

IPsecME Working Group
INTERNET-DRAFT

Intended Status: Proposed Standard
Expires: February 19, 2014

Y. Mao
Z. Wang
Hangzhou H3C Tech. Co., Ltd.
V. Manral
HP
August 18, 2013

Auto Discovery VPN Protocol
draft-mao-ipsecme-ad-vpn-protocol-02

Abstract

This document describes the Auto Discovery VPN (ADVPN) protocol, the use case and problem statement for which is described in [ADVPN_Problem]. The ADVPN protocol is used for enabling a large of number of entities to communicate directly among the peers, with minimal configuration and operator intervention. The solution uses IPsec[RFC4301] to protect communication between the peers.

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 and License Notice

Copyright (c) 2013 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. ADVPN Protocol Overview	3
1.2. Terminology	5
1.3. Conventions Used in This Document	5
2. Design Overview	5
3. How ADVPN Works	7
4. ADVPN protocol	8
4.1. Client Information Registration	8
4.2. Client Information Resolution	9
4.3. Private Network Information Management	10
4.4. Shortcut Decision	11
4.5. Redirect protocol	11
4.6. Keepalive protocol	12
4.7. Session Protocol	12
5. ADVPN Message Formats	13
5.1. ADVPN Fixed Header	13
5.2. Mandatory Part	15
5.2.1. Client Identification Header	15
5.2.2. Session Header	16
5.3. Payload Part	17
5.3.1. Client Information Payload	17
5.3.2. Destination Client Payload	19
5.3.3. Network Information Payload	20
5.3.4. Traffic Flow Payload	21
5.3.5. Keepalive Parameter Payload	23
5.3.6. Redirect Payload	23
6. Implementation Status	25
7. Security Considerations	26
8. IANA Considerations	26
9. Acknowledgments	26
10. References	26
10.1. Normative References	26
10.2. Informative References	27
Appendix A. Comparison Against ADVPN Requirements	27
Authors' Addresses	29

1. Introduction

The large scale deployment of IPsec[RFC4301] leads to lots of difficulties, such as configuration for each tunnel, adding or removing IPsec peer, etc. all need a lot of configuration/reconfiguration. Therefore, a protocol to establish IPsec tunnel dynamically without having the large overhead of configuration is needed. Auto Discovery VPN Problem Statement and Requirement [ADVPN_Problem] defines all requirements for the large scale IPsec deployment problem. This document defines the Auto Discovery VPN (ADVPN) protocol to satisfy these requirements.

1.1. ADVPN Protocol Overview

The overall ADVPN solution has control plane and data plane elements. The ADVPN protocol operates in the control plane and uses the standard IPsec [RFC4301] for the data plane. The IPsec data plane establishes and protects all the traffic in the ADVPN network.

The ADVPN protocol is a client and server protocol. The ADVPN Client(ADC) registers its information to the ADVPN Server(ADS). When the ADC wants to establish an IPsec tunnel with another ADC, the ADC requests and queries another ADC's information on the ADS. The ADVPN Client(ADC) is the forwarding device in the data plane. ADC implements IPsec to protect its own traffic or the traffic flowing through ADC.

The ADVPN Server(ADS) is ADVPN information controller, which is responsible for collection, maintenance and distribution of ADVPN Client(ADC) information and the control policy. The client information is registered by the ADVPN client. The client information includes private IP address, public IP address and private network information etc. The control policy is used to decide the topology should be star topology or full mesh topology, and the control policy is pushed to hub ADC from ADS.

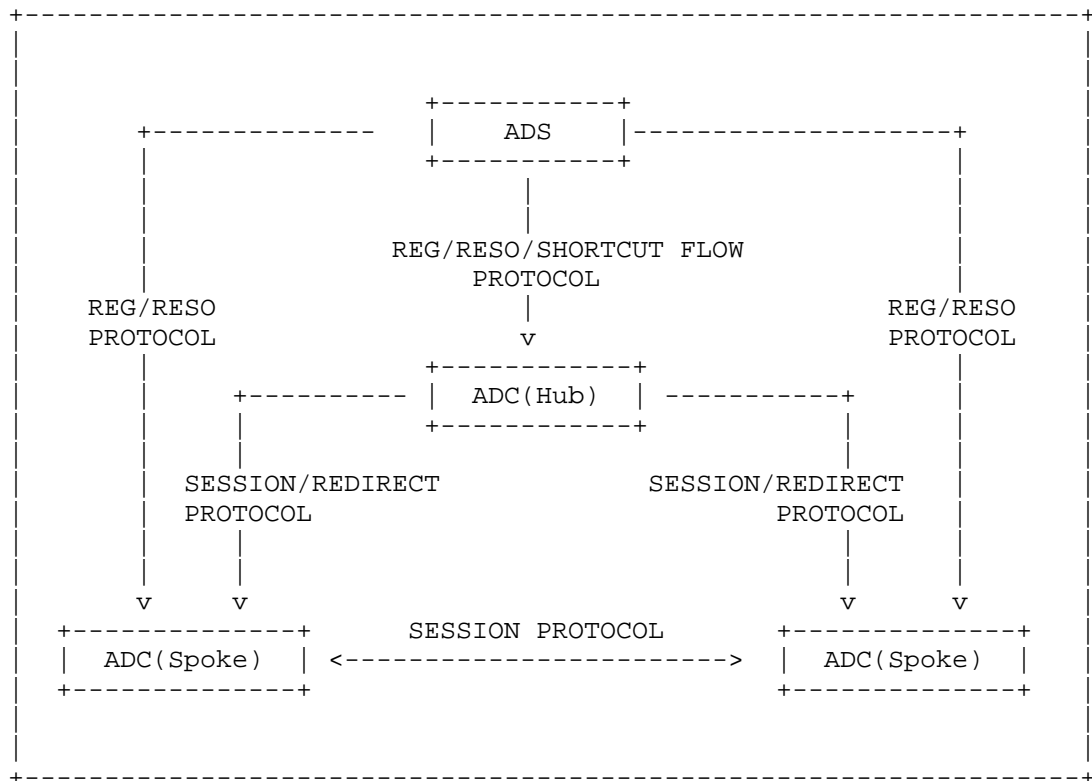


Figure 1. ADVPN Protocol Model

In this diagram, ADVPN protocol is implemented between ADC and ADS. All the ADCs register information on the ADS. When the ADC tries to build a direct IPsec tunnel with another ADC, it sends the Resolution message to ADS to query the information. In addition, to allow the shortcut path establishment between the ADCs, the ADS sends the Shortcut Traffic Flow message to the hub ADCs. Which ADC is hub ADC SHOULD be specified in the ADS by administrator.

ADVPN protocol can also be implemented between ADC and ADC. The ADC sends the Session message to another ADC to transfer ADC information; otherwise the other ADC has to query the ADS if there is a reverse traffic, which may not be practical in some cases. The hub ADC sends Redirect message to spoke ADC to trigger the spoke ADC send Resolution message to ADS.

With ADVPN protocol, the IPsec gateways and endpoints can obtain the IPsec peer's remote address from ADS. Therefore, establishing IPsec tunnel between them can scale well.

1.2. Terminology

Most terminology has been specified in the [ADVPN_Problem]. However, there are also some additional terminology in this document needs to be clarified.

ADVPN Server - This is an entity as ADVPN network controller. It maintains ADVPN client information database. It also can decide whether the spoke can directly communicate with the other spoke.

ADVPN Client - This is an entity as ADVPN peer. It registers its information to ADVPN server.

Hub ADC - This ADC is hub in the forwarding path.

Spoke ADC - This ADC is spoke in the forwarding path.

Private IP Address - Each ADVPN client has a logical private IP address. This address is static and as identity of ADVPN client. Through the private IP address, the Public IP address is discovered.

Public IP Address - This is IPsec peer address. This address can be dynamic and attached with Private IP Address. There is private IP address to public IP address mapping.

Private Network Information - This information includes the network behind the spoke and the next hop which is private IP address.

Shortcut Path - This is direct connectivity between spoke and spoke.

ADVPN Network - This is network composed of ADVPN peers

Source ADVPN Peer - This ADVPN peer is the initiator to establish IPsec tunnel.

Destination ADVPN Peer - This ADVPN peer is the responder to establish IPsec tunnel.

1.3. Conventions Used in This Document

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].

2. Design Overview

In the ADVPN network, there are thousands of ADVPN peers. It is difficult to pre-configure the SPD and PAD for each ADVPN peer

manually. In the use cases in the [ADVPN_Problem], the endpoints and gateways can use a dynamic IP address, it is impossible to specify the IP address towards this ADVPN peer in the remote ADVPN peer. Therefore, the design goal of ADVPN protocol described in this document is each ADVPN peer only configures its own SPD and PAD. If the ADVPN peer tries to setup an IPsec tunnel with another ADVPN peer, it queries the ADS and obtains the client information of another peer.

To discover the remote ADVPN peer, a private IP address is used as the search key. The search key is private IP address. The private IP address is logical Virtual Private Network (VPN) IP address. Although IPsec peer address is dynamic, the private IP address is static. The private IP address is also next hop towards the network behind the ADVPN peer. When the traffic flows through the source ADVPN peer, routing table is looked up, the next hop is the private IP address of destination ADVPN peer. The source ADVPN peer looks up session table to find the public IP address of destination ADVPN peer. Session table is the ADC information cache in the forwarding path, and it has the private IP address to public IP address mapping.

The private IP address with the private network information SHOULD be transferred via routing protocol (e.g. OSPF, BGP or RIP) or another means in the ADVPN network. A routing protocol can run over the IPsec tunnel between the ADCs and ADS. The routing protocol helps distribute routing information to all ADC's towards about the destination network behind the ADCs.

After a spoke ADC finishes the registration on ADS, this ADC also obtains the information of hub ADC from ADS. An IPsec tunnel is then established automatically between the spoke ADC and hub ADC. The routing protocol messages are protected and transferred in this IPsec tunnel and exchanged between the spoke ADC and hub ADC.

In the full mesh ADVPN network, there are tens of thousands of ADVPN peers. The spokes cannot be populated with a full routing table due to constraints on the device capabilities; therefore the network is left as a hub-and-spoke network initially. After routing information exchanges between the hub and spokes, routing table in the source ADVPN peer has the private IP address of hub ADVPN peer as the next hop to networks behind the destination ADVPN peer. Therefore, the traffic from source ADVPN peer to destination ADVPN peer is forwarded to hub first.

In order to have direct connectivity between the ADVPN peers, the ADVPN peer queries the direct route information on ADS. The spoke ADC SHOULD register its private IP address and network behind the ADC to the ADS. The private IP address is the next hop to network behind the

ADC. When the traffic is transferred through hub, the hub sends a redirect message to source ADVPN peer. When the source ADVPN peer receiving Redirect message, it send a Resolution message to ADS to query the direct route information and ADC information for the destination ADVPN peer. Receiving the resolution reply message, the source ADVPN peer adds a direct route in the route table and setup a direct IPsec tunnel with the destination ADVPN peer.

The hub decides whether the spoke can be allowed to have a direct communication with other spoke. The decision depends on the control policy pushed by the ADS after the hub ADC finishes registration in the ADS. The control policy is flow information. If the traffic matches the flow information, the hub sends a redirect message to source ADVPN peer to find a shortcut path to destination ADVPN peer.

3. How ADVPN Works

In the ADVPN solution, ADVPN protocol uses the IPsec protocol [RFC4301] data plane, as well as routing protocols to fulfill the ADVPN function. This section describes the process to establish a shortcut spoke-to-spoke IPsec tunnel in a mesh topology.

1. When the ADC device comes up, it registers its information to ADS. The information includes the private IP address, the public IP address and the network behind the ADC.

2. The ADS sends the registration reply message to the ADC. The spoke ADC obtains the information of hub ADC. The ADC creates the hub session in the session table. After that, the spoke ADC establishes an IPsec tunnel with hub ADC.

3. The ADS sends Shortcut Flow message to hub ADC in order to determine whether sending redirect message or not.

4. The spoke ADC send a Session Setup message to hub ADC protected by IPsec tunnel, the hub ADC has the spoke ADC's information.

5. All the route protocol packets run over IPsec tunnel between the spoke ADC and hub ADC. The route protocol packet is copied and sent to hub ADC. The ADVPN network has spoke-hub topology.

6. When the traffic towards destination ADVPN peer arrives in the source ADVPN peer device, from the routing table, the next hop is the private IP address of hub ADVPN peer. Match the private IP address in the session table to obtain the public IP address of hub ADVPN peer. By the public IP address, the spoke-to-hub IPsec SA is chosen to encapsulate the traffic.

7. The traffic arrives in the hub, after processing IPsec packet, the hub looks up routing table to determine if incoming and outgoing interface is in the same ADVPN network. If it is, this traffic is transferred through hub towards the destination spoke and there is shortcut path between them. If the traffic matches the Shortcut Flow table, the hub send a redirect message to source ADVPN peer.

8. The source ADVPN peer receives the redirect message, it sends Resolution Request message with destination IP address to ADS. ADS lookes in the ADC information database to find out the next hop to the destination IP address and related network information. The ADS sends a Resolution Response message to the source ADVPN peer.

9. The source ADVPN peer receives the Resolution Response message. Firstly, a route towards destination network is added into the route table. Secondly, the source ADVPN peer establishes an IPsec tunnel with the destination ADVPN peer. Lastly, the source ADVPN peer send a session message to destination ADVPN peer. The destination ADVPN peer can add the reverse route in its route table and the ADC information in session table.

4. ADVPN protocol

ADVPN protocol listens and sends on UDP port 2013(pending assignment by IANA). Since the UDP is a datagram(unreliable) protocol, all messages in ADVPN exist in pairs: a request and a response.

In the following descriptions, the payloads contained in the message are indicated by names as listed below.

Notation	Description

HDR	ADVPN header
CIDHdr	Client Identification Header
SessHdr	Session Header
CI	Client Information payload
DC	Destination Client Payload
NI	Network Information Payload
TF	Traffic Flow Payload
KP	Keepalive Parameter Payload
Red	Redirect Payload

The details of the contents of each header and payload are described in section 5. Payloads that may optionally appear will be shown in brackets, such as [CI]; this indicates that a Client Information payload can optionally be included.

4.1. Client Information Registration

The ADC sends a Registration Request message to ADS to register its ADVPN information. The ADVPN information is contained in the message using CI payload and NI payload. When receiving the Registration Request message, ADS adds this ADVPN client information to ADC information database and sends a Registration Request Response message to the ADC. The Registration Request Response message includes KP payload, which make ADC send a keepalive message to ADS periodically after registration. If the hub ADC has been registered in the ADS, CI payload SHOULD be also contained with hub's ADC information.

The registration messages with payloads are as follows:

Client		Server

HDR, CIDHdr, CI, [NI]	-->	
	<--	HDR, CIDHdr, KP, [CI]

When the hub ADC is registered, a Hub Information message with CI payload should be sent to all the ADCs in the registration status. The hub Information Acknowledgement message has no payload, only contains ADVPN header and Client Identification Header.

The hub messages with payloads are as follows:

Client		Server

	<--	HDR, CIDHdr, CI
HDR, CIDHdr	-->	

4.2. Client Information Resolution

There are two scenarios that the ADC needs to send Resolution Request message to ADS to query the client information. 1. During the packet processing in the forwarding path, the session table is consulted with private IP address, If there is no session, the spoke ADC sends a Resolution Request message to ADS to get the remote ADC's information related with the private IP address. Before a Resolution Request Response comes, the data packets are forwarded to hub. Note that a Resolution Request message for the private IP address MUST NOT be triggered by every packet. 2. The spoke ADC received a Redirect message from the hub ADC, the spoke ADC sends a Resolution Request to ADS to get the remote ADC's information with the destination address.

The Resolution Request message includes DC payload. If the next hop address and destination address both are contained in the payload, the ADS should loop up ADC information database with destination address firstly.

When the ADS receives a Resolution Request packet, it searches the ADC information database by private IP address or destination address. If an ADC is found, a Resolution Response message containing CI payload and NI payload is send to the spoke ADC. If there is no match, the error code is set in the header and no payload is included in the response message.

The resolution messages with payloads are as follows:

Client		Server

HDR, CIDHdr, DC	-->	
	<--	HDR, CIDHdr, [CI], [NI]

After receiving the remote ADC's information, the ADC create session cache based on the information in the CI payload. If there is NI payload, the route towards the destination address is added in the route table.

4.3. Private Network Information Management

To help the ADC find a shortcut path, the private network information database is collected and distributed by the ADS. The private network information is like a private network route table. The ADC can query the next hop towards the private network.

In the Registration Request message, the private network information can include in the NI payload and registered to ADS. After registration on the ADS, if the ADC's network information is changed(e.g. add, delete or modify), a Network Information Registration packet including the NI payload is send to the ADS to update the private network information database. After updating, a Network Information Registration Response is send back to the ADC, the response message has no payload, only contains ADVPN header and Client Identification Header.

If the private network information is updated on ADS, a Network Information Update message containing the NI payload MUST be send to all ADCs in order to these ADCs have the correct route in their route table. All the ADC MUST send back a Network Information Update Response message to ADS, the response message has no payload, only contains ADVPN header and Client Identification Header.

The private network information can also be transferred in the Session Setup message. In the session establish process, the private network route can be added in the remote ADC's route table in order to avoid ADS query for reverse traffic.

If receiving a Session Delete message from remote ADC, the ADC MUST tear down the session and clear the route added by the Session Setup message.

4.4. Shortcut Decision

Whether the shortcut path can be established depends on the control policy in the ADS. The ADS defines the flow information to allow the direct connectivity. After the hub ADC finishes the registration, the ADS send the Shortcut Flow message contains TF payload to hub ADC. After receiving the message and add the flow information to shortcut flow table, the hub ADC send back Shortcut Flow Acknowledgement message to ADS.

The hub ADC has a shortcut flow table to match the traffic through hub in the ADVPN network. If there is a match, the hub ADC send a Redirect message to source ADVPN peer.

If the control policy is changed(e.g. add, delete or modify) on the ADS, the ADS MUST send a Shortcut Flow message to the hub ADC to update the shortcut flow table. If the shortcut flow item is deleted, the hub ADC send a Redirect message to source ADVPN peer to tear down the direct IPsec Tunnel.

4.5. Redirect protocol

The hub ADC sends a Redirect message to spoke ADC means there is another path for traffic forwarding. In the hub ADC, when the traffic matches the shortcut flow table, a Redirect message containing the Red payload is sent to spoke ADC. The spoke ADC receives the Redirect message, sends a Redirect Response message to hub ADC and subsequently sends a Resolution message to ADS to query the shortcut information. The Redirect Response message has no payload, only contains ADVPN header and Session Header.

The redirect messages with payloads are as follows:

Hub Client		Spoke Client

HDR, SessHdr, Red	-->	
	<--	HDR, SessHdr

If a shortcut flow is deleted in the hub ADC, the hub ADC send a Redirect message to the ADCs which has received the Redirect message to trigger the shortcut resolution before. When receiving this Redirect message, the spoke ADC deletes the route related with the flow information. The shortcut IPsec tunnel is cleared up and the

traffic goes through the hub to transfer.

4.6. Keepalive protocol

After the ADC finishes registration on the ADS, the ADC SHOULD send a Keepalive Request message to ADS to prove its liveness. The Keepalive Request message contains the ADVPN header and Client Identification Header. The number of retries and length of timeouts depend on the keepalive parameter pushed from ADS by the Registration Response message. If the number of retry attempts is reached but the ADC does not receive Keepalive Response message, the connection between ADC and ADS is considered broken. The ADC clears up all the resources and registered to ADS again.

On ADS, receipt of any ADVPN message from ADC can prove the ADC's liveness. If the timeout is reached while the ADS does not receive Registration Response message, the ADS will clear all ADC information from ADC information database. If the ADC is hub, the ADS sends the Hub Information message to all the ADCs to notify the hub removing.

4.7. Session Protocol

Session table is an remote ADC information cache in the forwarding path. It is composed of private IP address, public IP address, and the index of IPsec SA. In the forwarding process, the session table MUST be consulted with private IP address. Each session matches to only one IPsec SA.

The function of session protocol is to facilitate the forwarding process, the session information transported to the remote peer avoids to query the ADS when reverse traffic flows.

There are two type sessions: permanent session established with hub, and dynamic session established between spokes. The permanent session is static and established after the spoke ADC and hub ADC finish registration. The dynamic session will be deleted if there is no traffic between spokes,

The permanent session is established between the spoke and hub or between the hubs. After receiving the Hub's ADC information from ADS by Registration Response message or Hub Information message, the spoke establishes an IPsec Tunnel with Hub firstly and then sends a Session Setup message to the hub, which is protected via IPsec tunnel. When receiving the Session Setup message, the hub ADC create a session for spoke ADC with the spoke's ADC information, and a Session Setup Response message will be send back to the spoke ADC.

When the spoke ADC receives the Resolution Response message for

shortcut path and obtains the remote spoke ADC's information, the spoke creates a IPsec tunnel with the remote spoke. After that, the spoke sends a Session Setup message to the remote spoke. The remote spoke creates a session information towards the spoke and sends back a Session Setup Response message. If there is private network information in the source spoke, a NI payload SHOULD be contained in the Session Setup message. The Session Setup Response message has no payload, only contains ADVPN header and Session Header.

The session messages with payloads are as follows:

Client		Client

HDR, SessHdr, [NI]	-->	
	<--	HDR, SessHdr

If there is no traffic in the spoke-to-spoke session, the spoke will send a Session Delete message to the remote spoke to remove the session item. After the session is removed, the IPsec tunnel is also cleared up.

5. ADVPN Message Formats

This section describes the format of ADVPN message. ADVPN messages begin immediately following the UDP header. An ADVPN message is composed of a Fixed Part, a Mandatory Part and a Payload Part. The Fixed Part is common to all ADVPN message types. The Mandatory Part MUST be present, but varies depending on message type. The Payload Part also varies depending on message type.

The length of the Fixed Part is fixed at 12 octets. The length of the Mandatory Part is determined on message type. The Payload Part length depends on the payload type.

5.1. ADVPN Fixed Header

The Fixed Part of the ADVPN message contains those elements of the ADVPN message which are always present and do not vary in size with the type of message.

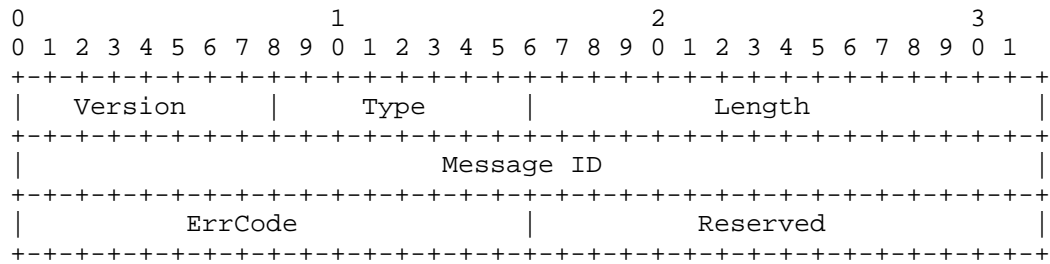


Figure 2. ADVPN Header Format

- o Version (1 octet) -- Indicates the version of the ADVPN protocol in use. Implementations based on this version of ADVPN MUST set the version to 0.
- o Type (1 octet) -- Indicates the type of ADVPN message being used.

Message Type	Value
Registration Request	1
Registration Response	2
Resolution Request	3
Resolution Response	4
Delete Request	5
Delete Response	6
Keepalive Request	7
Keepalive Response	8
Network Information Registration	9
Network Information Response	10
Shortcut Flow	11
Shortcut Flow Acknowledgement	12
Hub Information	13
Hub Information Acknowledgement	14
Redirect	15
Redirect Acknowledgements	16
Session Setup Request	17
Session Setup Response	18
Session Keepalive Request	19
Session Keepalive Response	20
Session Delete Request	21
Session Delete Response	22
Session Network Information Request	23
Session Network Information Response	24

- o Length (2 octet) -- Length of total message beginning with the Version element in octets.

- o Message ID (4 octet) -- Used to provide a unique identifier for the information contained in a Request message. This value is copied directly from a Request message into the Response message. When a sender receives the Response packet, it will match the Message ID with the Request packet of local sent. When a match is found then the Request is considered to be acknowledged.

The value is incremented each time a new Request message is sent. The same value **MUST** be used when resending a Request message. It is **RECOMMENDED** that the initial value for this number be 0.

- o ErrCode (2 octet) -- An error code indicating the type of error detected, chosen from the follow list:

Error description	Code
-----	-----
No ADC Found	1
Management Reject	2
Insufficient Resources	3

- o Reserved (2 octet) -- **MUST** be zero.

5.2. Mandatory Part

Different Mandatory Part of the ADVPN message exists in different message types, and is behind the ADVPN Header.

5.2.1. Client Identification Header

The Client Identification Header, denoted CIhdr in this document, is contained in the messages that are sent and received between ADC an ADS.

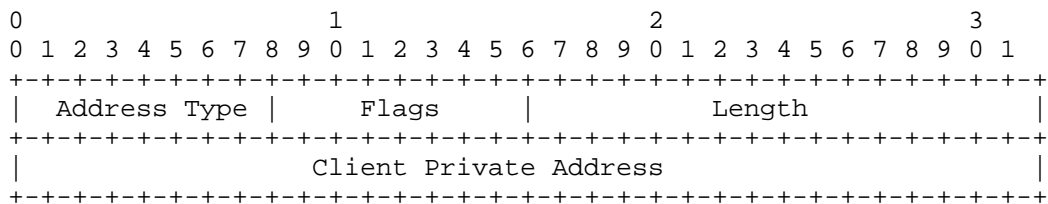


Figure 3. Client Identification Header Format

- o Address Type (1 octet) -- Identifies the address type of the client private Address.

Address Type	Value
-----	-----
IPv4 Address	1
IPv6 Address	2

- o Flags (1 octet) -- The flags field is coded as follows:

```

      0               1
    0 1 2 3 4 5 6 7
+---+---+---+---+---+
|   Unused   |H|
+---+---+---+---+---+

```

H-bit - The Hub bit. When set to 1, the Client is Hub, otherwise it is spoke.

- o Length (2 octet) -- Length in octets of the current mandatory part.
- o Client Private Address (variable length) -- The ADC's logical private IP address. The value for this field is specified by the IP version field. If the message is sent from ADC to ADS, this address is private IP address of ADC. If the message is sent from ADS to ADC, this address is private IP address of destination ADC.

5.2.2. Session Header

The Session Header, denoted Sesshdr in this document, is contained in the messages that are sent and received between ADCs.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Role ID  | Address Type |           Length           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Source IP Address                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Destination IP Address                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 4. Session Header Format

- o Role ID (1 octet) -- Identifies the forwarding role of the ADC.
 - 1 - Hub.
 - 2 - Spoke.
- o Address Type (2 octet) -- Identifies the type of Source and

Destination IP Address type.

- o Length (2 octet) -- Length in octets of the current mandatory part.
- o Source IP Address (variable length) -- Identifies the sender's logical private IP address. The address type is specified by the Address Type field.
- o Destination IP Address (variable length) -- Identifies the receiver's logical private IP address. The address type is specified by the Address Type field.

5.3. Payload Part

Each payload defined in the section 5.3.1 through 5.3.6 has the general payload TLV(Type, Length, Value) format.

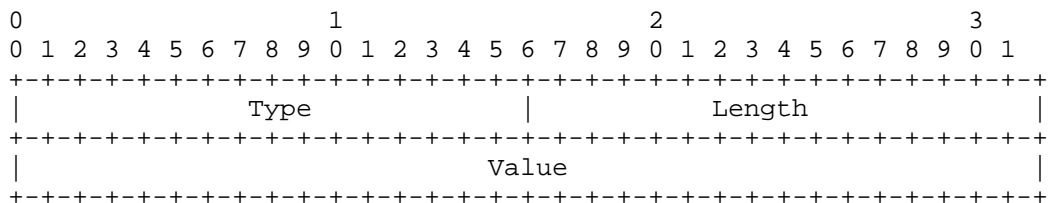


Figure 5. Payload TLV Format

The TLV fields are defined as follows:

- o Type (2 octet) -- The type of this payload.

Payload Type	Value
Client Information payload (CI)	1
Destination Client Payload (DC)	2
Network Information Payload (NI)	3
Traffic Flow Payload (TF)	4
Keepalive Parameter Payload (KP)	5
Redirect Payload (Red)	6

- o Length (2 octet) -- Length in octets of the current payload, including the type and length.
- o Value (variable octet) -- The value of this payload. Each payload has different content.

5.3.1. Client Information Payload

The Client Information Payload, denoted CI in this document, is used to be as part of Registration message to notify the ADS of ADC's information, or as part of Resolution Response message to notify the ADC of the remote ADC's information.

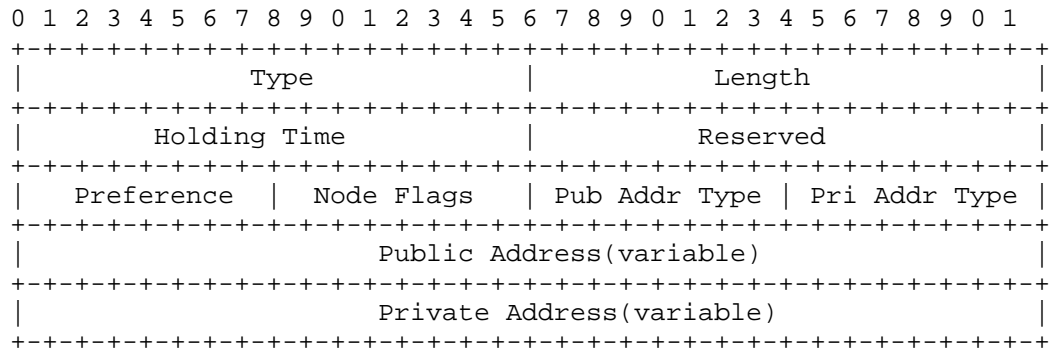
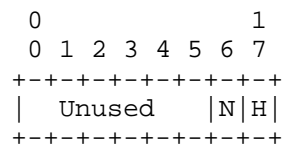


Figure 6. Client Information Payload Format

- o Holding Time (2 octet) -- Identifies the expire time(seconds) of the ADC's information obtained from ADS. The ADS SHOULD ignore this field when receiving the Registration Request message.
- o Reserved (2 octet) -- MUST be sent as zero; MUST be ignored on receipt.
- o Preference (1 octet) -- Identifies the ADC's preference. Higher values in the range 1 to 255 indicates higher preference. A zero value indicates no preference.
- o Node Flags (1 octet) -- The flags field is coded as follows:



H-bit - The hub bit. When set to 1, this current ADC is a hub, otherwise a spoke.

N-bit - The NAT bit. When set to 1, it indicates the ADC is behind the NAT.

- o Pub Addr Type (1 octet) -- Identifies the ADC's Public Address type.

1 -- IPv4

- 2 -- IPv6
- o Pri Addr Type (1 octet) -- Identifies the ADC's Private Address type.
 - 1 -- IPv4
 - 2 -- IPv6
- o Public Address (variable length) -- Identifies the ADC's IPsec peer address. The type of address for this field is specified by the Pub Addr Type field.
- o Private Address (variable length) -- Identifies the ADC's logical private IP address. The values for this field is specified by the Pri Addr Type field.

5.3.2. Destination Client Payload

The Destination Client Payload, denoted DC in this document, is used to be search key to discover the remote ADC's information.

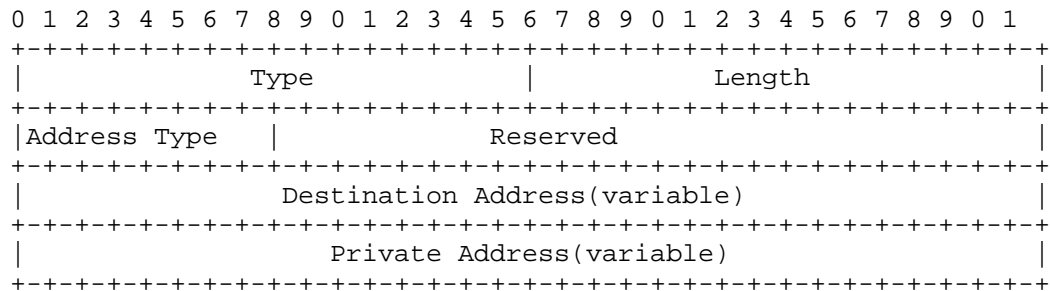


Figure 7. Destination Client Payload Format

- o Address Type (1 octet) -- Identifies the type of destination address and next hop address.
 - 1 -- IPv4
 - 2 -- IPv6
- o Reserved (3 octet) -- MUST be zero.
- o Destination IP Address (variable length) -- Identifies the destination IP address of communication. The value for this field is specified by the Address Type field.

- o Private Address (variable length) -- Identifies the next hop address to the destination network. The value for this field is specified by the Address Type field.

5.3.3. Network Information Payload

The Network Information Payload, denoted NI in this document, is used to carry private network information.

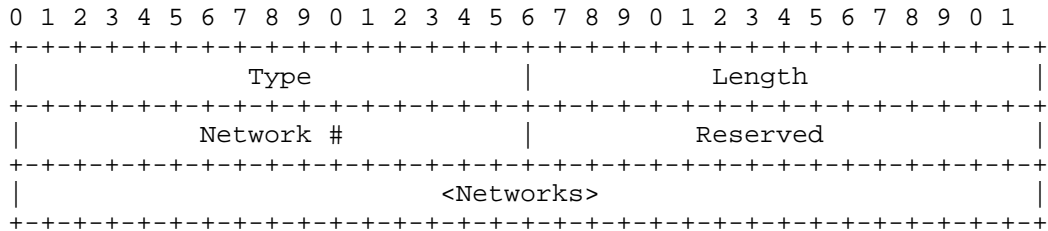


Figure 8. Network Information Payload Format

- o Network # (2 octet) -- Identifies the number of network this message contains.
- o Reserved (2 octet) -- MUST be zero.

Each network has the following formats:

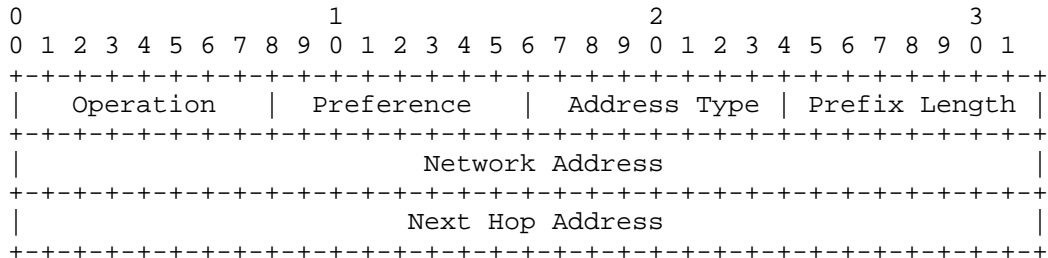


Figure 9. Network Format

- ```

o Operation (1 octet) -- Indicates the operation for the current
 network.

 0 - Reserved.

 1 - Add.

 2 - Delete.

```

3 - Update.

- o Preference (1 octet) -- Indicates the priority of network.
- o Address Type (1 octet) -- Indicates the Address type of network address and next hop address.

1 - IPv4 Address.

2 - IPv6 Address.

- o Prefix Length (1 octet) -- Is the octet length of the routing prefix.
- o Network Address (variable length) -- Identifies the address of the ADC's private network. The value for this field is specified by the Address Type field.
- o Next Hop Address(variable length) -- Identifies the next hop to the Network Address in the routing path. The Next Hop is also the ADC's private IP address. The value for this field is specified by the Address Type field.

#### 5.3.4. Traffic Flow Payload

The Traffic Flow Payload is denoted TF in this document. This payload contains the data flow information.

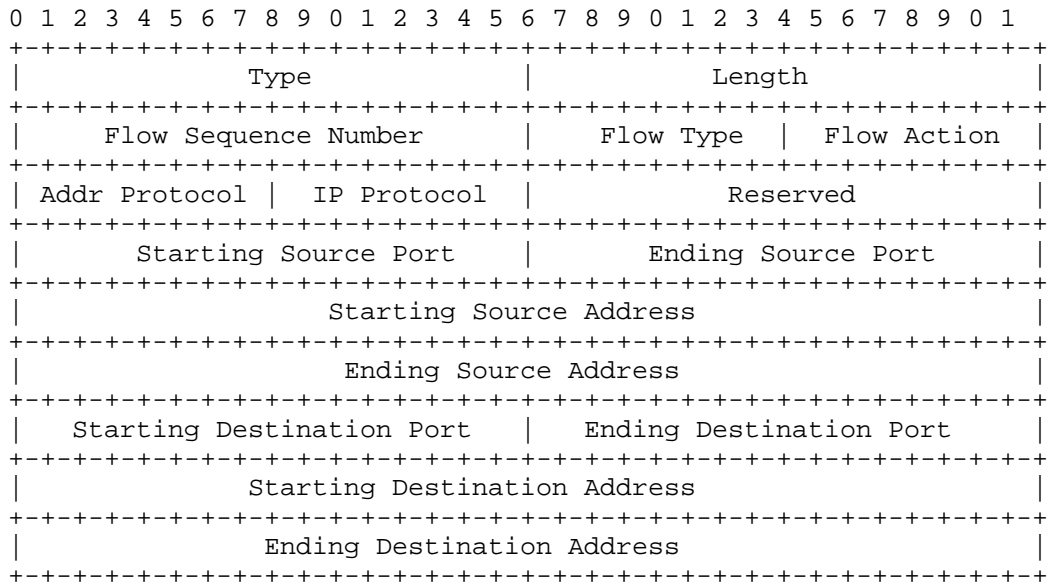


Figure 10. Shortcut Flow Payload Format

- o Flow Sequence Number (2 octet) -- Identifies the preference of the data flow. Lower values (in the range 0 to 65534) indicates higher preference.
- o Flow Type (1 octet) -- Identifies the traffic flow type.
  - 1 - Shortcut Data Flow.
- o Flow Action (1 octet) -- Identifies which action will to do when matching this flow.
  - 1 - Permit. The packet which matching the flow will trigger the specified function, such as redirection.
  - 2 - Deny. The packet which matching the flow will not trigger any function.
  - 3 - Discard. The packet which matching the flow will be discarded.
- o Addr Type (1 octet) -- Identifies which the data flow address type is, IPv4 or IPv6.
  - 1 - IPv4 Address.
  - 2 - IPv6 Address.

- o IP Protocol (1 octet) -- Identifies IP protocol of this data flow, such as UDP, TCP or ICMP etc.
- o Reserved (2 octet) -- MUST be zero.
- o Starting Source Port (2 octet) -- Value specifying the smallest source port number allowed.
- o Ending Source Port (2 octet) -- Value specifying the largest source port number allowed.
- o Starting Source Address (2 octet) -- The smallest source address.
- o Ending Source Address (2 octet) -- The largest source address.
- o Starting Destination Port (2 octet) -- Value specifying the smallest destination port number allowed.
- o Ending Destination Port (2 octet) -- Value specifying the largest destination port number allowed.
- o Starting Destination Address (2 octet) -- The smallest destination address.
- o Ending Destination Address (2 octet) -- The largest destination address.

### 5.3.5. Keepalive Parameter Payload

The Keepalive Parameter Payload is denoted KP in this document. This payload contains the parameter to sending keepalive message.

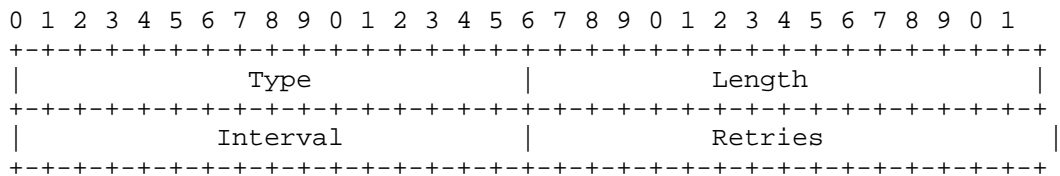


Figure 11. Keepalive Parameter Payload Format

- o Interval (2 octet) -- Identifies the timeout(seconds) of sending a Keepalive Request again.
- o Retries (2 octet) -- Identifies the number of retry attempts.

### 5.3.6. Redirect Payload

The Redirect Payload is denoted Red in this document. This payload contains the original packet information. The original packet is used for the spoke to send a Resolution Request packet to the ADS to get the peer's ADVPN information.

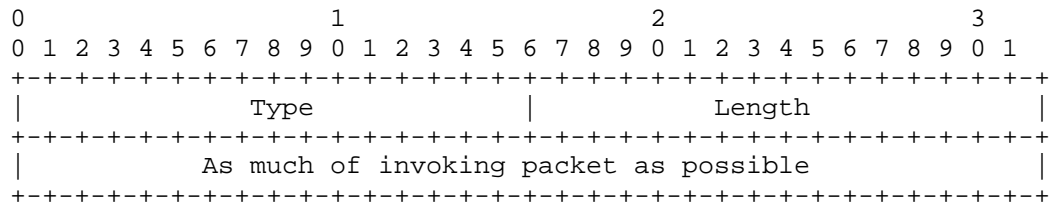


Figure 12. Redirect Payload Format



## 6. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

This draft is based on H3C/HP DVPN(Dynamic Virtual Private Network) solution. DVPN has been widely used since 2005. DVPN solution is not a single protocol, but an architecture. DVPN helps enterprises simplify the configuration and management of IPsec VPN tunnels. DVPN policies share security access, management, and quality of service (QoS) policies to easily connect thousands of remote branch and regional offices to the corporate headquarters or data centers. Administrators no longer need to login to each VPN device to manually set up site-to-site VPN tunnels at each branch or regional office, corporate headquarters, and data centers. DVPN is an innovation to simplify secure WAN connectivity for the enterprise.

DVPN is a complete and cost-effective solution that is ideal for the hub-and-spoke topology, the most common topology for enterprises, where you also have an option for mesh connectivity. DVPN spans across various domains such as routing, security, and address management.

The H3C/HP DVPN website link is:  
<http://h17007.www1.hp.com/us/en/networking/solutions/technology/dvpn/index.aspx>.

Although this ADVPN protocol comes from DVPN solution, it has been simplified and optimized to meet the requirements. The main differences is:

- o VAM(VPN Address Management) protocol and DVPN protocol is merged

into a single ADVPN protocol.

- o Security mechanism in VAM protocol is deleted, the ADVPN protocol should be protected by IPsec.
- o In DVPN, all packets and frames must be encapsulated in GRE first, and then be protected by IPsec. However, the ADVPN supports a pure IPsec encapsulation.

## 7. Security Considerations

The ADVPN protocol has no protocol-internal security mechanism, it relies on other security protocol to protect the ADVPN messages.

The messages between the ADC and ADS can be protected by the IPsec or SSL/TLS. The messages between ADCs is protected by IPsec.

It is highly recommended that the wildcard pre-shared-key in IKEv1 or IKEv2 is not used in the ADVPN, the attacker can access the ADVPN network if one ADVPN peer is compromised.

## 8. IANA Considerations

IANA may need to allocate additional values for the options presented in this document. The values of the protocol field needed to be assigned from the numbering space.

## 9. Acknowledgments

## 10. References

### 10.1. Normative References

- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC1776] Crocker, S., "The Address is the Message", RFC 1776, April 1 1995.
- [TRUTHS] Callon, R., "The Twelve Networking Truths", RFC 1925, April 1 1996.
- [ADVPN\_Problem]  
Hanna, S., "Auto Discovery VPN Problem Statement and Requirements", draft-ietf-ipsecme-p2p-vpn-problem-07.txt, June 2013.
- [IPSECARCH]  
Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

## 10.2. Informative References

- [NHRP] J. Luciani, "NBMA Next Hop Resolution Protocol (NHRP)", RFC2332, April 1998.
- [RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, April 1 2009.
- [RFC5514] Vyncke, E., "IPv6 over Social Networks", RFC 5514, April 1 2009.

## Appendix A. Comparison Against ADVPN Requirements

## Requirement #1:

In this ADVPN protocol, each ADC only needs to configure its own SPD and PAD. Adding or removing an ADC in the ADVPN topology does not need configuration change for any other ADC.

## Requirement #2:

Each ADC registers its public address(peer address) and private address to ADS, and the other ADC query the ADS to obtain the public address. Therefore, even the public address of one ADC is updated every time the device comes up, the other ADC can communicate with it without any configuration change.

## Requirement #3:

The tunneling protocol(e.g. GRE) and routing protocol(e.g. OSPF, BGP) can run over the spoke-to-hub and spoke-to-spoke IPsec tunnel with minimal configuration. These protocols do not need to aware of IPsec tunnel, and also IPsec does not need to be aware of these protocol packets.

## Requirement #4:

The ADS is the controller of the ADVPN network. It has control policy which decides whether the spoke can be allowed to have a direct communication with other spoke. The control policy is pushed to hub from the ADS after the hub ADC finishes registration. The detailed description about it is given in Section 4.4.

## Requirement #5:

To mitigate the affect of compromised ADVPN peer, each ADVPN peer SHOULD have unique and long term authentication credential such as

certificate used for IKEv1 and IKEv2 negotiation. The wildcard pre-shared-key SHOULD NOT be used for ADVPN connection.

Requirement #6:

When the endpoint roams, if its public address changes, it will be as a new ADC to rejoin ADVPN network. After re-registering to ADS, it connects the hub gateway again. The data traffic can be transferred through new spoke-to-hub IPsec tunnel and spoke-to-spoke IPsec tunnel.

Requirement #7:

The information of hub ADCs is maintained by ADS. if the endpoints roams across the hub gateway, the new hub ADC' information will be pushed to the spoke ADCs, and new IPsec tunnel is established between the ADC and new hub ADC. The traffic is migrated from one gateway to another gateway.

Requirement #8:

All the ADVPN peers can be located behind NAT boxes. After ADCs registration, the ADS detects which ADC is behind NAT box. The ADS updates the public IP address/Port information of spoke ADC with the modifying IP address/Port by NAT box from hub ADC after the IPsec tunnel is established between the spoke and hub. Therefore, even two spokes are both behind NAT boxes, they can also establish the direct connectivity.

Requirement #9:

All the tables in the ADVPN protocol are reportable and manageable, such as IP Address Mapping Database, Private Network Information Database, Session Table and Shortcut Flow Table etc. Change of IPsec SA such as establishment and expiration can be logged and reported as necessary events. This document does not create a MIB.

Requirement #10:

To support allied and federated environments, the ADVPN peer SHOULD use the certificate as the credential for connections. With the PKI trust architecture, the ADVPN peer from different organizations can be able to connect to each other.

Requirement #11:

The ADS is the controller of ADVPN network. The administrator can configure the control policy on ADS to determine the ADVPN network is

a Star, Full mesh or a partial full mesh topology.

Requirement #12:

The ADVPN protocol can cooperated with IGMP and PIM to provide multicast function. PIM packets run over spoke-to-hub IPsec tunnel to establish the PIM neighbor. Multicast traffic is selective replicated to spokes on the hub.

Requirement #13:

In the ADVPN protocol, all the changes such as IPsec SA establishment, shortcut route injection etc. can be monitored, logged and reported to help trouble shooting.

Requirement #14:

When L3VPN runs over IPsec tunnel, GRE or other tunnel protocol can be transport-link protocol. These tunneling protocols can run over the spoke-to-hub and spoke-to-spoke IPsec tunnel in ADVPN network.

Requirement #15:

The ADC can register its QoS policy information to ADS, and when the other ADC query the ADC's information, it can obtain the related QoS policy information.

Requirement #16:

In the ADVPN protocol, the administrator can specify more than one hub in the ADS for the ADC. The ADC gets all the hub ADC information after registration and establishes IPsec tunnel with each hub ADC.

Authors' Addresses

Yu Mao(Toby Mao)  
Hangzhou H3C Tech. Co., Ltd.  
Oriental Electronic Bld., No. 2  
Chuangye Road  
Shang-Di Information Industry  
Hai-Dian District  
Beijing 100085  
China

EMail: yumao9@gmail.com

ZhanQun Wang  
Hangzhou H3C Tech. Co., Ltd.

Oriental Electronic Bld., No. 2  
Chuangye Road  
Shang-Di Information Industry  
Hai-Dian District  
Beijing 100085  
China

EMail: wangzhanqun.ietf@gmail.com

Vishwas Manral  
Hewlett-Packard Co.  
19111 Pruneridge Ave.  
Cupertino, CA 95113  
USA

Email: vishwas.manral@hp.com

IPsecME Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: October 15, 2014

Y. Nir  
Check Point  
April 13, 2014

Handing Over Child SAs Following Re-Authentication in IKEv2  
draft-nir-ipsecme-cafr-04

Abstract

This document describes an extension to the IKEv2 protocol whereby Child SAs are moved to the new IKE SA following re-authentication. This allows for a smoother transition with no loss of connectivity.

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 October 15, 2014.

Copyright Notice

Copyright (c) 2014 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. Conventions Used in This Document . . . . .              | 3 |
| 2. Handing Over Child SAs . . . . .                           | 4 |
| 2.1. The HAND_OVER_CHILD_SAS Notification . . . . .           | 4 |
| 2.2. Verifying the HAND_OVER_CHILD_SAS Notification . . . . . | 5 |
| 3. The Illustrated Protocol . . . . .                         | 5 |
| 4. Interaction with Other Standards . . . . .                 | 6 |
| 5. Acknowledgements . . . . .                                 | 6 |
| 6. IANA Considerations . . . . .                              | 6 |
| 7. Security Considerations . . . . .                          | 6 |
| 8. Changes from Previous Versions . . . . .                   | 7 |
| 9. References . . . . .                                       | 7 |
| 9.1. Normative References . . . . .                           | 7 |
| 9.2. Informative References . . . . .                         | 7 |
| Author's Address . . . . .                                    | 7 |



## 1. Introduction

The Internet Key Exchange version 2 (IKEv2) protocol, as specified in [RFC5996bis] associates Child SAs with the IKE SAs under which the exchange that created them took place. With the deletion of the IKE SA due to expiry, policy change, or an explicit message from the peer, the child SAs associated with it are implicitly closed as described in section 1.4.1 of the IKEv2 document. This behavior is not desired when IKE SAs are replaced rather than deleted, because those child SAs could still be valid and there is no security reason to create new ones prematurely.

There are two cases where an IKE SA is replaced.

1. Rekeying, where new keys are generated. This is described in section 2.18 of RFC 5996. This is done mainly for key freshness.
2. Re-Authentication, where both sides authenticate, and new keys are generated. This is done as part of a risk management policy, to limit the time that compromised IKE SA keys can be used to provide the attacker access to the network. No reauthentication exchange is specified in the RFC. Instead, it's simply the Initial and Authentication exchanges done as if from scratch. This is described in section 2.8.3 of RFC 5996.

For rekeying, RFC 5996 provides a way to avoid having to re-create all child SAs. When an IKE SA is rekeyed, all the Child SAs under the old IKE SA are inherited by the new IKE SA, so that the subsequent deletion of the old IKE SA does not affect the Child SAs. This behavior is described in section 2.8 paragraph 4 of RFC 5996.

For reauthentication, RFC 5996 does not provide a similar mechanism, and section 2.8.3 explicitly says that Child SAs need to be created from scratch. This is often inconvenient, as IPsec systems usually create Child SAs only in response to traffic and multiple Child SAs may exist for a single IKE SA. The protocol extension in this draft closes this gap.

### 1.1. Conventions Used in This Document

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].

The terms IKE SA, Child SA, Rekeying, and Reauthentication are as described in the RFC 5996.

## 2. Handing Over Child SAs

This document defines a new notification that can be sent over an old IKE SA, just after an IKE\_AUTH exchange has been used to re-authenticate. The notification tells the peer to transfer all Child SAs that belong to the current (old) IKE SA to be owned by the new IKE SA, so that when the old IKE SA is deleted, those Child SAs are not. If both peers send this notification, all Child SAs belonging to the old IKE SA are immediately inherited by the new IKE SA.

In addition to the Child SAs, any IP address assigned to either peer through the use of the CFG payload (as described in section 2.19 of RFC 5996), is also associated with the new IKE SA.

The new notification MAY be accompanied by a DELETE payload, so as to transfer the Child SAs and delete the old IKE SA at the same time. These payloads don't have to be in the same exchange, and it is perfectly valid for the initiator to send the HAND\_OVER\_CHILD\_SA notification in one exchange, and only then send the DELETE payload in a different exchange. A responder, however, MUST support receiving both payloads in the same exchange, and MUST transfer the child SAs and assigned IP address before acting on the DELETE payload.

### 2.1. The HAND\_OVER\_CHILD\_SAS Notification

The HAND\_OVER\_CHILD\_SA notification is formatted as follows:

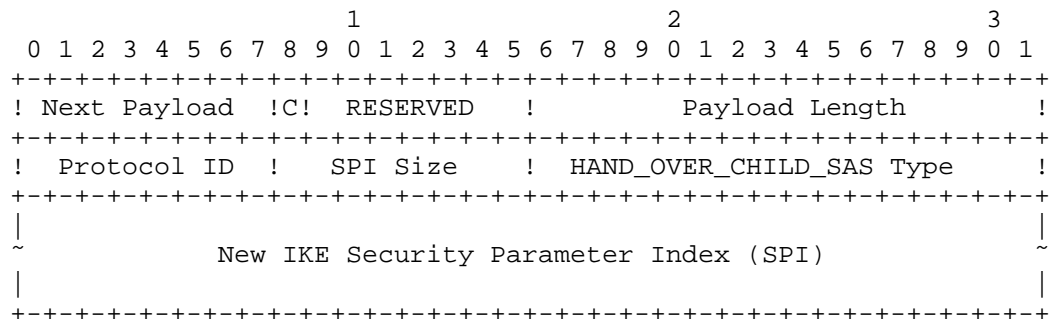


Figure 1

- o Protocol ID (1 octet) MUST be zero, as specified in Section 3.10 of RFC 5996.
- o SPI Size (1 octet) MUST be zero, in conformance with Section 3.10 of RFC 5996.

- o HAND\_OVER\_CHILD\_SAS Notify Message Type (2 octets) - MUST be xxxxxx, the value assigned for HAND\_OVER\_CHILD\_SAS. TBA by IANA.
- o Notification Data, or New IKE Security Parameter Index (16 or zero octets) - In the request, this field contains the concatenated SPIs of the new IKE SA. The Initiator SPI comes first, similar to the first 16 bytes of the IKE header. Note that this is not the SPI field of the notification payload, but the data field. In the response, this field is omitted (zero-length).

## 2.2. Verifying the HAND\_OVER\_CHILD\_SAS Notification

To go through with the new IKE SA inheriting the SAs of the old IKE SA, all of the following MUST apply:

- o Both sides have to be successfully authenticated, and the new IKE SA has to be established.
- o The authenticated identities of both sides under the new IKE SA are the same as those under the old IKE SA. If the authenticated identity of one peer differs from the authenticated identity that it had in the previous IKE SA, the Responder MUST NOT return the HAND\_OVER\_CHILD\_SAS notification. Such an error indicates either an attack or a bug in the peer, so this should be logged and reported.
- o The New IKE SPIs in the notifications from both peers MUST match bit for bit.

If the new IKE SA is not fully authenticated, or if the peer authenticated identity in the new IKE SA is not the same as in the current IKE SA, a conformant Responder MUST NOT send the HAND\_OVER\_CHILD\_SAS Notification, and MUST not move the Child SAs.

If the Initiator has not sent the HAND\_OVER\_CHILD\_SAS notification, but has received it in a response, it MUST ignore it and MUST NOT move the Child SAs.

If the Initiator has sent the notification, but the Responder has not sent it, then the Initiator MUST NOT move the Child SAs.

If the Initiator has sent the notification, but the notification from the Responder contains IKE SPIs (whether correct or not), then the Initiator MUST send a SYNTAX\_ERROR notification and MUST NOT transfer the Child SAs.

## 3. The Illustrated Protocol

The Informational exchange after creating a new IKE SA:

| Initiator                                                                 | Responder                                                      |
|---------------------------------------------------------------------------|----------------------------------------------------------------|
| -----                                                                     |                                                                |
| HDR, SK {<br>N(HAND_OVER_IKE_SAS, new IKE SA SPIs),<br>DELETE<br>}<br>--> | HDR, SK {<br>N(HAND_OVER_IKE_SAS, new IKE SA SPIs)<br><--<br>} |

Figure 2

Note that in the above figure, the HDR has the IKE SPIs of the old IKE SAs, and the SK payload uses the keys of the old IKE SA, because this message is sent over the old IKE SA.

#### 4. Interaction with Other Standards

This document changes things so that there is often no need to create new Child SAs along with the new IKE SA when reauthenticating. This makes the full IKE\_AUTH exchange with the piggy-backed Child SA exchange (as described in RFC 5996) superfluous. Implementations should consider implementing the childless extension of IKEv2 ([RFC6023]) in addition to this specification.

#### 5. Acknowledgements

The author would like to thank Valery Smyslov for the suggestion of moving the hand-over from the IKE\_AUTH to an Informational under the old IKE SA and other suggestions. This changed (in version -01) simplified the protocol significantly. Tero Kivinen provided valuable input about the security considerations and error handling.

#### 6. IANA Considerations

IANA is requested to assign a notify message type from the status types range (16418-40959) of the "IKEv2 Notify Message Types" registry with name "HAND\_OVER\_CHILD\_SAS"

#### 7. Security Considerations

The HAND\_OVER\_CHILD\_SAS notification is sent protected by the old IKE SA. This protects against stealing child SAs. The requirement for

sameness of authenticated identity protects against errors by one peer transferring child SAs to some other peer. It also protects against an attempt by one endpoint to transfer ownership of SAs to another endpoint, so as to assume the authorization assigned by the peer to the other endpoint.

## 8. Changes from Previous Versions

[NOTE TO RFC EDITOR: PLEASE REMOVE THIS SECTION]

Version -01 moved the sending of the notification from the IKE\_AUTH exchange that is part of reauthentication to the Informational exchange that is part of closing the old IKE SA. This made cryptographic binding to the old IKE SA unnecessary.

Version -02 changed the notification payload so that the IKE SPI of the other IKE SA is now in the data field of the notification payload, rather than the SPI field. This makes it more in line with how the notification payload is defined in RFC 5996.

Version -03 tightened the security considerations, the format of the notification in the response, and error handling.

## 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.

[RFC5996bis]  
Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", draft-kivinen-ipsecme-ikev2-rfc5996bis-00 (work in progress), August 2013.

### 9.2. Informative References

[RFC6023] Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, October 2010.

Author's Address

Yoav Nir  
Check Point Software Technologies Ltd.  
5 Hasolelim st.  
Tel Aviv 6789735  
Israel

Email: [ynir.ietf@gmail.com](mailto:ynir.ietf@gmail.com)

