveness of the GCKS.

In exchange (3), the GM initially verifies freshness of the nonce Nr so as to detect a replay attack. It then proceeds to confirm the authorization of the GCKS by referring to the policy token. If the GCKS is an authorized entity, the GM uses the key scope information to know how to proceed with respect to key generation. The adjacency list is used to note the list of legitimate neighbors and the allowed interfaces on which they can appear online. Once this is done, the GM sends an acknowledgement. This acknowledgement includes a hash for integrity purposes. If the GCKS is not authorized, the GM needs to end the communication with the GCKS. The behavior in such cases can be determined by the policies specified in the policy token.

The hashes are pseudorandom functions (prf) computed as shown in Figure 4.

```

HASH(1) = prf(SKEYID_a, M-ID | Ni | RP_ID)
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | Nr | SA | CERT | K_SCOPE |
              PT | ADJ)
HASH(3) = prf(SKEYID_a, M-ID | Ni_b | Nr_b)

```

Figure 4: Hashes used in Step 2

According to [RFC6407], "Each HASH calculation is a pseudo-random

function ("prf") over the message ID (M-ID) from the ISAKMP header concatenated with the entire message that follows the hash including all payload headers, but excluding any padding added for encryption." SKEYID_a is included in the hashes to ensure that both parties have the step 1 key. The hashes include the nonces from previous messages to ensure that both the parties have the exchanged nonces. This is used for data origin authentication purposes. Hence Ni_b and Nr_b refer to Ni and Nr from exchanges (1) and (2) respectively.

An important function of hashes is to provide message integrity. The receiver computes the hash of the received message and compares it with the hash value received to determine whether the message has been tampered with or not.

Once the GM has received this information, it generates the TEK and determines the parameters to be used for its outgoing SA. Here the functionality of the LKS of the GM as a generator of keys comes into play. Since the key scope being discussed now is one key per sending router, the LKS of each GM generates one TEK. The key generation is to be followed by key information exchange with legitimate neighbors so that the incoming SAs can be determined. It is to be noted that this key generation can even be done at the beginning of step 4 once the inter-GM mutual authentication has happened in step 3.

5.2.3. Step 3 - GM-GM mutual authentication

After the GM generates TEK based information, before exchanging it with its neighbors, it needs to ensure that a secure TEK exchange can take place. This is done in step 3 by each GM engaging in a unicast communication with each of its legitimate neighbors through any of the ISAKMP group of unicast key management protocols, such as IKE. This protocol provides peer authentication as well as a secret key to provide confidentiality, authentication and message integrity for step 4, which is the actual TEK exchange step. We call this secret key as SKEYID_b. The legitimate neighbors are determined by referring to the adjacency information given by the GCKS to the GM in step 2. During peer authentication in step 3, the certificate given to the GM by the GCKS could be used.

5.2.3.1. Message Exchanges for Step 3

The protocol message exchanges for this step are the standard IKE exchanges since we propose using IKE for this step.

5.2.4. Step 4 - Key Management Message Exchanges between GMs

This is the step where the TEK information is exchanged between GMs that need to communicate with each other. Unicast communication is

anyway between two peers. For multicast communication, since we are dealing with control traffic only, and control traffic is typically link-local, each router on a link needs to be aware of the TEK of all other routers on the same link. These legitimate neighbors are determined from the adjacency information received from the GCKS. The LKS of the corresponding GMs communicate to exchange their TEK information in order to help them populate their incoming and outgoing SAs.

Messages in this step are secured by the key generated by the step 3 protocol, that is, SKEYID_b. This key helps provide authentication as well as confidentiality.

In step 4, the LKS of the GM pushes the SA information corresponding to its TEK to each of its neighbors. The LKS also requests TEK information from its neighbors. Each of the neighbors then sends its outgoing TEK information and this is maintained as an incoming key on the querying LKS. As a result of step 4, all GMs have the TEK information corresponding to all their neighbors so that a secure control traffic exchange can start.

5.2.4.1. Message Exchanges for Step 4

The message exchanges for Step 4 are shown in Figure 5.

```
GMi->GMr: HDR*, HASH(4), N1, CERT1      (4)
GMr->GMi: HDR*, HASH(5), N2, CERT2      (5)
GMi->GMr: HDR*, HASH(6), SA1, KD1, KREQ (6)
GMr->GMi: HDR*, HASH(7), SA2, KD2      (7)
```

Figure 5: Message exchanges for Step 4

GMi and GMr depict the initiator and the responder GMs respectively.

The message exchanges in this step are similar to those in step 2 in that the HDR is an ISAKMP header payload with a message id M-ID. The '*' indicates that the message contents following the header are encrypted. The encryption is now done with the key SKEYID_b derived in step 3. This ensures both authentication and secrecy. Hashes are used for ensuring message integrity and data origin authentication. Nonces are used to resist replay attacks and to ensure peer liveness.

In exchanges (4) and (5), we show mutual authentication between GMs through the certificates received from the GCKS in step 2. CERT1 is the certificate received by GMi and CERT2 is the one received by GMr from the GCKS. Authentication would have happened in step 3 so exchanges (4) and (5) can be eliminated. They have been shown here for the sake of completeness.

In exchange (6), the initiator GM communicates to its neighbor its outgoing SA parameters in SA1 as well as the outgoing TEK information explicitly in KD1. This is the TEK that it will be using henceforth to secure its control packets. It also requests the outgoing SA information from the neighboring GM so that it can be installed as incoming SA information on the querying GM. This request is represented by KREQ, which stands for Key Request.

In exchange (7), the neighboring GM responds with its outgoing SA information in SA2 as well as the TEK in KD2. This will be the TEK the neighboring GM will use henceforth to secure its control packets.

As already mentioned, the nonces N1 and N2 help provide replay protection and a confirmation that the peer is alive.

The hashes are pseudorandom functions computed as shown in Figure 6.

HASH(4) = prf(SKEYID_b, M-ID		N1		CERT1)	
HASH(5) = prf(SKEYID_b, M-ID		N1_b		N2	CERT2)
HASH(6) = prf(SKEYID_b, M-ID		N1_b		N2_b	SA1 KD1 KREQ)
HASH(7) = prf(SKEYID_b, M-ID		N1_b		N2_b	SA2 KD2)

Figure 6: Hashes used in Step 4

Hash computation is similar to that explained in step 2. In step 4 hashes are computed by applying a pseudorandom function to the key SKEYID_b, along with the message id concatenated with the message contents following the hash. Also, nonces from a message exchange are included in the hash computation of the subsequent exchanges in order to ensure that both parties have the nonces just exchanged. This helps in data origin authentication. Hence N1_b and N2_b refer to N1 and N2 in exchanges (4) and (5) respectively. Hashes are very essential to ensure message integrity and to confirm that the messages have not been modified (possibly by an intruder) during transit.

All information received by the LKS of a GM from the GCKS as well as from neighboring LKSes is written to stable storage persistent across reboots. This can be effectively used to avoid flooding the GCKS with requests on a router reboot. This is one of the advantages of the proposed design over GDOI [RFC6407], where, when routers reboot they come back up with no information and the GCKS is flooded with requests. The routing protocol is notified by the KMP about the new SA being available in the key table for it to protect its control traffic.

The routing protocol security mechanism would store the incoming and outgoing SA information, and the adjacency information into the

relevant databases.

As we can see, confidentiality and authentication has been ensured for all steps by means of secret keys and certificates.

In the following section, we shall see the small variations required in the basic protocol design proposed above, in order to handle the various categories of keying groups.

5.2.5. Variations for handling other Keying Groups

We have seen the different granularities possible for a keying group, that is, the different key scopes, in Section 2. We have also seen that the design proposed in Section 5.2 is able to handle the keying group where there is a separate key per sending router. This has been achieved by each router generating its own key, which would be the same for all its interfaces. Hence each router has a different SA for outgoing traffic and multiple SAs for incoming traffic, one corresponding to each neighbor. It is to be noted here that the key generation being done locally could have a small possibility of two routers ending up with the same key when they generate it randomly. However, if a good random number generator is used for key generation, the probability of ending up with the same key is drastically reduced. This extremely small possibility can be ignored since the method more importantly has the advantages that it reduces the load on the GCKS. Also the GCKS does not have the need to be aware of the individual keys of each router. This could be considered as a case of tradeoff.

In this section, we shall see how the remaining cases of keying groups can be handled. They can actually be handled by minor variations to the basic design. In essence, these variations can be implemented by the GM interpreting the key scope information given to it by the GCKS in step 2, and thereby knowing whether to expect keys from the GCKS or to derive them itself. This also makes the GM aware of the path to be followed. As we shall see, in a majority of cases it is step 4 that gets slightly altered.

Same key for the entire AD - Let us take the most coarsely grained case, namely, a keying group per AD. Since all routers have to share the same key (TEK), the centralized GCKS is the one that should generate it. Every GM gets the TEK and other SA parameters directly from the GCKS in step 2. The TEK information received from the GCKS can be stored as both the outgoing as well as the incoming key since all GMs share the same key. Therefore, step 4 can be eliminated. However, step 3, which involves GMs authenticating neighboring GMs is necessary before the GMs can start exchanging control packets.

In essence, this variation of key scope can be implemented by the GM interpreting the key scope information given to it from the GCKS in step 2, and thereby knowing that it should expect the TEK from the GCKS (TEK is also received in the same step).

Key per link - This is another flavor of keying groups wherein there exists a TEK per link, that is, a key is shared by all routers sharing a link. This can be handled in a manner similar to the single key per router case described as far as steps 1, 2 and 3 are concerned. However, there is a slight variation required in step 4. Previously, the LKS of each GM generated a single key to be used on all interfaces of the GM. However in this case, an LKS needs to generate as many TEKs as the number of its interfaces by interacting with the neighbors on the respective links. This is done by GMs on a link interacting to derive a TEK and other SA parameters through any of the mutual key agreement protocols. Some examples of protocols that could be used for this purpose are MRKMP [I-D.hartman-karp-mrkmp], group Diffie-Hellman, and the STS protocol. Since MRKMP specifies how keys can be generated and distributed on a LAN by electing a GCKS, it can be used for TEK generation for the case where the key scope is per link. The TEK and the other SA parameters generated are stored by all LKSes sharing the link as the outgoing and incoming parameters on that particular link. This procedure is repeated by all GMs for all their links in turn.

Key per sending router per interface - The only difference here when compared to the separate key per router case is that in that case, each GM generates a single TEK to be used on all of its interfaces, whereas, here each GM generates a different TEK for each of its interfaces. In step 4, it gives each neighbor the TEK that it plans to use on the connecting link between them.

Key per peer - This is the last category of keying groups. This refers to unicast communication where peer routers exchange control packets. Here the SA parameters corresponding to the traffic key TEK and the TEK itself can be generated using a unicast key management protocol such as IKE or even KMPRP. However, an important point to note here is that adjacency management is necessary even for this case since routers should exchange keys only with legitimate neighbors. This can be achieved only by having a central authority that is aware of all valid adjacencies. Our design handles this. Steps 1, 2 and 3 of the design are sufficient. The key derived in step 3, namely, SKEYID_b serves as the TEK.

We have mentioned that the SA parameters along with the TEK are either delivered to the GMs by the GCKS (for the single key per AD case) or generated by the GMs themselves, possibly through

interactions with other GMs (for the other keying groups, depending on the particular category). A parameter that could have a slightly different behavior is the SPI. This is also one of the parameters of an SA. However the range of SPIs to be used in an AD could be decided by the administrator. Whatever be the category of keying group, it could so happen that the administrator chooses to have the same SPI for all GMs. In this case, the GCKS could deliver the SPI to the GMs along with the policy for the remaining parameters of the SA. It could also be that the administrator wants each GM to use a different SPI for its outgoing traffic. In this case, the GCKS should not be overloaded with the task of generating a different SPI for each GM. GMs should generate the SPI themselves, possibly with communication with other GMs. If that happens, even for the single key per AD category of keying groups, the SPI is generated by the GMs, although the TEK may be obtained from the GCKS (since the TEK is to be the same for all GMs for this category of key scope). In other words, the key scope may be different from the scope of the SPI used in the AD. Our design is flexible enough to handle this since the SA policy handed down by the GCKS to the GMs would indicate to the GM the exact steps to be followed.

In all cases of keying groups, the LKS stores SA information to persistent storage to be used across reboots. Keys are stored into the key table [I-D.ietf-karp-crypto-key-table] and the KMP informs the same to the routing protocol, which would start using the keys to secure its control traffic. This is the step 5 mentioned in the explanation of the concept of hierarchical design in Section 5.1.3.

6. Other Aspects of the Key Management Problem

In this section, we address some of the other important aspects of the key management problem. Firstly we show how this automated system allows key updates to be done as frequently as desired. Soon after that, we show how various good-to-have features have been incorporated in the proposed design. Some of these features are scalability, incremental deployment ability, effective handling of router reboots and smooth key rollover. Addition of these features would help in achieving the requirements stated in Section 3.

6.1. Key Updates

Keys used by the routing protocols to secure their traffic need to be updated at regular intervals. They may have to be updated at other non-specific times as well depending on the requirement. There are a couple of reasons why key updates are required:

- o As a good practice in order to protect against passive intruders who could have obtained access to the keys and could be eavesdropping the traffic.
- o Whenever a new member comes up on a link, in order to ensure PBS. This means that the new member should not be able to get access to keys currently being used on the link since that could mean that the member can comprehend old messages exchanged on the link when it was not part of it.
- o Whenever a member leaves, in order to ensure PFS. This means that going forward, even if the old member manages to get hold of messages exchanged among the remaining members on the same link, it should not be able to comprehend them.

One of the important points to be noted here is that PFS and PBS can be achieved very easily and in a straight forward way for unicast communication. Unicast communication involves a pair of routers that share keys for securing their traffic. Every pair of routers derives its own set of keys and those keys are known only to that particular pair of routers. Hence a change in any one of the members of the pair of routers would mean that the old keys are no longer valid and new keys are derived for communication. This automatically takes care of PFS and PBS. When a router, say R1, is uninstalled, the keys used by the other routers for pairwise (unicast) communication with R1 are no longer used. This ensures PFS. When a new router, say R2, is installed, all routers engaging in a unicast communication with it derive new pairwise keys with it. This ensures PBS.

For multicast communication, key updates are essential on a router uninstallation or an installation to ensure PFS and PBS respectively. This is because in multicast communication, multiple routers share the same key and a key remains valid even if one of the routers involved in the communication is changed. To achieve PFS and PBS, keys have to be updated so that the leaving or entering routers do not have access to information they are not entitled to.

We now have to determine what are the keys that need to be updated. For regular updates, it is quite obvious that the traffic keys of all the routers would have to be changed. The other case to consider is when the routers in an AD change, either due to an installation or an uninstallation. It is interesting to note that when the same traffic key is used for the entire AD, that key should be changed, leading to the effect of changing the keys for all the routers. However, for all other key scopes, only the keys corresponding to the neighbors of the leaving/ entering router need to be changed. This is because as far as control traffic is concerned, routers have knowledge of the keys of their neighbors only. Of course the adjacencies and hence the neighbors, may be defined differently for the various routing protocols.

One of the major problems with the manual method of key management is that keys cannot be updated as frequently as desired. This is due to the lack of authorized people to carry out the task. This issue can be easily overcome by an automated key management system. Let us see how these two cases of regular rekey and a rekey on a router installation/ uninstallation can be handled by the automated key management system we propose.

6.2. Regular Key Updates

In this section, we discuss how our design for automated key management aids key updates at regular intervals. The interval at which key updates are to be done is determined from the policies handed down by the Policy Server entity described in Section 4.2. These policies are handed down by the Policy Server to the GCKS in the form of a policy token, which in turn is handed down by the GCKS to the GMs in Step 2 of the protocol as explained in Section 5.2. We now need to see how key updates for all variations of keying groups can be addressed. As we shall see, when all routers in the AD share the same traffic key, the centralized GCKS is the generator of the new key, whereas in all other cases, the GMs generate the new keys appropriately. This is in fact similar to the process of initial key generation described in Section 5.2.

6.2.1. Same key for the entire AD

First, let us take the case of having a single key for the entire AD. Here, when a rekey is required, the GCKS generates the new traffic key and unicasts it to each individual GM. This ensures that all GMs share the same new TEK after the rekey. As an alternative to transferring the new TEK through unicast communication, the GCKS and all GMs in the AD could share a key called a 'TEK Encryption Key'. This key could be used by the GCKS for encrypting the new TEK derived, and multicasting to all GMs. The advantage of this approach over the unicast method is that it eliminates the need to have multiple key update messages sent out by the GCKS, one corresponding to each GM. This in turn reduces the network traffic. However, the downside to the multicast approach is the overhead of maintaining a group key (and appropriately updating it) just for the rekey purposes. This is a case of tradeoff.

6.2.2. Key per link

In this category of keying group, routers sharing a link also share the traffic key for that link. Here when a TEK update is required, GMs on a link execute one of the key agreement protocols such as MRKMP, group Diffie-Hellman or the STS protocol to derive a new TEK. This is similar to the manner in which they interact to derive the

initial TEK for the link. The interval after which the TEK should be changed is of course determined from the policy token.

6.2.3. Key per sending router

In this case, every router has a different TEK that it uses for securing its control traffic. When a rekey is required, each GM generates a new TEK individually and then communicates the same to all its neighbors. The neighbors update the incoming TEK information corresponding to that router in their databases.

6.2.4. Key per sending router per interface

This case is very similar to the previous one. The only difference is that here, each GM generates as many new TEKs as the number of its interfaces, one per interface. The GM then communicates to each of its neighbors the TEK it plans to use on the interface corresponding to that particular neighbor.

6.2.5. Key per peer

This is the unicast case. Keys can be updated just by every pair of routers executing a unicast key management protocol such as IKE.

In all the above cases, the LKS updates the key store as well as its persistent storage with the updated key information. The KMP notifies the routing protocol of a change in the keys used to secure the control traffic.

6.3. Router Installation/ Uninstallation

Along with the regular key updates, keys need to be updated even when an existing router is uninstalled or a new router is installed. These are for PFS and PBS purposes respectively as already explained in Section 6.1. There are a couple of differences between key updates in these cases when compared with the regular key updates.

- o Regular traffic key updates require that the traffic keys corresponding to all routers in the AD be updated. However, key updates on a router removal or addition require only the keys corresponding to the neighbors of the leaving or entering router to be changed. This is because routers have knowledge of the keys corresponding to their neighbors only as far as control traffic is concerned. But if it so happens that the same traffic key is being used for all routers in the AD, then a change in the key automatically implies that the key gets changed for all the routers.

- o Regular key updates are done at intervals determined from the policy token given by the Policy Server. However, key updates on a router removal or addition are done based on instructions given by the GCKS in such a situation. This is because routers in the AD (other than the GCKS) would not be aware of the fact that a particular router is either installed or uninstalled.

Apart from these differences, the process of key updates during a router change is very similar to the regular key updates. We shall now discuss briefly how key updates on a router change can be handled for each of the categories of keying groups.

6.3.1. Same key for the entire AD

For this category of key scope, the same traffic key is shared by all routers in the AD. When a router is removed or a new router is installed, the GCKS derives a new TEK and unicasts it to each of the routers in the AD.

As an alternative to transferring the new key through unicast method, the GCKS and all GMs could share a key called the 'TEK Encryption Key'. If this option is followed, first of all, the TEK Encryption Key would have to be changed on a router change. Then for the case of router installation, the GCKS multicasts the new TEK Encryption Key, encrypted in the old key to all existing routers. It then unicasts the new TEK Encryption Key to the newly installed router. After this, the GCKS derives a new TEK and multicasts it to all the routers after encrypting it in the new TEK Encryption Key. This can be decoded by the new router as well since it now possesses the latest TEK Encryption Key. For the case of router uninstallation, the GCKS changes the TEK Encryption Key and unicasts it to all the remaining routers. The new TEK Encryption Key cannot be multicast in this case since the old router would also be able to decrypt it. Changing of the TEK would be the same as for router installation. The new TEK is sent in a multicast message to all routers encrypted in the new TEK Encryption Key.

When compared with the unicast method of key updates, this multicast method has the advantage of low bandwidth consumption. However the disadvantage of the multicast method is that an extra key, the TEK Encryption Key, now needs to be maintained and updated accurately. So the exact method chosen depends on the administrator.

6.3.2. Key per link

For this case, on a router installation or an uninstallation, the GCKS informs the neighbors of that router. These routers interact with each other (and with the new router if it is a case of router

installation) and derive a new traffic key for that particular link where the neighbor change has occurred. Any of the mutual key agreement protocols such as MRKMP, group Diffie-Hellman or the STS protocol can be used.

6.3.3. Key per sending router

Here again the GCKS appropriately informs the neighbors of the affected router. Each such neighbor runs a randomized key generation algorithm to derive a new traffic key and communicates the key to its neighbors. This is very similar to the case of regular key updates.

6.3.4. Key per sending router per interface

This category of keying group can also be handled in an easy manner. The GCKS informs the neighbors of the affected router. Each such router derives a new traffic key for that interface on which the neighbor change has occurred. The router then communicates the new key to its new set of neighbors on that particular interface.

6.3.5. Key per peer

As already explained, key updates on a router change are not valid for unicast communication. This is because in unicast communication, a key is shared by only two routers. A router addition or a removal results in a change in a particular pair (or pairs) of routers. Hence new keys are anyway derived to be shared by the new pair. Thus this can be considered as an automatic update of keys without any explicit processing.

6.4. Router Reboots

Router reboots form a very important case to be considered in any design pertaining to networks. Especially in a centralized architecture, care should be taken to prevent the central entity from being stormed with requests when multiple routers happen to reboot almost simultaneously. In our architecture, it is the persistent storage of the distributed LKS that plays a major role on a router reboot. As already seen the LKS of each GM writes to persistent storage some configuration and policy information such as the key scope, adjacencies, SAs, the traffic keys corresponding to itself and its neighbors, certificate received from the GCKS, and the policy token. Hence on a GM reboot, the LKS retrieves information from the persistent storage. This is an extremely important feature since it avoids the GCKS being flooded with requests for information when multiple routers in the AD happen to reboot.

However, information retrieval from the persistent storage may not

always be sufficient. Occasionally a rekey could have happened when a router was down. This could have been either a regular rekey or a rekey due to a router installation or removal. These cases should be dealt with in an appropriate manner so as to ensure that the rebooted router gets the latest SA and adjacency information.

In order to handle these cases, a router needs to query its neighbors on a reboot. This is done as soon as the router has rebooted and read the relevant information from its persistent store. The neighbors communicate their traffic key and SA information to the rebooted router. Depending on this information as well as the key scope information retrieved from the persistent storage, the rebooted router can handle a rekey appropriately. This interaction with the neighbors for the different cases of key scopes is explained below:

Same key for the entire AD - To handle this case, a router gets the TEK related information initially from one of its neighbors. It compares this key with the key corresponding to that neighbor (which is the same as its own key since the same key is shared by all routers in the AD) as retrieved from the persistent storage. If the two keys match, then it is evident that no rekey has happened on the neighbor. Since the key scope is such that the same key is used for the entire AD, it can be concluded that there has been no rekey in the AD. Hence the rebooted router need not do anything else. If the keys are in mismatch, the rebooted router concludes that a rekey has happened in the AD, either due to a regular key update or due to a key update based on a router change. In either case, the router changes its outgoing traffic key to be the same as the new one got from its neighbor. This helps maintain consistency of all traffic keys across the AD.

Key per link - For this case, the rebooted router queries its neighbors in turn, one neighbor on each of its links. Again it compares the traffic key received from its neighbor with the corresponding information retrieved from its persistent store. If the two keys match, it means that there has been no rekey on that link. If the keys are in mismatch, it means that a rekey has happened on the link. The rebooted router then changes its own outgoing traffic key on that link to be the same as the new key got from the neighbor. In either case, the router proceeds with querying its neighbors on its remaining links. This is different from the previous case where a single key was used by all routers in the AD. This is because in the key per link case, determining whether a rekey has happened on a particular link does not help determine the status on other links. Hence at least one neighbor on each link has to be queried.

Key per sending router - For this case, the rebooted router starts by querying one neighbor on each of its interfaces. If the traffic keys of all the queried neighbors are the same as the corresponding keys retrieved from the persistent storage of the rebooted router, there is nothing to be done. If there is at least one neighbor whose key has changed, the rebooted router changes its own key and communicates it to its neighbors. The rebooted router can stop querying its neighbors at this point. An interesting observation here is that a neighbor's key could have changed either due to a regular rekey or due to an installation/ uninstallation of its neighboring router. This neighboring router may or may not be a common neighbor to the rebooted router. Since the exact situation cannot be determined, the rebooted router just goes ahead with its key change once it sees that the key of its neighbor has changed. This should be fine since an extra key update is not harmful.

Key per sending router per interface - This case is similar to the key per link case. The rebooted router queries one neighbor per interface and compares the traffic key information received with the corresponding information from the persistent key store. If the keys match, there has been neither a regular update nor a router change on that interface. If the keys do not match, it means that there has been a key update either as part of a regular rekey or due to a neighbor change on that interface. Hence the rebooted router derives a new traffic key for that interface and communicates the same to its neighbors on that interface. The router then proceeds with querying its neighbors on the remaining interfaces to determine whether the keys used on its remaining interfaces are required to be changed or not.

Key per peer - This category of keying group represents unicast communication. Here when a router comes back up after a reboot, it queries its counterpart for the traffic keys corresponding to this pair of routers. Since for unicast communication, a pair of routers together derives traffic keys, new keys for this pair would not be available as yet even though a regular rekey interval may have passed when the router was down. Therefore the two routers could engage in a unicast key management protocol such as IKE to derive new traffic keys or could decide to proceed with using the old keys itself till the next rekey interval has passed.

The method described above helps ensure that in a majority of cases, rekeys that could have happened when a router was down are handled. There are a couple of cases to be considered as yet.

Firstly, the rebooted router should verify whether the adjacencies as

retrieved from its persistent storage are accurate still. They could now be stale due to the fact that a router could have been installed/uninstalled when it was rebooting.

Secondly, in the discussion above regarding the ways in which reboots can be handled for the different categories of keying groups, we have mentioned that a router queries only one neighbor in some cases and one neighbor per link or interface in other cases. A situation could arise wherein the queried neighbor itself had gone through a reboot resulting in its own key being stale. This in turn would mean that the querying router cannot rely on the information got from this single neighbor.

One way in which both of these issues could be addressed is for the rebooted router to query the GCKS to get the updated information. However we do not want the GCKS to be flooded with requests from the various routers in the AD. Hence there are two layers of protection designed as follows:

- o As already explained, the rebooted router retrieves information from its persistent store. It then queries its neighbors and appropriately changes its keys or realises that a key update is not required.
- o Once this is done, in order to query the GCKS, the rebooted router chooses a random time interval so as to avoid clashes with other routers querying the GCKS.

Due to the randomness introduced, chances of the GCKS being flooded with requests are reduced. The GCKS when queried, could give the router information corresponding to its new adjacencies, probably the time of change of its adjacencies and any other relevant rekey information. This enables the rebooted router to know whether its traffic keys are stale or not.

Another fine point here is that very rarely the rekey process could be in progress when the router comes up. This is a corner case and is being left for future work.

6.5. Scalability

Any system that has widespread deployment should be designed keeping the scalability feature in mind. If scalability is overlooked during the design phase, the system would fail on high loads when actually deployed.

We have designed the automated key management system so as to make it scalable. We have already mentioned that we are limiting the scope of our problem to key and adjacency management within an AD. Even

within an AD since the number of routers is not fixed, the system should be able to handle a variable/ large number of routers. The proposed protocol involves a set of GCKS-GM interactions and a set of GM-GM interactions. The GM-GM communication is only among neighboring GMs and hence scalability is not an issue for that. Even for the GCKS-GM communication in the normal case, there should not be any issue since all GMs are not installed or turned on at the same time. However, a situation to be considered is when the GMs reboot. It could so happen that due to a power outage, all GMs in the AD go down and come back up at approximately the same time. It is extremely important to ensure that the GCKS is not stormed with requests at this point.

Our proposal handles this case in a couple of ways. Firstly we have seen that the LKS of each GM maintains a stable storage. All important pieces of information, such as the ones got from the GCKS and from the neighboring GMs are written to this storage, which is persistent across reboots. Hence a GM after a reboot, reads information directly from its persistent storage thereby preventing the GCKS from being flooded with requests. Secondly after retrieving information from the local storage, when the GMs need to query the GCKS itself, they do so by starting a timer and querying at a random time interval. This plays a major role in preventing the GCKS from being overloaded thereby leading to scalability.

Another factor that enables partial distribution of functionality thereby enhancing scalability is the presence of the Standby GCKS. If a situation arises such that the active GCKS fails (which could be due to an overload), the Standby GCKS would immediately take over the functionality of the active one. This eliminates a single point of failure and hence allows the system to withstand higher loads, or more number of GMs in the AD.

6.6. Option to Turn Off Adjacency Management

We have already discussed why it is important for an automated key management system to manage adjacencies well. In fact, this is because routing protocol updates are usually exchanged with neighbors, which in turn leads to the requirement that communicating routers should be legitimate neighbors. It is a good practice to have adjacency management turned on in a network so that for any router, only its legitimate neighbors and all of its legitimate neighbors get to know the keys it uses for securing its control traffic.

However, sometimes an administrator may decide to turn off adjacency checks because his network of routers is probably too small and the extra overhead is not required. This would mean that any router is

then allowed to query for and receive the traffic keys of any other router in the network even though the routers may not be neighbors. If adjacency management is turned off, even routing protocols would respond to all control packets without performing adjacency checks. This definitely reduces security in the network.

If the key scope is such that the same traffic key is used throughout the AD, not much harm is caused if a router gives its key information to any other router in the AD since all routers share the same key. Of course mutual authentication of the routers should happen in order to know if the routers are valid members of the AD. However, an administrator could use the key per sender model, for example, and turn off adjacency management. The administrator then relies on the physical adjacency to ensure that a router far away from another router does not query it for keys.

6.7. Incremental Deployment

Whenever a new system is to be deployed in the real world, the ease with which that can be done is of utmost importance. Network operators may not be ready to switch over to a new system if it is not easy to deploy it. Also, operators using a certain setup, when switching over to a new one would usually want to deploy the new system on an incremental basis. This would help them detect problems in the new system, if any, and then decide whether to completely move to the new model or not. We have designed our automated key management system keeping this requirement in mind. The model we have proposed can be deployed on a per interface basis. This means that initially GMs could be manually configured with the TEKs for some of their interfaces, and made to run the key management protocol to derive TEKs corresponding to the other interfaces. This is for the case of separate key per interface of each router. The other cases of keying groups can be handled in a similar manner. Secondly, the new system can be used to provide TEKs for one routing protocol at a time. This again makes the transition from the manual method of configuration to the automated method smooth.

6.8. Smooth Key Rollover

Whenever the TEK is changed, smooth key rollover should be ensured so that no packets are dropped during the process of key transitions. In order to achieve this, while transitioning from the old key to the new one, for a short duration routers have to accept messages secured using either key. This allows for the time delay involved in the new keys being received by all routers participating in that particular communication. After a certain time period as determined by a timer, the old key information could be cleared. For smooth key rollover in multicast communication, these points have been explained in more

detail in [RFC5374]. For unicast communication, either this method could be followed or the two participating routers could exchange new keys and acknowledge the receipt of the keys just before beginning to use them.

6.9. Eliminating Single Point of Failure

The proposed design for key management describes the use of a centralized GCKS as the controller and co-ordinator for the entire AD. In any centralized system, there is a possibility of having a single point of failure. In such a system, if the central entity goes down, it could so happen that the entire system stops functioning due to loss of important data. This can be avoided by having a backup entity to take over when the primary controller goes down. This is precisely what is proposed in our design in Section 4.2. We propose maintaining a Standby GCKS, which is always kept in sync with the primary GCKS. This can be done by correctly syncing all data from the active to the standby at regular intervals. The appropriate interval could be determined by the policies handed down by the Policy Server to the GCKS. Whenever the active goes down, the standby can immediately take over its responsibility thereby preventing any interruption in the functioning of the system. This introduces a certain degree of distribution of functionality and hence can successfully eliminate a single point of failure.

7. Detailed Packet Formats

TBD

8. IANA Considerations

This document has no actions for IANA.

9. Acknowledgements

10. Change History (RFC Editor: Delete Before Publishing)

[NOTE TO RFC EDITOR: this section for use during I-D stage only. Please remove before publishing as RFC.]

atwood-karp-akam-rp-02

- o Inserted ASCII art for figures and hashes
- o Resolved internal cross-references
- o Resolved external citations

atwood-karp-akam-rp-01

- o copied in the rest of the relevant material from Revathi's thesis
- o added overview material on protocol operations

atwood-karp-akam-rp-00 (original submission, based on Revathi's thesis)

- o copied in some sections of the thesis that are relevant to the specification.

11. Needs Work in Next Draft (RFC Editor: Delete Before Publishing)

[NOTE TO RFC EDITOR: this section for use during I-D stage only. Please remove before publishing as RFC.]

List of stuff that still needs work

- o
- o
- o Create the section on packet formats
- o
- o

12. References

12.1. Normative References

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

12.2. Informative References

[I-D.hartman-karp-mrkmp]

Hartman, S., Zhang, D., and G. Lebovitz, "Multicast Router Key Management Protocol (MaRK)", draft-hartman-karp-mrkmp-04 (work in progress), March 2012.

[I-D.ietf-karp-crypto-key-table]

Housley, R., Polk, T., Hartman, S., and D. Zhang, "Database of Long-Lived Symmetric Cryptographic Keys", draft-ietf-karp-crypto-key-table-03 (work in progress),

June 2012.

[I-D.ietf-karp-ops-model]

Hartman, S. and D. Zhang, "Operations Model for Router Keying", draft-ietf-karp-ops-model-03 (work in progress), July 2012.

[I-D.ietf-karp-threats-reqs]

Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Overview, Threats, and Requirements", draft-ietf-karp-threats-reqs-05 (work in progress), May 2012.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.

[RFC4535] Harney, H., Meth, U., Colegrove, A., and G. Gross, "GSAKMP: Group Secure Association Key Management Protocol", RFC 4535, June 2006.

[RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, November 2008.

[RFC5796] Atwood, W., Islam, S., and M. Siami, "Authentication and Confidentiality in Protocol Independent Multicast Sparse Mode (PIM-SM) Link-Local Messages", RFC 5796, March 2010.

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

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, October 2011.

[RFC6518] Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Design Guidelines", RFC 6518, February 2012.

[atwo2009:AKM]

Atwood, J., "Automated Key Management for Router Updates", October 2009.

Authors' Addresses

William Atwood
Concordia University/CSE
1455 de Maisonneuve Blvd, West
Montreal, QC H3G 1M8
Canada

Phone: +1(514)848-2424 ext3046
Email: william.atwood@concordia.ca
URI: <http://users.encs.concordia.ca/~bill>

Revathi Bangalore Somanatha
Concordia University/CSE
1455 de Maisonneuve Blvd, West
Montreal, QC H3G 1M8
Canada

Email: revathi.bs@gmail.com

Working Group
Internet-Draft
Intended status: Informational
Expires: January 31, 2013

U. Chunduri
A. Tian
Ericsson Inc.
July 30, 2012

KARP KMP: Simplified Peer Authentication
draft-chunduri-karp-kmp-router-fingerprints-00

Abstract

This document describes the usage of Router Fingerprint Authentication (RFA) with public keys. This can be used as a peer authentication method with KARP Key Management Protocol (KMP). KARP KMP automates key negotiation for securing TCP-based pairwise Routing Protocols (RPs) like BGP, LDP. The advantage of RFA is, neither it requires out-of-band, mutually agreeable symmetric keys nor a full PKI based system (trust anchor or CA certificates) for mutual authentication of the peers with KARP KMP deployments. Usage of Router Fingerprints give a significant operational improvement from symmetric key based systems and yet provide a secure authentication technique.

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."

This Internet-Draft will expire on January 31, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
1.2. Acronyms	4
2. Router Fingerprint	4
3. Usage of Router Fingerprints with KARP KMP	5
4. Impact on the PAD	5
5. Publishing Router Fingerprints	6
6. IANA Considerations	6
7. Security Considerations	6
8. Acknowledgements	6
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Authors' Addresses	8

1. Introduction

A Key Management Protocol (KMP) framework for TCP-based pair wise routing protocols (BGP [RFC4271], PCEP [RFC5440], MSDP [RFC3618] and LDP [RFC5036]) is detailed in [chunduri-karp-using-ikev2-with-tcp-ao]. Usage of IKEv2 [RFC5996] as the KMP is also described in the same document. This draft explores a simple and secure authentication method, which can be used for KARP KMP deployments.

Currently operators don't often change the manual keys deployed for protecting the Routing Protocol (RP) messages because of various reasons as noted in Section 2.3 of KARP threat document [I-D.ietf-karp-threats-reqs]. One of the KARP WG goals is to define mechanisms to support key changes for all RPs which use either Manual Key Management (MKM) or KMP with out much operational overhead.

Apart from Peer's identity verification, authentication and parameter negotiation, deployment of KMP can be more useful, when it comes to rekey the keys used by RPs. Rekeying can be achieved with out the operator's intervention and as per the provisioned rekey policy. But, the usage of IKEv2 KMP opens up numerous possibilities for peer authentication. Various peer authentication mechanisms with the advantages/drawbacks of each mechanisms are described in the Appendix of the [chunduri-karp-using-ikev2-with-tcp-ao] document.

If symmetric pre-shared keys are used by IKEv2 KMP to authenticate the peer before generating the shared key(s), apart from the other issues with symmetric keys, the problem still remain the same when it comes to changing these keys.

To reduce the operational costs for changing the keys at peering points with 100s of peers, this document describes the use of one of the available IKEv2 KMP peer authentication methods with raw or x.509 encoded public keys (to be called as Router Fingerprints in the rest of the document). Router Fingerprint Authentication (RFA) mechanism in conjunction with KARP KMP require neither out-of-band symmetric keys nor a fully functional PKI based system with trust anchor certificates as explained further in Section 2.

Section 2 describes the Router Fingerprints in the context of various KMPs and specifically for IKEv2 KMP. Generation and usage of the Router Fingerprints is described in Section 3 and Section 5 describes an error free method for publishing the Router Fingerprints.

1.1. Requirements Language

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].

1.2. Acronyms

EE	-	End Entity
KMP	-	Key Management Protocol (auto key management)
MKM	-	Manual Key management Protocols
PAD	-	Peer Authorization Database
RFA	-	Router Fingerprint Authentication
RP	-	Routing Protocol

2. Router Fingerprint

Router Fingerprint is a sequence of bytes used to authenticate the public key before using the same to authenticate the peer in the context of KMP.

Various forms of the fingerprint mechanism based on the public keys are already in use as defined in [RFC4252] and [RFC4253]. Fingerprints are also used primarily for root key authentication in x.509 based PKI [RFC5280]. This document only highlights the usage of raw public key based authentication mechanism already defined in [RFC5996] for KARP deployments.

To generate a fingerprint:

1. A router need to generate an asymmetric Private/Public key pair. Asymmetric crypto algorithms based on RSA [RFC3447] or for shorter and still secure keys Elliptic Curve Cryptography (ECC) [RFC4492] can be used for generating the Private/Public key pair.
2. Once the Asymmetric key pair is generated, if needed, the public key can be in the form of raw public key as specified in [RFC5996] or can be encoded with any additional data (specific to the router) and can be in the form of more easily administrable X.509 PKI Certificate profile [RFC5280].

3. The result should be hashed with a cryptographic hash function, preferably SHA-256 or hash functions with similar strength (see more discussion on choosing preferred hash function in Section 7).

The fingerprint generated is not a secret and can be distributed publicly. This is further discussed in Section 5.

3. Usage of Router Fingerprints with KARP KMP

To use Router Fingerprints authentication with KARP KMP, a Private/Public key-pair MUST be generated by the router as specified in Section 2. Base IKEv2 [RFC5996] standard supports only raw RSA based public keys. The type of the public keys and encoding has to be more generic to deploy this peer authentication method.

With current specification [RFC5996] when sender needs to get the certificate of the receiver, Certificate Request payload (CERTREQ as specified in [RFC5996]) is sent with cert encoding set to "Raw RSA Key" and Certification Authority field is empty. The receiver of this CERTREQ payload, uses PKCS #1 encoding for the generated RSA Public Key and sends the same in CERT payload as Certificate Data with Certificate Encoding set to "Raw RSA Key" as described in Section 3.6 of IKEv2 [RFC5996]. Once the public key of the sender is received, the verification MUST be done with the already published/stored fingerprints of the sender.

As noted above the current IKEv2[RFC5996] specification only supports raw RSA public keys. [I-D.kivinen-ipsecme-oob-pubkey] enhances support for other types of public keys and also defines new encoding format to carry the public key fingerprint in the CERT payload. For RPs to use Router Fingerprint Authentication in the context of IKEv2 MUST follow the encoding format as specified in [I-D.kivinen-ipsecme-oob-pubkey].

4. Impact on the PAD

The Peer Authorization Database (PAD) and the role it plays in peer authentication is defined in section 4.4.3 of [RFC4301]. One of the functions of the PAD is to provide the authentication data for each peer. [RFC4301] supports X.509 certificate or pre-shared secret authentication data types. So, it is necessary to encode the raw public keys as X.509 certificates before sending the same in CERT payload. Though the public key received is in the form of x.509 certificate, for RFA, the PAD entry need not contain a trust anchor via which the end entity (EE) certificate or the public key for the

peer must be verifiable. The PAD entry MUST rather contain the published finger print of the peer.

5. Publishing Router Fingerprints

The router fingerprint generated is not a secret and can be exchanged out-of-band through Service Level Agreements (SLAs) at the RP peering points or can be distributed publicly. A KARP KMP deployment using router fingerprints need to resort to out-of-band public key validation procedure to verify authenticity of the keys being used. The router fingerprints should be part of the KMP Peer Authorization Database (PAD) to validate the public key received in the KMP messages. For conveying router fingerprints data bytes in a clear unambiguous way PGP (Pretty Good Privacy) wordlists can be used.

6. IANA Considerations

This document defines no new namespaces.

7. Security Considerations

If collision attacks are perceived as a threat, the hash function to generate the fingerprints SHOULD also possess the property of collision-resistance. To mitigate preimage attacks, the cryptographic hash function used for a fingerprint SHOULD possess the property of second preimage resistance.

If generated fingerprints are truncated to make those short, the truncated fingerprints MUST be long enough to preserve the relevant properties of the hash function against brute-force search attacks.

Considering the above facts, it's recommended to use SHA-256 or similar hash functions with good security properties to generate the fingerprints.

8. Acknowledgements

The authors would like to thank Jari Arkko for initial and valuable discussions on operationally simplified authentication mechanisms in general and RFA mechanism as described in this document in particular. Thanks to Tero Kivinen for extended discussion on applicability and usage of authentication method described for KARP KMP. Thanks to Joel Halpern for supporting this work and providing continuous feedback.

9. References

9.1. Normative References

- [I-D.chunduri-karp-using-ikev2-with-tcp-ao]
Chunduri, U., Tian, A., and J. Touch, "Using IKEv2 with TCP-AO", draft-chunduri-karp-using-ikev2-with-tcp-ao-01 (work in progress), March 2012.
- [I-D.kivinen-ipsecme-oob-pubkey]
Kivinen, T., Wouters, P., and H. Tschofenig, "More Raw Public Keys for IKEv2", draft-kivinen-ipsecme-oob-pubkey-00 (work in progress), March 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

9.2. Informative References

- [I-D.ietf-karp-threats-reqs]
Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Overview, Threats, and Requirements", draft-ietf-karp-threats-reqs-05 (work in progress), May 2012.
- [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
- [RFC3618] Fenner, B. and D. Meyer, "Multicast Source Discovery Protocol (MSDP)", RFC 3618, October 2003.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.
- [RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.
- [RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.
- [RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5440] Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.
- [RFC6518] Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Design Guidelines", RFC 6518, February 2012.

Authors' Addresses

Uma Chunduri
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Phone: +1 (408) 750-5678
Email: uma.chunduri@ericsson.com

Albert Tian
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Phone: +1 (408) 750-5210
Email: albert.tian@ericsson.com

Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

U. Chunduri
A. Tian
Ericsson Inc.
J. Touch
USC/ISI
July 16, 2012

Using IKEv2 with TCP-AO
draft-chunduri-karp-using-ikev2-with-tcp-ao-02

Abstract

This document describes a mechanism to secure TCP-based pairwise Routing Protocol (RP) associations using the IKEv2 Key Management Protocol (KMP) integrated with TCP-AO. A Gatekeeper (GK) mechanism is introduced to allow TCP-AO to coordinate with IKEv2 without fundamental modification to either. This document also introduces extensions to IKEv2 and its Security Associations to enable its key negotiation to support TCP-AO. The document also includes a summary of IKEv2 authentication methods available for peer authentication for use in protecting routing 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."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Acronyms	4
2. Motivation and Overview	4
3. The Gatekeeper	6
3.1. RP interface to the Gatekeeper	6
3.2. Interface to the PAD	7
3.3. KMP interface to Gatekeeper	8
3.3.1. Interface to KARP Crypto Key Table	9
3.4. TCP-AO interface to Gatekeeper	10
3.5. Impact of Policy changes	11
4. Extensions required for IKEv2	11
4.1. Non IPsec DOI	11
4.1.1. Security Association Extensions	12
4.2. Simple Traffic Selectors Negotiation	13
5. IANA Considerations	13
6. Security Considerations	13
7. Acknowledgements	13
8. Appendix A	13
8.1. BGP Multi Session and transport level differentiation . .	13
8.2. Applicable Authentications methods	14
8.2.1. Symmetric key based authentication	14
8.2.2. Asymmetric key based authentication	15
8.2.3. EAP based authentication	15
9. References	16
9.1. Normative References	16
9.2. Informative References	17
Authors' Addresses	18

1. Introduction

A security threat analysis for TCP-based routing protocols (BGP [RFC4271], PCEP [RFC5440], MSDP [RFC3618] and LDP [RFC5036]) is detailed in [ietf-karp-routing-tcp-analysis]. The KARP design guide [RFC6518] suggests various requirements and options for obtaining keys to protect the routing protocols and recommends using a Key Management Protocol (KMP) to automate key establishment, as well as rekeying to continuously protect the routing protocols.

This document analyzes the TCP-based pairwise Routing Protocol (RP) requirements needed to integrate the IKEv2[RFC5996] KMP together with TCP-AO[RFC5925] to protect routing protocols.

This document introduces a new Gatekeeper module, which provides a common interface and minimizes the changes for all routing protocols (BGP, PCEP, MSDP and LDP) to be integrated with KMP. The Gatekeeper module does the SA management and interaction with KMP as well as TCP-AO protocol. The purpose of the Gatekeeper is to act as a shim between IKEv2 and TCP-AO, so that TCP-AO and the Gatekeeper together act like IPsec to IKEv2 (since IKEv2 is designed to tightly interact with IPsec). This document defines this common interface between all TCP-based pairwise routing protocols with Gatekeeper and IKEv2 [RFC5996].

Currently IKEv2 can establish only Security Association (SA) for IPsec. A few extensions are needed for IKEv2 to establish SA for TCP-based routing protocols when TCP-AO is used for protection. Section 4 discusses the summary of extensions required for IKEv2 protocol for key establishment, traffic selectors negotiation and SA establishment to support the keying and parameters needed by TCP-AO.

One of the services provided by IKEv2 KMP is peer authentication. This happens before traffic keys are established between IKEv2 peers. As IKEv2 KMP provides a variety of authentications methods; Section 8.2 discusses various Symmetric, Asymmetric and EAP based KMP authentication options available. The goal of Section 8.2 is to summarize vastly scattered information for choosing the right authentication method by operators for peer authentication with low operational overhead and yet secure mechanism especially suitable for routing environments.

1.1. Requirements Language

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].

1.2. Acronyms

BGP	-	Border Gateway Protocol
EAP	-	Extensible Authentication Protocol
GKR	-	Gatekeeper Record
IKEv2	-	Internet Key Exchange Protocol Version 2
IPsec	-	Security Architecture for the Internet Protocol
KDF	-	Key Derivation Function
KMP	-	Key Management Protocol (auto key management)
LDP	-	Label Distribution Protocol
MKM	-	Manual Key management Protocols
MKT	-	Master Key Tuples as defined in TCP-AO
MSDP	-	Multicast Source Discovery Protocol
PAD	-	Peer Authorization Database
PCEP	-	Path Computation Element Communication Protocol
RP	-	Routing Protocol
SA	-	Security Association
TCP-AO	-	TCP Authentication Option

2. Motivation and Overview

IKEv2 assumes IPsec triggers new SA requests, manages SA timers and rekeys SAs as needed. TCP-AO assumes an external key manager, which could support functions like Master key triggering, SA timers, and rekey triggering to get all the parameters required including Master key to protect the TCP session. To bridge the gap between IKEv2 and TCP-AO, this document defines a Gatekeeper module as described in Section 3.

The motivation of this document is to offload Security Association (SA) management and to provide a generic and common interface for all TCP-based RPs to integrate with KMPs in general and specifically with

IKEv2 KMP.

The following diagram depicts the Gatekeeper module interfaces with all protocols involved i.e., TCP-based RPs, IKEv2 KMP, and TCP-AO. This also shows the interaction with various databases viz., Peer Authorization Database (PAD) and Crypto Key Tables interaction with the Gatekeeper.

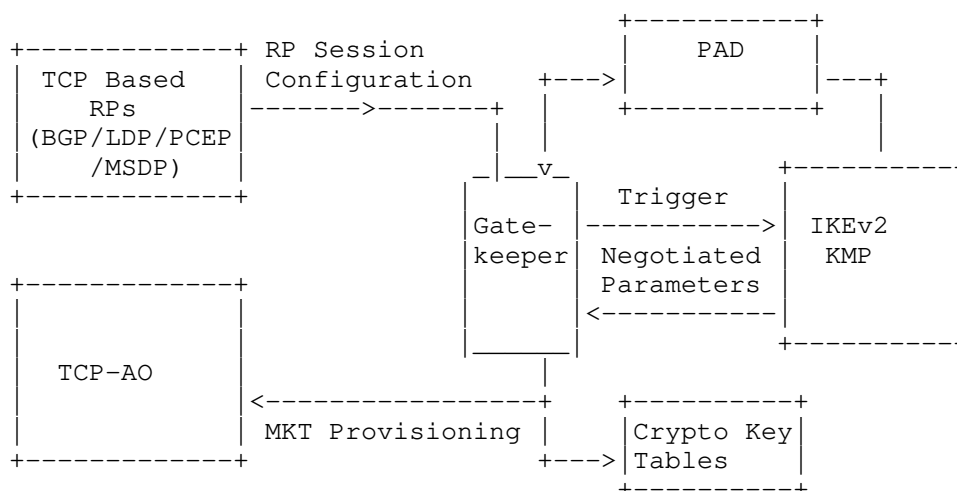


Figure 1: KARP KMP: Using IKEv2 with TCP-AO

In Figure 1, before initiating the TCP connection, all TCP-based RPs communicate the provisioned configuration to Gatekeeper module. A entry in the KMP peer authentication/authorization is provisioned in PAD as defined in Section 4.4.3 of [RFC4301] and pointer to this entry SHOULD be part of the RP configuration. This facilitates Gatekeeper to issue a corresponding request, with all the proposed alternatives at the RP to the IKEv2 KMP, so IKEv2 can negotiate the needed parameters. When the local peer is acting as a responder, security policy information populated at the Gatekeeper can be referenced through PAD by IKEv2 KMP to create the CHILD_SAs. Either way, the negotiated parameters are kept in the crypto key table database as specified in [ietf-karp-crypto_key-table] and this information is basis for provisioning MKTs in the TCP-AO.

Gatekeeper maintains the KMP SAs and initiates rekey triggers as needed to provision new MKTs for the long-lived TCP sessions protected by TCP-AO. The Gatekeeper installs these new keys in

TCP-AO consistent with TCP-AO's support for key changes.

Section 3 describes in detail the role of Gatekeeper and its interfaces to all the protocols and the databases it interacts with. Section 3.3.1, Section 3.2 describes the database and the interaction with the Gatekeeper in detail.

3. The Gatekeeper

TCP-AO has a different model of security associations and key management than IPsec. IKEv2 is designed to support IPsec's model. This document introduces the Gatekeeper to enable IKEv2 to support key and parameter negotiation that can be used in TCP-AO, as identified in Section 2.

The Gatekeeper maintains a Gatekeeper record (GKR) to keep track of TCP-AO MKTs. For long-lived TCP connections MKTs can be rolled over by rekeying creating new MKTs and installing them in TCP-AO. The GKR can be viewed as a superset of MKT; it maintains and tracks the lifetime of the provisioned MKT, and includes other per-connection parameters needed by TCP-AO, such as algorithm, key length, etc. [RFC5926]. It also maintains the reference to PAD and Crypto Key Table entries to facilitate RP security parameters negotiation with IKEv2 KMP.

The next section defines the Gatekeeper module interface between TCP-based RPs (BGP, LDP, MSDP, PCEP), interface with IKEv2, TCP-AO and other key databases.

3.1. RP interface to the Gatekeeper

When a routing protocol is configured to use TCP-AO with KMP (by not specifying the keys or through some other means), TCP connection identifiers, all configured Message Authentication Code (MAC) algorithms, all configured Key Derivation Function (KDF) parameters, rekey lifetime and the TCP option flag (i.e., all additional parameters specified in [RFC5926]) are provisioned in the Gatekeeper record. This provisioning includes the reference to PAD, which has all the information to authorize and authenticate IKEv2 peer.

If the same routing protocol (RP) needs differentiate transport sessions to differently securing separate TCP connections between the same endpoints, TCP connection identifiers either the full socket pair (i.e., local IP address, remote IP address, local TCP port, and remote TCP port) or partial socket pair values (as indicated with wildcards) need to be provisioned. GKRs SHOULD thus support full or partial socket pair specification.

In general, a full socket pair is not needed for negotiating the TCP-AO MKT with KMP. As specified in Section 3.1 of TCP-AO [RFC5925], socket pair values can be partially specified using ranges, masks, wildcards, or any other suitable indication. These provisioned socket pair parameters are supplied to KMP as context in which to negotiate traffic selectors for which the MKT or Master key should be used in TCP-AO.

For more details on cases where a full socket pair is needed before opening the connection, please refer Section 8.1. Provisioning of the Gatekeeper record SHOULD be done before opening the TCP connection. From the RP interface, the record created in Gatekeeper contains only the RP's connection information, and this information is given to KMP (IKEv2) to obtain the negotiated parameters to provision the MKT to protect the underlying TCP session by [RFC5925].

3.2. Interface to the PAD

The Peer Authorization Database (PAD) for IPsec is described in Section 4.4.3 of [RFC4301]. This section describes the embodiments of the same in the context of RP security associations and security policies provisioned at the routing protocols. This is still the link between policies provisioned at the routing protocol and the SAs created by IKEv2 KMP. Instead of the Security Policy Database (SPD), Gatekeeper record holds the data for traffic selectors for child SA creation.

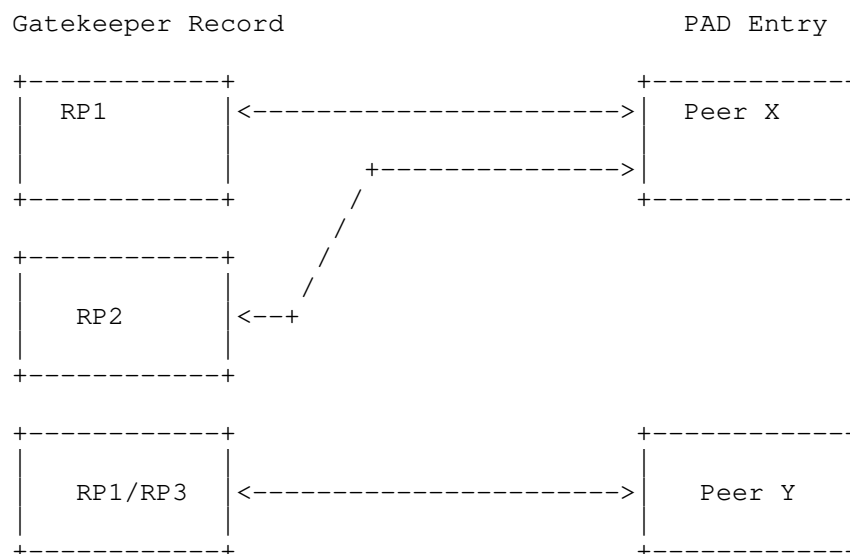


Figure 2: KARP KMP: Gatekeeper interface to the PAD

As shown in Figure 2, multiple RPs can point to the same peer and in this case, a PAD entry holds the reference to both the corresponding Gatekeeper records. The PAD entry for the IKEv2 peer is used to constrain the creation of child SAs; specifically, the PAD entry specifies how the Gatekeeper record is searched using a traffic selector proposal from a peer. For CHILD_SA creation, peer IP addresses asserted in traffic selector payloads SHOULD be used for Gatekeeper record lookups based on the remote IP address field portion of a Gatekeeper Record entry.

3.3. KMP interface to Gatekeeper

As an initiator, IKEv2 expects an external trigger that contains the information required to negotiate security associations. There needs to be a way to trigger the KMP to initiate negotiation with all the provisioned parameters of a Gatekeeper record by a TCP-based RP. A similar trigger is also required to rekey, to maintain the negotiated SAs for long-lived connections. As a responder to the peer IKEv2 requests and CHILD_SA creation; Gatekeeper record is consulted through the reference in PAD.

The purpose of this section is to define a common interface between the Gatekeeper and the IKEv2 KMP and also to list all the negotiated parameters to form an entry in the Crypto Key Tables.

3.3.1. Interface to KARP Crypto Key Table

KMP negotiated parameters are kept in the crypto key table database as specified in [ietf-karp-crypto_key-table]. The database is characterized as a table, where each row represents a single long-lived symmetric cryptographic key or Master key. The Gatekeeper record SHOULD have a reference to the Crypto Key Table Entry. One of the reasons to separate the negotiated parameters in a different table is to alleviate the population manually or through a some external source.

The following are the details:

1. At the time of a new connection, a trigger to the KMP occurs to negotiate the session-specific parameters with the needed information on MAC algorithm, KDF parameter, Traffic Selectors, and the TCP option flag from the Gatekeeper record. The Gatekeeper at the peer is expected to have similar provisioning in place for responding to the received KMP request.
2. A KMP session identifier, provided by a successful key negotiation by the KMP, needs to be stored and should be used when the Gatekeeper make decision based on the lifetime to rekey the existing session.
3. MKT IDs (as specified in Section 3.1 of TCP-AO [RFC5925]) require a SendID and a RecvID for each MKT, mutually agreed by the connection endpoints. These 1-byte quantities need to be negotiated by the KMP with the peer to populate in the MKT. These fields are populated as "LocalKeyName" and "PeerKeyName" in the Crypto Key Table entry.
4. Crypto Key Table "Peers" field SHOULD be populated with the peer IP address.
5. KMP-negotiated KDF parameters for each session used to generate traffic keys from master keys to be populated in MKT. The same is referred as "KDF" in a corresponding Crypto Key Table entry.
6. A KMP-negotiated MAC algorithm, MKT connection identifiers (negotiated traffic selectors) and optionally life time for traffic keys for each session, need to be populated in MKT. The same is referred as "AlgID" in a corresponding Crypto Key Table entry.
7. The "Key" field defined in Crypto Key Table contains a long-lived symmetric cryptographic key or Master Key in the format of a lower-case hexadecimal string. The size of the Key depends on

the KDF and the AlgID.

8. IKEv2 does not negotiate rekey lifetime and rekeying is based on local operator policy. The Gatekeeper adds this capability, tracking the key lifetime provisioned at TCP-based RP and explicitly triggering the KMP to rekey when indicated. This rekey trigger then creates a new MKT for the underlying TCP connection. Implementations can proactively negotiate a new MKT Master Key before the lifetime of the current Master key expires.

3.4. TCP-AO interface to Gatekeeper

TCP-AO expects an external entity to provision its MKTs in order to protect TCP sessions. The Gatekeeper module provides this function so that all TCP-based RPs can benefit from this common interface.

The following are the details of the interface between TCP-AO and the GK:

1. After getting the negotiated parameters and mutually authenticated Master keys from the KMP, the Gatekeeper inserts a corresponding MKT and parameters into TCP-AO. The session-specific parameters include negotiated Connection identifiers, MAC algorithms, KDFs, KeyIDs, the TCP option flag and the Master Key given by the KMP.
2. MKT IDs (as specified in Section 3.1 of TCP-AO [RFC5925]) require a SendID and a RecvID for each MKT, which are mutually agreed by the connection endpoints. These 1-byte quantities need to be part of the MKT when the KMP key(s) are populated in MKT.
3. For long-lived TCP sessions, the Gatekeeper removes the old MKTs from TCP-AO after rekeying the corresponding new MKTs, to continuously protect the underlying TCP sessions.
4. In general, restarted TCP sessions can use existing MKT in TCP-AO i.e., IKEv2 need not be retriggered, since new key and parameter negotiation is not needed due to the protection already provided by TCP-AO (refer Section 5.3.1 of TCP-AO [RFC5925]). However, if GKR and hence TCP-AO MKT is created with full socket pair (in other words without using ranges, masks, wildcards for socket pair values, for the cases as specified in Section 8.1), then IKEv2 needs to be retriggered to get the new master key for the corresponding restarted TCP session.

3.5. Impact of Policy changes

Once the routing session is secured by TCP-AO, any security policy changes initiated by the operator at RP MUST cause a tear down of the existing session and MUST be replaced with a new CHILD_SA at IKEV2 KMP and corresponding new MKT at TCP-AO. Similarly, any changes in the peer Authentication data at PAD MUST cause re-authentication of the peer at IKEv2 KMP with changed credentials and also due to this change, all CHILD_SAs/MKTs need to re-negotiated.

4. Extensions required for IKEv2

There can be two ways to derive a KMP that is suitable for TCP-based routing protocols:

- a. Create a new KMP for routing protocols, e.g., based on IKEv2 (as proposed in [mahesh-karp-rkmp]).
- b. Extend IKEv2 to be suitable for TCP-based routing protocols.

In this section, we would like to explore option (b).

This section summarizes the extensions required for IKEv2 to negotiate non-IPsec SAs for TCP-based routing protocols. The authors acknowledge that some of the items below are already discussed in KARP WG, but the details presented here are different.

Routing protocols that use this extended IKEv2 KMP can continuously benefit from the new authentication methods and any other new features which might be added to [RFC5996].

4.1. Non IPsec DOI

IKEv2 is designed for performing mutual authentication with the peers and establishing and maintaining Security Associations for IPsec. IKEv2 defines the IKE_AUTH and CREATE_CHILD_SA exchanges, consisting of payloads and processing guidelines for IPsec Domain of Interpretation (DOI); this need to be generalized to exchange other protocol-specific parameters to be useful for RPs.

IKEv2 is designed to be extensible with additional parameters. The extensions proposed here can be deployed within that context, running over the existing IKEv2 port number and using existing IKEv2 tunneling mechanisms where needed.

The current IKEv2 CREATE_CHILD_SA exchange can be used to rekey the IKE SA and the master key. This document does not propose any

changes or extensions to re-establishing IKE SA through the CREATE_CHILD_SA exchange.

4.1.1. Security Association Extensions

The IKEv2 Security Association (SA) payload is used to negotiate attributes of a Security Association. This payload contains multiple proposals, as configured in the routing protocol. Possible extensions to be made are:

1. A new Protocol ID, to be added in the proposal substructure with TCP-AO as new protocol (IANA-TBD).
2. An Integrity Algorithm (INTEG), as defined in the transform substructure needs to be mandated for the new TCP-AO Protocol.
3. Authentication algorithms and associated parameters as defined in [RFC5926] should be extended to the current list in IKEv2 Transform Type 3 (Integrity Algorithm), for TCP-AO usage (IANA-TBD).
4. The Diffie-Hellman group (D-H) transform type can be used for TCP-AO proposal as an optional transform.
5. A new transform type is needed to represent the KDF for traffic key derivation by TCP-AO (IANA-TBD) and also needs to be mandated for the new TCP-AO Protocol. KDF algorithms and associated parameters as defined in [RFC5926] should be listed for this new transform in IKEv2 for TCP-AO usage (IANA-TBD).
6. A new transform type is needed to represent the TCP-AO KeyIDs (IANA-TBD). The Initiator KeyID represents the SendID and the Responder KeyID represents the RecvID in the TCP-AO MKT.
7. A new transform type needs to be created to indicate TCP options coverage by TCP-AO (IANA-TBD).
8. The valid transform types (as defined in Section 3.3.3 of [RFC5996]) for TCP-AO with mandatory and optional types need to be listed.
9. Attribute negotiation rules need to be extended for TCP-AO protocol.

4.2. Simple Traffic Selectors Negotiation

The Traffic Selectors defined in IKEv2 [RFC5996] have huge potential to negotiate the particular traffic to be secured, agreeable to both initiator and responder. For a routing protocol SA, traffic selectors negotiation presents a simple case and does not require any changes. A single connection or multiple connections with different source ports can be negotiated with a single CREATE_CHILD_SA exchange. The IP Protocol ID in the traffic selector field (as defined in Section 3.13.1 of [RFC5996]) can always be TCP for the routing protocol SAs.

The above is an attempt to summarize the brief list of changes in this approach and this section will be revisited.

5. IANA Considerations

This document requests that IANA to allocate new parameters as described in Section 4.1.1.

6. Security Considerations

This document does not introduce any new security threats for IKEv2 [RFC5996] or TCP-AO [RFC5925]. For more detailed security considerations please refer the Security Considerations section of the KARP Design Guide [RFC6518] document as well as KARP threat document [I-D.ietf-karp-threats-reqs].

7. Acknowledgements

The authors would like to thank Joel Halpern for his initial discussions and providing feedback on the document. The authors also thank Tero Kivinin and Dan Harkins for reviewing the document and Ron Bonica for his initial requirement discussions. Thanks to Sam Hartman for his KARP working group discussions on this topic.

The Gatekeeper module is originally proposed by Joe Touch.

8. Appendix A

8.1. BGP Multi Session and transport level differentiation

[ietf-idr-bgp-multisession] describes MP-BGP, which uses multiple TCP sessions between a pair of BGP speakers. Each TCP session is used to

exchange routes related by some session-based attribute, such as AFI/SAFI. The reason transport level distinction is required could be because of operator policy. Though it is less likely to see different MAC/KDF parameters for each of these sessions, it is possible rekey lifetimes or TCP option flags for TCP-AO can be different for each of these AFI/SAFI based sessions.

If transport level separation is required for all sessions between a pair of BGP speakers, a unique and full socket pair (i.e., a local IP address, a remote IP address, a local TCP port, and a remote TCP port) MUST be known before establishing a TCP connection. The full socket pair is required for both unique MKT creation in TCP-AO, as well as for the KMP to negotiate unique Master keys for each connection.

The use of different IP addresses to differentiate connections in multi session BGP is discouraged in [ietf-idr-bgp-multisession] and the destination port is always BGP. As a result, the only option for transport level differentiation is by knowing the source port of the connection being initiated. This is required to negotiate unique KMP SAs by the Gatekeeper, as well as to configure unique TCP-AO MKTs for each TCP connection. How source port lock-down is done is beyond the scope of this document (this is an implementation issue) and this can be achieved in many different ways before making the TCP connection.

The Gatekeeper interface, defined in Section 3, is oblivious to this issue and can well accommodate this requirement.

8.2. Applicable Authentications methods

One advantage that IKEv2 provides is the largest selection of key management and parameter coordination authentication methods suitable for various environments. The goal of this section is to look at various KMP authentication options available and recommend suitable options for use in negotiating keys and other parameters for routing protocol protection.

As some of the authentication mechanisms are optional in IKEv2, one mandatory authentication mechanism from the list below needs to be selected for routing environments to ensure inter-operability and quicker adoption. This section attempts to summarize the available options and constraints surrounding the options.

8.2.1. Symmetric key based authentication

IKEv2 [RFC5996] allows for authentication of the IKEv2 peers using a symmetric pre-shared key. For symmetric pre-shared key peer authentication, deployments need to consider the following as per

[RFC5996]:

1. Deriving a shared secret from a password, name, or other low-entropy source is not secure. These sources are subject to dictionary and social-engineering attacks, among others.
2. The pre-shared key should not be derived solely from a user-chosen password without incorporating another source of randomness.
3. If password-based authentication is used for bootstrapping the IKE_SA, then one of the EAP methods as described in Section 8.2.3 needs to be used.

One of the IPsecME WG charter goals is to provide IKEv2 [RFC5996] a secure password authentication mechanism which is protected against off-line dictionary attacks, without requiring the use of certificates or Extensible Authentication Protocol (EAP), even when using the low-entropy shared secrets. There are couple of documents which try to address this issue and the work is still in progress.

8.2.2. Asymmetric key based authentication

Another peer authentication mechanism for IKEv2 uses is asymmetric key certificates or public key signatures. This approach relies on a Public Key Infrastructure using X.509 (PKIX) Certificates. If this can be deployed for IKEv2 peer authentication, it will be one of the most secure authentication mechanisms. With this authentication option, there is no need for out-of-band shared keys between peers for mutual authentication.

Apart from RSA and DSS digital signatures for public key authentication provided by IKEv2, [RFC4754] introduces Elliptic Curve Digital Signature Algorithm (ECDSA) signatures. ECDSA provides additional benefits including computational efficiency, small signature sizes, and minimal bandwidth compared to other available digital signature methods.

8.2.3. EAP based authentication

In addition to supporting authentication using shared secrets and public key signatures, IKEv2 also supports authentication based on the Extensible Authentication Protocol (EAP), defined in [RFC3748]. EAP is an authentication framework that supports multiple authentication mechanisms. IKEv2 provides EAP authentication because public key signatures and shared secrets are not flexible enough to meet the requirements of many deployment scenarios. For KARP KMP, EAP-Only Authentication in IKEv2 as specified in [RFC5998] can be

explored.

By using EAP, IKEv2 KMP can leverage existing authentication infrastructure and credential databases, because EAP allows users to choose a method suitable for existing credentials. Routing protocols today use password-based pre-shared keys to integrity protect the routing protocol messages. The same pre-shared key can be used to bootstrap the KMP and as a potential authentication key in KMP. With appropriate password based EAP methods, stronger keys can be generated without using certificates.

For authenticating the nodes running routing protocols, EAP and the IKEv2 endpoints are co-located (so no separate EAP server required). When EAP is deployed, authenticating the IKEv2 responder using both EAP and public key signatures could be redundant. EAP methods that offer mutual authentication and key agreement can be used to provide responder authentication in IKEv2 completely based on EAP.

Section 4 of [RFC5998] lists safe EAP methods to support EAP_ONLY_AUTHENTICATION. For routing protocols deployment, because an EAP server is co-located with IKEv2 responder, channel binding capability of the selected EAP method is irrelevant. Various qualified mutual authentication methods are listed in [RFC5998]; of these, a password based methods [RFC4746], [RFC5931], [RFC6124] can offer potential EAP alternative for pre-shared key only authentication.

From the list above, Encrypted Key Exchange (EKE) as described in [RFC6124] is relatively lightweight and provides mutual authentication. This method also offers secure and robust authentication, even with an operator provisioned weak password in the presence of a strong adversary.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.
- [RFC5926] Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, June 2010.

- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.
- [RFC5998] Eronen, P., Tschofenig, H., and Y. Sheffer, "An Extension for EAP-Only Authentication in IKEv2", RFC 5998, September 2010.

9.2. Informative References

- [I-D.ietf-idr-bgp-multisession]
Scudder, J., Appanna, C., and I. Varlashkin, "Multisession BGP", draft-ietf-idr-bgp-multisession-06 (work in progress), March 2011.
- [I-D.ietf-karp-crypto-key-table]
Housley, R., Polk, T., Hartman, S., and D. Zhang, "Database of Long-Lived Symmetric Cryptographic Keys", draft-ietf-karp-crypto-key-table-03 (work in progress), June 2012.
- [I-D.ietf-karp-routing-tcp-analysis]
Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Security According to KARP Design Guide", draft-ietf-karp-routing-tcp-analysis-00 (work in progress), June 2011.
- [I-D.ietf-karp-threats-reqs]
Lebovitz, G., Bhatia, M., and R. White, "The Threat Analysis and Requirements for Cryptographic Authentication of Routing Protocols' Transports", draft-ietf-karp-threats-reqs-03 (work in progress), June 2011.
- [I-D.maresh-karp-rkmp]
Jethanandani, M., Zhang, D., Weis, B., Patel, K., and S. Hartman, "Key Management for Pairwise Routing Protocol", draft-maresh-karp-rkmp-00 (work in progress), October 2011.
- [RFC3618] Fenner, B. and D. Meyer, "Multicast Source Discovery Protocol (MSDP)", RFC 3618, October 2003.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic

Key Management", BCP 107, RFC 4107, June 2005.

- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4746] Clancy, T. and W. Arbaugh, "Extensible Authentication Protocol (EAP) Password Authenticated Exchange", RFC 4746, November 2006.
- [RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, January 2007.
- [RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.
- [RFC5440] Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.
- [RFC5931] Harkins, D. and G. Zorn, "Extensible Authentication Protocol (EAP) Authentication Using Only a Password", RFC 5931, August 2010.
- [RFC6124] Sheffer, Y., Zorn, G., Tschofenig, H., and S. Fluhrer, "An EAP Authentication Method Based on the Encrypted Key Exchange (EKE) Protocol", RFC 6124, February 2011.
- [RFC6518] Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Design Guidelines", RFC 6518, February 2012.

Authors' Addresses

Uma Chunduri
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Phone: +1 (408) 750-5678
Email: uma.chunduri@ericsson.com

Albert Tian
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Phone: +1 (408) 750-5210
Email: albert.tian@ericsson.com

Joe Touch
USC/ISI
4676 Admiralty Way,
Marina del Rey, California 90292-6695
USA

Phone: +1 (310) 448-9151
Email: touch@isi.edu

INTERNET-DRAFT
Internet Engineering Task Force (IETF)
Intended Status: Standards Track

R. Housley
Vigil Security
T. Polk
NIST
S. Hartman
Painless Security
D. Zhang
Huawei
29 June 2012

Expires: 29 December 2012

Database of Long-Lived Symmetric Cryptographic Keys
<draft-ietf-karp-crypto-key-table-03.txt>

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document specifies the information contained in a conceptual database of long-lived cryptographic keys used by many different security protocols. The database is designed to support both manual and automated key management. In addition to describing the schema for the database, this document describes the operations that can be performed on the database as well as the requirements for the security protocols that wish to use the database. In many typical scenarios, the security protocols do not directly use the long-lived key, but rather a key derivation function is used to derive a short-lived key from a long-lived key.

1. Introduction

This document specifies the information that needs to be included in a database of long-lived cryptographic keys in order to key the authentication of security protocols such as cryptographic authentication for routing protocols. This conceptual database is designed to separate protocol-specific aspects from both manual and automated key management. The intent is to allow many different implementation approaches to the specified cryptographic key database, while simplifying specification and heterogeneous deployments. This conceptual database avoids the need to build knowledge of any security protocol into key management protocols. It minimizes protocol-specific knowledge in operational/management interfaces, but it constrains where that knowledge can appear. Textual conventions are provided for the representation of keys and other identifiers. These conventions should be used when presenting keys and identifiers to operational/management interfaces or reading keys/identifiers from these interfaces. It is an operational requirement that all implementations represent the keys and key identifiers in the same way so that cross-vendor configuration instructions can be provided.

Security protocols such as TCP-AO [RFC5925] are expected to use per-connection state. Implementations may need to supply keys to the protocol-specific databases as the associated entries in the conceptual database are manipulated. In many instances, the long-lived keys are not used directly in security protocols, but rather a key derivation function is used to derive short-lived key from the long-lived keys in the database. In other instances, security protocols will directly use the long-lived key from the database. The database design supports both use cases.

2. Conceptual Database Structure

The database is characterized as a table, where each row represents a single long-lived symmetric cryptographic key. Normally, each key should only have one row. Only in the (hopefully) very rare cases where a key is used for more than one purpose, multiple rows will contain the same key value. The columns in the table represent the key value and attributes of the key.

To accommodate manual key management, the format of the fields has been purposefully chosen to allow updates with a plain text editor.

The columns that the table consists of are listed as follows:

LocalKeyName

LocalKeyName is a string identifying the key when it is received in a packet. A protocol may restrict the form of a key name. For example, many routing protocols will restrict key names to integers that can be represented in 16 or 32 bits.

PeerKeyName

For unicast communication, PeerKeyName on one system matches LocalKeyName on the other system. Similar to LocalKeyName, the protocol may restrict the form of this identifier and will often restrict it to be an integer. For group keys, the protocol will typically require this field be an empty string as the sending and the receiving key names need to be the same.

Peers

Typically for unicast keys, this field lists the peer systems that have this key in their database. For group keys this field names the groups for which the key is appropriate. For example, this might name a routing area for a multicast routing protocol. Formally, this field provides a protocol-specific set of restrictions on the scope in which the key is appropriate. The form of the identifiers in the Peers field is specified by the protocol.

Interfaces

The Interfaces field identifies the set of physical and/or virtual interfaces for which it is appropriate to use this key. When the long-lived value in the Key field is intended for use on any interface, this field is set to "all". The interfaces field consists of a set of strings; the form of these strings is specified by the implementation and is independent of the protocol in question. Protocols may require support for the interfaces field or may indicate that support for constraining keys based on interface is not required. As an example, TCP-AO implementations are unlikely to make the decision of what interface to use prior to key selection. In this case, the implementations are expected to use the same keying material across all of the interfaces and then require the "all" setting.

Protocol

The Protocol field identifies a single security protocol where this key may be used to provide cryptographic protection. This specification establishes a registry for this field; the registry also specifies the format of the following field, ProtocolSpecificInfo, for each registered protocol.

ProtocolSpecificInfo

This field contains the protocol-specified information which may be useful for a protocol to apply the key correctly. Note that such information must not be required for a protocol to locate an appropriate key. When a protocol does not need the information in ProtocolSpecificInfo, it will require this field be empty.

KDF

The KDF field indicates which key derivation function is used to generate short-lived keys from the long-lived value in the Key field. When the long-lived value in the Key field is intended for direct use, the KDF field is set to "none". This document establishes an IANA registry for the values in the KDF field to simplify references in future specifications. The protocol indicates what (if any) KDFs are valid.

AlgID

The AlgID field indicates the cryptographic algorithm used with the security protocol for the specified peer. The algorithm may be an encryption algorithm and mode (such as AES-128-CBC), an authentication algorithm (such as HMAC-SHA1-96 or AES-128-CMAC), or any other symmetric cryptographic algorithm needed by a security protocol. If the KDF field contains "none", then the long-lived key is used directly with this

algorithm, otherwise the derived short-lived key is used with this algorithm. When the long-lived key is used to generate a set of short-lived keys for use with the security protocol, the AlgID field identifies a ciphersuite rather than a single cryptographic algorithm. This document establishes an IANA registry for the values in the AlgID field to simplify references in future specifications. Protocols indicate which algorithms are appropriate.

Key

The Key field contains a long-lived symmetric cryptographic key in the format of a lower-case hexadecimal string. The size of the Key depends on the KDF and the AlgID. For instance, a KDF=none and AlgID=AES128 requires a 128-bit key, which is represented by 32 hexadecimal digits.

Direction

The Direction field indicates whether this key may be used for inbound traffic, outbound traffic, both, or whether the key has been disabled and may not currently be used at all. The supported values are "in", "out", "both", and "disabled", respectively. The Protocol field will determine which of these values are valid.

SendNotBefore

The NotBefore field specifies the earliest date and time in Universal Coordinated Time (UTC) at which this key should be considered for use when sending traffic. The format is YYYYMMDDHHSSZ, where four digits specify the year, two digits specify the month, two digits specify the day, two digits specify the hour, two digits specify the minute, and two digits specify the second. The "Z" is included as a clear indication that the time is in UTC.

SendNotAfter

The SendNotAfter field specifies the latest date and time at which this key should be considered for use when sending traffic. The format is the same as the NotBefore field.

RcvNotBefore

The RcvNotBefore field specifies the earliest date and time in Universal Coordinated Time (UTC) at which this key should be considered for use when processing received traffic. The format is YYYYMMDDHHSSZ, where four digits specify the year, two digits specify the month, two digits specify the day, two digits specify the hour, two digits specify the minute, and two digits specify the second. The "Z" is included as a clear indication that the time is in UTC.

RcvNotAfter

The RcvNotAfter field specifies the latest date and time at which this key should be considered for use when processing received traffic. The format of this field is identical to the format of NotBefore.

3. Key Selection and Rollover

A protocol may directly consult the key table to find the key to use on an outgoing packet. The protocol provides a protocol (P) and a peer identifier (H) into the key selection function. Optionally, an interface identifier (I) may also need to be provided. Any key that satisfies the following conditions may be selected:

- (1) the Peer field includes H;
- (2) the Protocol field matches P;
- (3) If an interface is specified, the Interfaces field includes I or "all";
- (4) the Direction field is either "out" or "both"; and
- (5) SendNotBefore <= current time <= SendNotAfter.

During algorithm transition, multiple entries may simultaneously exist associated with different cryptographic algorithms or ciphersuites. Systems should support selection of keys based on algorithm preference.

In addition, multiple entries with overlapping valid periods are expected to be employed to provide orderly key rollover. In these cases, the expectation is that systems will transition to the newest key available. To meet this requirement, this specification recommends supplementing the key selection algorithm with the following differentiation: select the long-lived key specifying the most recent time in the NotBefore field.

In order to look up a key for verifying an incoming packet, the protocol provides its protocol (P), the peer identifier (H), the key identifier (L), and optionally the interface (I). If one key matches the following conditions it is selected:

- (1) the Peer field includes H;
- (2) the Protocol field matches P;

- (3) if the Interface field is provided, it includes I or is "all";
- (4) the Direction field is either "in" or "both";
- (5) the LocalKeyName is L; and
- (5) RcvNotBefore <= current time <= RcvNotAfter.

Note that the key usage is loosely bound by the times specified in the NotBefore and NotAfter fields. New security associations should not be established except within the period of use specified by these fields, while allowing some grace time for clock skew. However, if a security association has already been established based on a particular long-lived key, exceeding the lifetime does not have any direct impact. The implementations of security protocols that involve long-lived security association should be designed to periodically interrogate the database and rollover to new keys without tearing down the security association.

Rather than consulting the conceptual database, a security protocol such as TCP-AO may update its own tables as keys are added and removed. In this case, the protocol needs to maintain its own key information.

4. Application of the Database in a Security Protocol

In order to use the key table database in a protocol specification, a protocol needs to specify certain information. This section enumerates items that a protocol must specify.

- (1) The ways of mapping the information in a key table row to the information needed to produce an outgoing packet; specified either as an explanation of how to fill in authentication-related fields in a packet based on key table information, or for protocols such as TCP-AO how to construct Master Key Tuples (MKTs) or other protocol-specific structures from a key table row
- (2) The ways of locating the peer identifier (a member of the Peers set) and the LocalKeyName inside an incoming packet
- (3) The methods of verifying a packet given a key table row; this may be stated directly or in terms of protocol-specific structures such as MKTs
- (4) The form and validation rules for LocalKeyName and PeerKeyName; if either of these is an integer, the conventions in Section 5.1 are used as a vendor-independent format

- (5) The form and validation rules for members of the Peers set
- (6) The algorithms and KDFs supported
- (7) The form of the ProtocolSpecifics field
- (8) The rules for canonicalizing LocalKeyName, PeerKeyName, entries in the Peers set, or ProtocolSpecifics; this may include normalizations such as lower-casing hexadecimal strings
- (9) The Indication whether the support for Interfaces is required by this protocol

5. Textual Conventions

5.1 Key Names

When a key for a given protocol is identified by an integer key identifier, the associated key name will be represented as lower case hexadecimal integers with the most significant octet first. This integer is padded with leading 0's until the width of the key identifier field in the protocol is reached.

5.2 Keys

A key is represented as a lower-case hexadecimal string with the most significant octet of the key first. As discussed in Section 2, the length of this string depends on the associated algorithm and KDF.

6. Operational Considerations

If the valid periods for long-lived keys do not overlap or the system clocks are inconsistent, it is possible to construct scenarios where systems cannot agree upon a long-lived key. When installing a series of keys to be used one after another (sometimes called a key chain), operators should configure the NotAfter field of the preceding key to be several days after the NotBefore field of the subsequent key to address the clock skew issue.

7. Security Considerations

Management of encryption and authentication keys has been a significant operational problem, both in terms of key synchronization and key selection. For instance, the current guidance [RFC3562] warns against sharing TCP MD5 keying material between systems, and recommends changing keys according to a schedule. The same general operational issues are relevant for the management of other cryptographic keys.

It has been recognized in [RFC4107] that automated key management is not viable in multiple scenarios. The conceptual database specified in this document is designed to accommodate both manual key management and automated key management. A future specification to automatically populate rows in the database is envisioned.

Designers should recognize the warning provided in [RFC4107]:

Automated key management and manual key management provide very different features. In particular, the protocol associated with an automated key management technique will confirm the liveness of the peer, protect against replay, authenticate the source of the short-term session key, associate protocol state information with the short-term session key, and ensure that a fresh short-term session key is generated. Moreover, an automated key management protocol can improve the interoperability by including negotiation mechanisms for cryptographic algorithms. These valuable features are impossible or extremely cumbersome to accomplish with manual key management.

8. IANA Considerations

This specification defines three registries.

8.1. KeyTable Protocols

This document requests establishment of a registry called "KeyTable Protocols". The following subsection describes the registry; the second subsection provides initial values for IEEE 802.1X.

8.1.1. KeyTable Protocols Registry Definition

All assignments to the KeyTable Protocols registry are made on a specification required basis per Section 4.1 of [RFC5226].

Each registration entry must contain the three fields:

- Protocol Name (unique within the registry);
- Specification; and
- Protocol Specific Values.

The specification needs to describe parameters required for using the conceptual database as outlined in Section 4. For existing protocols, this typically means that the specification will focus more on the application of the protocol with the key tables rather than being a general specification of the security protocol. New protocols may of course combine information on how to use the key tables database with the protocol specification.

8.1.2. KeyTable Protocols Registry Initial Values

Protocol Name: IEEE 802.1X

Specification: IEEE Std 802.1X-2010, "IEEE Standard for Local and Metropolitan Area Networks -- Port-Based Network Access Control".

Protocol Specific Values: there are two:

- A Key Management Domain (KMD).
A string of up to 253 UTF-8 characters that names the transmitting authenticator's key management domain.
- A Network Identifier (NID).
A string of up to 100 UTF-8 characters that identifies a network service. The NID can also be null, indicating the key is associated with a default service.

8.2. KeyTable KDFs

This document requests the establishment of a registry called "KeyTable KDFs". The remainder of this section describes the registry.

All assignments to the KeyTable KDFs registry are made on a First Come First Served basis per Section 4.1 of RFC 5226.

8.3. KeyTable AlgIDs

This document requests establishment of a registry called "KeyTable AlgIDs". The remainder of this section describes the registry.

All assignments to the KeyTable KDFs registry are made on a First

Come First Served basis per Section 4.1 of RFC 5226.

9. Acknowledgments

This document reflects many discussions with many different people over many years. In particular, the authors thank Jari Arkko, Ran Atkinson, Ron Bonica, Ross Callon, Lars Eggert, Pasi Eronen, Adrian Farrel, Sam Hartman, Gregory Lebovitz, Sandy Murphy, Eric Rescorla, Mike Shand, Dave Ward, and Brian Weis for their insights.

10. Informational References

- [RFC3562] Leech, M., "Key Management Considerations for the TCP MD5 Signature Option", RFC 3562, July 2003.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", RFC 4107, BCP 107, June 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

Authors' Addresses

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
EMail: housley@vigilsec.com

Tim Polk
National Institute of Standards and Technology
100 Bureau Drive, Mail Stop 8930
Gaithersburg, MD 20899-8930
USA
EMail: tim.polk@nist.gov

Sam Hartman
Painless Security, LLC
USA
Email: hartmans@painless-security.com

Dacheng Zhang
Huawei

INTERNET-DRAFT

June 29, 2012

China
Email: zhangdacheng@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2013

S. Hartman
Painless Security
D. Zhang
Huawei
July 12, 2012

Operations Model for Router Keying
draft-ietf-karp-ops-model-03.txt

Abstract

Developing an operational and management model for routing protocol security that works across protocols will be critical to the success of routing protocol security efforts. This document discusses issues and begins to consider development of these models.

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."

This Internet-Draft will expire on January 13, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements notation	4
3. Breakdown of KARP configuration	5
3.1. Integrity of the Key Table	6
3.2. Management of Key Table	6
3.3. Interactions with Automated Key Management	7
3.4. VRFs	7
4. Credentials and Authorization	8
4.1. Preshared Keys	9
4.2. Asymmetric Keys	11
4.3. Public Key Infrastructure	11
4.4. The role of Central Servers	12
5. Grouping Peers Together	13
6. Administrator Involvement	15
6.1. Enrollment	15
6.2. Handling Faults	15
7. Upgrade Considerations	17
8. Security Considerations	18
9. Acknowledgments	19
10. References	20
10.1. Normative References	20
10.2. Informative References	20
Authors' Addresses	21

1. Introduction

The KARP working group is designing improvements to the cryptographic authentication of IETF routing protocols. These improvements include improvements to how integrity functions are handled within each protocol as well as designing an automated key management solution.

This document discusses issues to consider when thinking about the operational and management model for KARP. Each implementation will take its own approach to management; this is one area for vendor differentiation. However, it is desirable to have a common baseline for the management objects allowing administrators, security architects and protocol designers to understand what management capabilities they can depend on in heterogeneous environments. Similarly, designing and deploying the protocol will be easier with thought paid to a common operational model. This will also help with the design of NetConf schemas or MIBs later.

2. Requirements 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 [RFC2119].

3. Breakdown of KARP configuration

There are multiple ways of structuring configuration information. One factor to consider is the scope of the configuration information. Several protocols are peer-to-peer routing protocols where a different key could potentially be used for each neighbor. Other protocols require the same group key to be used for all nodes in an administrative domain or routing area. In other cases, the same group key needs to be used for all routers on an interface, but different group keys can be used for each interface.

Within situations where a per-interface, per-area or per-peer key can be used for manually configured long-term keys, that flexibility may not be desirable from an operational standpoint. For example consider OSPF [RFC2328]. Each OSPF link needs to use the same authentication configuration, including the set of keys used for reception and the set of keys used for transmission, but may use different keys for different links. The most general management model would be to configure keys per link. However for deployments where the area uses the same key it would be strongly desirable to configure the key as a property of the area. If the keys are configured per-link, they can get out of sync. In order to support generality of configuration and common operational situations, it would be desirable to have some sort of inheritance where default configurations are made per-area unless overridden per-interface.

As described in [I-D.ietf-karp-crypto-key-table], the cryptographic keys are separated from the interface configuration into their own configuration store. Each routing protocol is responsible for defining the form of the Peer specification used by that protocol. Thus each routing protocol needs to define the scope of keys. For group keying, the Peer specification names the group. A protocol could define a Peer specification indicating the key had a link scope and also a Peer specification for scoping a key to a specific area. For link-scoped keys it is generally best to define a single Peer specification indicating the key has a link scope and to use interface restrictions to restrict the key to the appropriate link.

Operational Requirements: KARP MUST support configuration of keys at the most general scope for the underlying protocol; protocols supporting per-peer keys MUST permit configuration of per-peer keys, protocols supporting per-interface keys MUST support configuration of per-interface keys, and so on. KARP MUST NOT permit configuration of an inappropriate key scope. For example, configuration of separate keys per interface MUST NOT be supported for a protocol requiring per-area keys. This restriction can be enforced by rules specified by each routing protocol for validating key table entries.

3.1. Integrity of the Key Table

The routing key table [I-D.ietf-karp-crypto-key-table] provides a very general mechanism to abstract the storage of keys for routing protocols. To avoid misconfiguration and simplify problem determination, the router MUST verify the internal consistency of entries added to the table. Routing protocols describe how their protocol interacts with the key table including what validation MUST be performed. At a minimum, the router MUST verify:

- o The cryptographic algorithms are valid for the protocol.
- o The key derivation function is valid for the protocol.
- o The direction is valid for the protocol; for example protocols that require the same session key be used in both directions MUST have a direction of both.
- o The peer specification is consistent with the protocol.

Other checks are possible. For example the router could verify that if a key is associated with a peer, that peer is a configured peer for the specified protocol. However, this may be undesirable. It may be desirable to load a key table when some peers have not yet been configured. Also, it may be desirable to share portions of a key table across devices even when their current configuration does not require an adjacency with a particular peer in the interest of uniform configuration or preparing for fail-over.

3.2. Management of Key Table

Several management operations will be quite common. For service provider deployments the configuration management system can simply update the key table. However, for smaller deployments, efficient management operations are important.

As part of adding a new key it is typically desirable to set an expiration time for an old key. The management interface SHOULD provide a mechanism to easily update the expiration time for a current key used with a given peer or interface. Also when adding a key it is desirable to push the key out to nodes that will need it, allowing use for receiving packets then later enabling transmit. This can be accomplished automatically by providing a delay between when a key becomes valid for reception and transmission. However, some environments may not be able to predict when all the necessary changes will be made. In these cases having a mechanism to enable a key for sending is desirable.

The key table's schema supports these operations. However equipment can improve usability by providing convenient functions to effect these common changes.

3.3. Interactions with Automated Key Management

Consideration is required for how an automated key management protocol will assign key IDs for group keys. All members of the group may need to use the same key ID. This requires careful coordination of global key IDs. Interactions with the peer key ID field may make this easier; this requires additional study.

Automated key management protocols also assign keys for single peers. If the key ID is global and needs to be coordinated between the receiver and transmitter, then there is complexity in key management protocols.

3.4. VRFs

Many core and enterprise routers support multiple routing instances. For example a router serving multiple VPNs is likely to have a forwarding/routing instance for each of these VPNs. We need to decide how the key table and other configuration information for KARP interacts with this. The obvious first-order answer is that each routing instance gets its own key table. However, we need to consider how these instances interact with each other and confirm this makes sense.

4. Credentials and Authorization

Several methods for authentication have been proposed for KARP. The simplest is preshared keys used directly as traffic keys. In this mode, the traffic integrity keys are directly configured. This is the mode supported by most of today's routing protocols.

As discussed in [I-D.polk-saag-rtg-auth-keytable], preshared keys can be used as the input to a key derivation function (KDF) to generate traffic keys. For example the TCP Authentication Option (TCP-AO) [RFC5925] derives keys based on the initial TCP session state. Typically a KDF will combine a long-term key with public inputs exchanged as part of the protocol to form fresh session keys. a KDF could potentially be used with some inputs that are configured along with the long-term key. Also, it's possible that inputs to a KDF will be private and exchanged as part of the protocol, although this will be uncommon in KARP's uses of KDFs.

Preshared keys could also be used by an automated key management protocol. In this mode, preshared keys would be used for authentication. However traffic keys would be generated by some key agreement mechanism or transported in a key encryption key derived from the preshared key. This mode may provide better replay protection. Also, in the absence of active attackers, key agreement strategies such as Diffie-Hellman can be used to produce high-quality traffic keys even from relatively weak preshared keys.

Public keys can be used for authentication. The design guide [I-D.ietf-karp-design-guide] describes a mode in which routers have the hashes of peer routers' public keys. In this mode, a traditional public-key infrastructure is not required. The advantage of this mode is that a router only contains its own keying material, limiting the scope of a compromise. The disadvantage is that when a router is added or deleted from the set of authorized routers, all routers that peer need to be updated. Note that self-signed certificates are a common way of communicating public-keys in this style of authentication.

Certificates signed by a certification authority or some other PKI could be used. The advantage of this approach is that routers may not need to be directly updated when peers are added or removed. The disadvantage is that more complexity and cost is required.

Each of these approaches has a different set of management and operational requirements. Key differences include how authorization is handled and how identity works. This section discusses these differences.

4.1. Preshared Keys

In the protocol, manual preshared keys are either unnamed or named by a small integer (typically 16 or 32 bits) key ID. Implementations that support multiple keys for protocols that have no names for keys need to try all possible keys before deciding a packet cannot be validated [RFC4808]. Typically key IDs are names used by one group or peer.

Manual preshared keys are often known by a group of peers rather than just one other peer. This is an interesting security property: unlike with digitally signed messages or protocols where symmetric keys are known only to two parties, it is impossible to identify the peer sending a message cryptographically. However, it is possible to show that the sender of a message is one of the parties who knows the preshared key. Within the routing threat model the peer sending a message can be identified only because peers are trusted and thus can be assumed to correctly label the packets they send. This contrasts with a protocol where cryptographic means such as digital signatures are used to verify the origin of a message. As a consequence, authorization is typically based on knowing the preshared key rather than on being a particular peer. Note that once an authorization decision is made, the peer can assert its identity; this identity is trusted just as the routing information from the peer is trusted. Doing an additional check for authorization based on the identity included in the packet would provide little value: an attacker who somehow had the key could claim the identity of an authorized peer and an attacker without the key should be unable to claim the identity of any peer. Such a check is not required by the KARP threat model: inside attacks are not in scope.

Preshared keys used with key derivation function similarly to manual preshared keys. However to form the actual traffic keys, session or peer specific information is combined with the key. From an authorization standpoint, the derivation key works the same as a manual key. An additional routing protocol step or transport step forms the key that is actually used.

Preshared keys that are used via automatic key management have not been specified for KARP. Their naming and authorization may differ from existing uses of preshared keys in routing protocols. In particular, such keys may end up being known only by two peers. Alternatively they may also be known by a group of peers. Authorization could potentially be based on peer identity, although it is likely that knowing the right key will be sufficient. There does not appear to be a compelling reason to decouple the authorization of a key for some purpose from authorization of peers holding that key to perform the authorized function.

Care needs to be taken when symmetric keys are used for multiple purposes. Consider the implications of using the same preshared key for two interfaces: it becomes impossible to cryptographically distinguish a router on one interface from a router on another interface. So, a router that is trusted to participate in a routing protocol on one interface becomes implicitly trusted for the other interfaces that share the key. For many cases, such as link-state routers in the same routing area, there is no significant advantage that an attacker could gain from this trust within the KARP threat model. However, distance-vector protocols, such as BGP and RIP, permit routes to be filtered across a trust boundary. For these protocols, participation in one interface might be more advantageous than another. Operationally, when this trust distinction is important to a deployment, different keys need to be used on each side of the trust boundary. Key derivation can help prevent this problem in cases of accidental misconfiguration. However, key derivation cannot protect against a situation where a system was incorrectly trusted to have the key used to perform the derivation. To the extent that there are multiple zones of trust and a routing protocol is determining whether a particular router is within a certain zone, the question of untrusted actors is within the scope of the routing threat model.

Key derivation can be part of a management solution to a desire to have multiple keys for different zones of trust. A master key could be combined with peer, link or area identifiers to form a router-specific preshared key that is loaded onto routers. Provided that the master key lives only on the management server and not the individual routers, trust is preserved. However in many cases, generating independent keys for the routers and storing the result is more practical. If the master key were somehow compromised, all the resulting keys would need to be changed. However if independent keys are used, the scope of a compromise may be more limited.

More subtle problems with key separation can appear in protocol design. Two protocols that use the same traffic keys may work together in unintended ways permitting one protocol to be used to attack the other. Consider two hypothetical protocols. Protocol A starts its messages with a set of extensions that are ignored if not understood. Protocol B has a fixed header at the beginning of its messages but ends messages with extension information. It may be that the same message is valid both as part of protocol A and protocol B. An attacker may be able to gain an advantage by getting a router to generate this message with one protocol under situations where the other protocol would not generate the message. This hypothetical example is overly simplistic; real-world attacks exploiting key separation weaknesses tend to be complicated and involve specific properties of the cryptographic functions involved.

The key point is that whenever the same key is used in multiple protocols, attacks may be possible. All the involved protocols need to be analyzed to understand the scope of potential attacks.

Key separation attacks interact with the KARP operational model in a number of ways. Administrators need to be aware of situations where using the same manual traffic key with two different protocols (or the same protocol in different contexts) creates attack opportunities. Design teams should consider how their protocol might interact with other routing protocols and describe any attacks discovered so that administrators can understand the operational implications. When designing automated key management or new cryptographic authentication within routing protocols, we need to be aware that administrators expect to be able to use the same preshared keys in multiple contexts. As a result, we should use appropriate key derivation functions so that different cryptographic keys are used even when the same initial input key is used.

4.2. Asymmetric Keys

Outside of a PKI, public keys are expected to be known by the hash of a key or (potentially self-signed) certificate. The Session Description Protocol provides a standardized mechanism for naming keys (in that case certificates) based on hashes (section 5 [RFC4572]). KARP SHOULD adopt this approach or another approach already standardized within the IETF rather than inventing a new mechanism for naming public keys.

A public key is typically expected to belong to one peer. As a peer generates new keys and retires old keys, its public key may change. For this reason, from a management standpoint, peers should be thought of as associated with multiple public keys rather than as containing a single public key hash as an attribute of the peer object.

Authorization of public keys could be done either by key hash or by peer identity. Performing authorizations by peer identity should make it easier to update the key of a peer without risk of losing authorizations for that peer. However management interfaces need to be carefully designed to avoid making this extra level of indirection complicated for operators.

4.3. Public Key Infrastructure

When a PKI is used, certificates are used. The certificate binds a key to a name of a peer. The key management protocol is responsible for exchanging certificates and validating them to a trust anchor.

Authorization needs to be done in terms of peer identities not in terms of keys. One reason for this is that when a peer changes its key, the new certificate needs to be sufficient for authentication to continue functioning even though the key has never been seen before.

Potentially authorization could be performed in terms of groups of peers rather than single peers. An advantage of this is that it may be possible to add a new router with no authentication related configuration of the peers of that router. For example, a domain could decide that any router with a particular keyPurposeID signed by the organization's certificate authority is permitted to join the IGP. Just as in configurations where cryptographic authentication is not used, automatic discovery of this router can establish appropriate adjacencies.

Assuming that potentially self-signed certificates are used by routers that wish to use public keys but that do not need a PKI, then PKI and the infrastructureless mode of public-key operation described in the previous section can work well together. One router could identify its peers based on names and use certificate validation. Another router could use hashes of certificates. This could be very useful for border routers between two organizations. Smaller organizations could use public keys and larger organizations could use PKI.

4.4. The role of Central Servers

An area to explore is the role of central servers like RADIUS or directories. As discussed in the design-guide, a system where keys are pushed by a central management system is undesirable as an end result for KARP. However central servers may play a role in authorization and key rollover. For example a node could send a hash of a public key to a RADIUS server.

If central servers do play a role it will be critical to make sure that they are not required during routine operation or a cold-start of a network. They are more likely to play a role in enrollment of new peers or key migration/compromise.

Another area where central servers may play a role is for group key agreement. As an example, [I-D.liu-ospfv3-automated-keying-req] discusses the potential need for key agreement servers in OSPF. Other routing protocols that use multicast or broadcast such as IS-IS are likely to need a similar approach.

5. Grouping Peers Together

One significant management consideration will be the grouping of management objects necessary to determine who is authorized to act as a peer for a given routing action. As discussed previously, the following objects are potentially required:

- o Key objects are required. Symmetric keys may be preshared. Asymmetric public keys may be used directly for authorization as well. During key transitions more than one key may refer to a given peer. Group preshared keys may refer to multiple peers.
- o A peer is a router that this router might wish to communicate with. Peers may be identified by names or keys.
- o Groups of peers may be authorized for a given routing protocol.

Establishing a management model is difficult because of the complex relationships between each set of objects. As discussed there may be more than one key for a peer. However in the preshared key case, there may be more than one peer for a key. This is true both for group security association protocols such as an IGP or one-to-one protocols where the same key is used administratively. In some of these situations, it may be undesirable to explicitly enumerate the peers in the configuration; for example IGP peers are auto-discovered for broadcast links but not for non-broadcast multi-access links.

Peers may be identified either by name or key. If peers are identified by key it is probably strongly desirable from an operational standpoint to consider any peer identifiers or name to be a local matter and not require the names or identifiers to be synchronized. Obviously if peers are identified by names (for example with certificates in a PKI), identifiers need to be synchronized between the authorized peer and the peer making the authorization decision.

In many cases peers will explicitly be identified. In these cases it is possible to attach the authorization information (keys or identifiers) to the peer's configuration object. Two cases do not involve enumerating peers. The first is the case where preshared keys are shared among a group of peers. It is likely that this case can be treated from a management standpoint as a single peer representing all the peers that share the keys. The other case is one where certificates in a PKI are used to introduce peers to a router. In this case, rather than configuring peers, , the router needs to be configured with information on what certificates represent acceptable peers.

Another consideration is what routing protocols share peers. For example it may be common for LDP peers to also be peers of some other routing protocol. Also, RSVP-TE may be associated with some TE-based IGP. In some of these cases it would be desirable to use the same authorization information for both routing protocols.

In order to develop a management model for authorization, the working group needs to consider several questions. What protocols support auto-discovery of peers? What protocols require more configuration of a peer than simply the peer's authorization information and network address? What management operations are going to be common as security information for peers is configured and updated? What operations will be common while performing key transitions or while migrating to new security technologies?

6. Administrator Involvement

One key operational question is what areas will administrator involvement be required. Likely areas where involvement may be useful includes enrollment of new peers. Fault recovery should also be considered.

6.1. Enrollment

One area where the management of routing security needs to be optimized is the deployment of a new router. In some cases a new router may be deployed on an existing network where routing to management servers is already available. In other cases, routers may be deployed as part of connecting or creating a site. Here, the router and infrastructure may not be available until the router has securely authenticated. This problem is similar to the problem of getting initial configuration of routing instances onto the router. However, especially in cases where asymmetric keys or per-peer preshared keys are used, the configuration of other routers needs to be modified to bring up the security association. Also, there has been discussion of generating keys on routers and not allowing them to leave devices. This also impacts what strategies are possible. For example this might mean that routers need to be booted in a secure environment where keys can be generated, and public keys copied to a management server to push out the new public key to potential peers. Then, the router needs to be packaged, moved to where it will be deployed and set up. Alternatives are possible; it is critical that we understand how what we propose impacts operators.

We need to work through examples with operators familiar with specific real-world deployment practices and understand how proposed security mechanisms will interact with these practices.

6.2. Handling Faults

Faults may interact with operational practice in at least two ways. First, security solutions may introduce faults. For example if certificates expire in a PKI, previous adjacencies may no longer form. Operational practice will require a way of repairing these errors. This may end up being very similar to deploying a router that is connecting a new site as the security fault may have partitioned the network. However, unlike a new deployment, the event is unplanned. Strategies such as configuring a router and shipping it to a site may not be appropriate for recovering a fault even though they may be more useful for new deployments.

Notifications will play a critical role in avoiding security faults. Implementations SHOULD use appropriate mechanisms to notify operators

as security resources are about to expire. Notifications can include messages to consoles, logged events, SNMP traps, or notifications within a routing protocol. One strategy is to have increasing escalations of notifications.

Monitoring will also play a important role in avoiding security faults such as certificate expiration. However, the protocols MUST still have adequate operational mechanisms to recover from these situations. Also, some faults, such as those resulting from a compromise or actual attack on a facility are inherent and may not be prevented.

A second class of faults is equipment faults that impact security. For example if keys are stored on a router and never moved from that device, failure of a router implies a need to update security provisioning on the replacement router and its peers.

To address these operational considerations, we should identify circumstances surrounding recovery from today's faults and understand how protocols will impact mechanisms used today.

7. Upgrade Considerations

It needs to be possible to deploy automated key management in an organization without either having to disable existing security or disrupting routing. As a result, it needs to be possible to perform a phased upgrade from manual keying to automated key management. This upgrade procedure needs to be easy and have a very low risk of disrupting routing. Today, many operators do not update keys because the perceived risk of an attack is lower than the complexity of and update and risk of routing disruptions.

For peer-to-peer protocols such as BGP, this can be relatively easy. First, code that supports automated key management needs to be loaded on both peers. Then the adjacency can be upgraded. The configuration can be updated to switch to automated key management when the second router reboots. Alternatively, if the key management protocols involved can detect that both peers now support automated key management, then a key can potentially be negotiated for an existing session.

The situation is more complicated for multicast protocols. It's probably not reasonable to bring down an entire link to reconfigure it as using automated key management. Two approaches should be considered. One is to support key table rows supporting the automated key management and manually configured keying for the same link at the same time. Coordinating this may be tricky. Another possibility is for the automated key management protocol to actually select the same traffic key that is being used manually. This could potentially be accomplished by having an option in the key management protocol to export the current manual group key through the automated key management protocol. Then after all nodes are configured with automated key management, manual key entries can be removed. The next re-key after all nodes have manual entries removed will generate a new fresh key.

8. Security Considerations

This document does not define a protocol. It does discuss the operational and management implications of several security technologies.

9. Acknowledgments

Funding for Sam Hartman's work on this memo is provided by Huawei.

The authors would like to thank Gregory Lebovitz, Russ White and Bill Atwood for valuable reviews.

10. References

10.1. Normative References

- [I-D.ietf-karp-crypto-key-table]
Housley, R., Polk, T., Hartman, S., and D. Zhang,
"Database of Long-Lived Symmetric Cryptographic Keys",
draft-ietf-karp-crypto-key-table-03 (work in progress),
June 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

- [I-D.ietf-karp-design-guide]
Lebovitz, G. and M. Bhatia, "Keying and Authentication for
Routing Protocols (KARP) Design Guidelines",
draft-ietf-karp-design-guide-10 (work in progress),
December 2011.
- [I-D.liu-ospfv3-automated-keying-req]
Liu, Y., "OSPFv3 Automated Group Keying Requirements",
draft-liu-ospfv3-automated-keying-req-01 (work in
progress), July 2007.
- [I-D.polk-saag-rtg-auth-keytable]
Polk, T. and R. Housley, "Routing Authentication Using A
Database of Long-Lived Cryptographic Keys",
draft-polk-saag-rtg-auth-keytable-05 (work in progress),
November 2010.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC4572] Lennox, J., "Connection-Oriented Media Transport over the
Transport Layer Security (TLS) Protocol in the Session
Description Protocol (SDP)", RFC 4572, July 2006.
- [RFC4808] Bellovin, S., "Key Change Strategies for TCP-MD5",
RFC 4808, March 2007.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
Authentication Option", RFC 5925, June 2010.

Authors' Addresses

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu

Dacheng Zhang
Huawei

Email: zhangdacheng@huawei.com

