

Network Working Group
Internet-Draft
Obsoletes: 6407 (if approved)
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: October 8, 2022

V. Smyslov
ELVIS-PLUS
B. Weis
Independent
April 6, 2022

Group Key Management using IKEv2
draft-ietf-ipsecme-g-ikev2-06

Abstract

This document presents an extension to the Internet Key Exchange version 2 (IKEv2) protocol for the purpose of a group key management. The protocol is in conformance with the Multicast Security (MSEC) key management architecture, which contains two components: member registration and group rekeying. Both components require a Group Controller/Key Server to download IPsec group security associations to authorized members of a group. The group members then exchange IP multicast or other group traffic as IPsec packets. This document obsoletes RFC 6407. This documents also updates RFC 7296 by renaming one of transform types defined there.

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 <https://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 8, 2022.

Copyright Notice

Copyright (c) 2022 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

(<https://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 and Overview	3
1.1. Requirements Notation	5
1.2. Terminology	5
2. G-IKEv2 Protocol	7
2.1. G-IKEv2 Integration into IKEv2 Protocol	7
2.1.1. G-IKEv2 Transport and Port	7
2.2. G-IKEv2 Payloads	8
2.3. G-IKEv2 Member Registration and Secure Channel Establishment	9
2.3.1. GSA_AUTH exchange	9
2.3.2. GSA_REGISTRATION Exchange	11
2.3.3. GM Registration Operations	12
2.3.4. GCKS Registration Operations	14
2.4. Group Maintenance Channel	15
2.4.1. GSA_REKEY	16
2.4.2. GSA_INBAND_REKEY Exchange	22
2.4.3. Deletion of SAs	22
2.5. Counter-based modes of operation	23
2.5.1. Allocation of SIDs	24
2.5.2. GM Usage of SIDs	25
2.6. Replay Protection for Multicast Data-Security SAs	25
3. Group Key Management and Access Control	26
3.1. Key Wrap Keys	26
3.1.1. Default Key Wrap Key	27
3.2. GCKS Key Management Semantics	27
3.2.1. Forward Access Control Requirements	28
3.3. GM Key Management Semantics	28
3.4. SA Keys	30
4. Header and Payload Formats	31
4.1. G-IKEv2 Header	31
4.2. Group Identification Payload	31
4.3. Security Association - GM Supported Transforms Payload	31
4.4. Group Security Association Payload	32
4.4.1. Group Policies	32
4.4.2. Group Security Association Policy Substructure	33
4.4.3. Group Associated Policy Substructure	40
4.5. Key Download Payload	42
4.5.1. Wrapped Key Format	42

4.5.2. Group Key Packet Substructure	44
4.5.3. Member Key Packet Substructure	46
4.6. Delete Payload	48
4.7. Notify Payload	48
4.7.1. USE_TRANSPORT_MODE Notification	49
4.8. Authentication Payload	50
5. Using G-IKEv2 Attributes	50
6. Interaction with other IKEv2 Protocol Extensions	52
6.1. Mixing Preshared Keys in IKEv2 for Post-quantum Security	53
7. Security Considerations	55
7.1. GSA Registration and Secure Channel	55
7.2. GSA Maintenance Channel	55
7.2.1. Authentication/Authorization	55
7.2.2. Confidentiality	55
7.2.3. Man-in-the-Middle Attack Protection	55
7.2.4. Replay/Reflection Attack Protection	55
8. IANA Considerations	56
8.1. New Registries	56
8.2. Changes in the Existing IKEv2 Registries	57
9. Acknowledgements	59
10. Contributors	60
11. References	60
11.1. Normative References	60
11.2. Informative References	61
Appendix A. Use of LKH in G-IKEv2	65
A.1. Notation	65
A.2. Group Creation	65
A.3. Simple Group SA Rekey	66
A.4. Group Member Exclusion	66
Authors' Addresses	68

1. Introduction and Overview

A group key management protocol provides IPsec keys and policy to a set of IPsec devices which are authorized to communicate using a Group Security Association (GSA) defined in [RFC3740]. The data communications within the group (e.g., IP multicast packets) are protected by a key pushed to the group members (GMs) by the Group Controller/Key Server (GCKS). This document presents an extension to IKEv2 [RFC7296] called G-IKEv2, that allows to perform a group key management.

G-IKEv2 conforms to the Multicast Group Security Architecture [RFC3740], Multicast Extensions to the Security Architecture for the Internet Protocol [RFC5374] and the Multicast Security (MSEC) Group Key Management Architecture [RFC4046]. G-IKEv2 replaces GDOI [RFC6407], which defines a similar group key management protocol using IKEv1 [RFC2409] (since deprecated by IKEv2). When G-IKEv2 is

used, group key management use cases can benefit from the simplicity, increased robustness and cryptographic improvements of IKEv2 (see Appendix A of [RFC7296]).

A GM begins a "registration" exchange when it first joins the group. With G-IKEv2, the GCKS authenticates and authorizes GMs, then pushes policy and keys used by the group to the GM. G-IKEv2 includes two "registration" exchanges. The first is the GSA_AUTH exchange (Section 2.3.1), which is used when a GM first conatcts a GCKS. The second is the GSA_REGISTRATION exchange (Section 2.3.2), which a GM can use within an established IKE SA. Group rekeys are accomplished using either the GSA_REKEY pseudo-exchange (a single message distributed to all GMs, usually as a multicast message), or as a GSA_INBAND_REKEY exchange delivered individually to group members using existing IKE SAs).

Large and small groups may use different sets of these protocols. When a large group of devices are communicating, the GCKS is likely to use the GSA_REKEY message for efficiency. This is shown in Figure 1. (Note: For clarity, IKE_SA_INIT is omitted from the figure.)

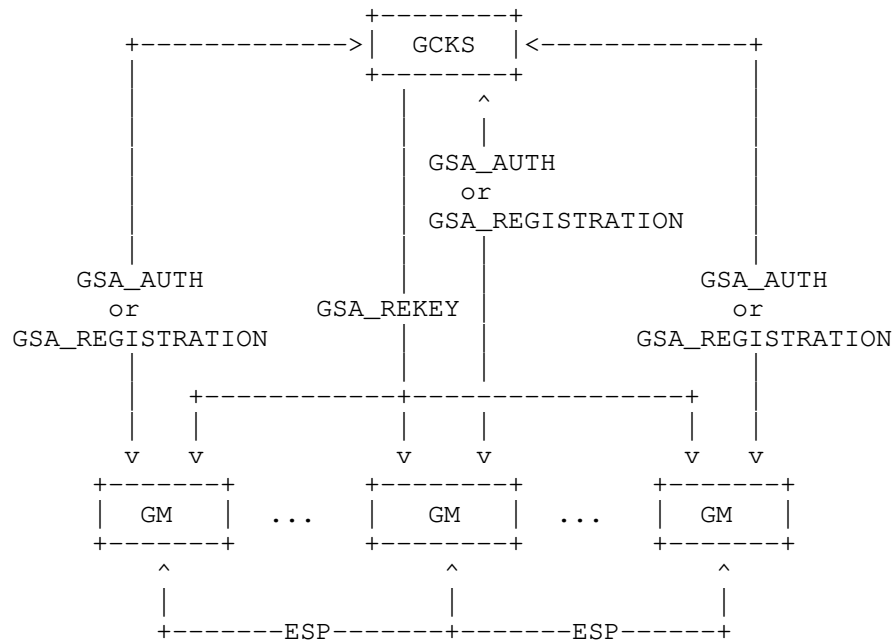


Figure 1: G-IKEv2 used in large groups

Alternatively, a small group may simply use the GSA_AUTH as a registration protocol, where the GCKS issues rekeys using the GSA_INBAND_REKEY within the same IKE SA. The GCKS is also likely to be a GM in a small group (as shown in Figure 2.)

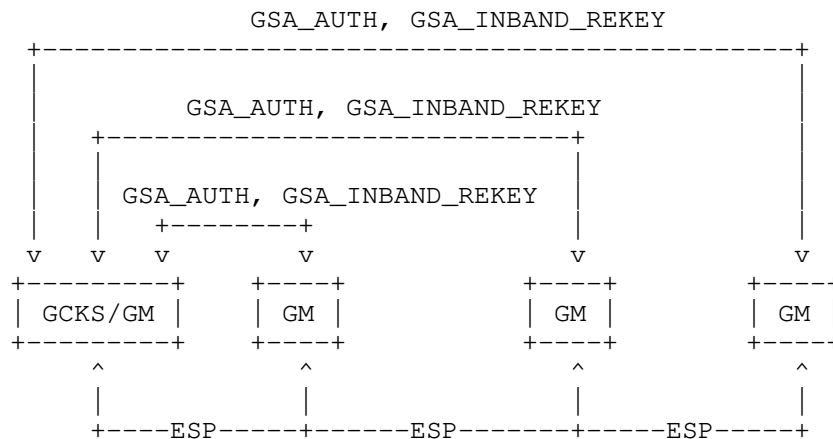


Figure 2: G-IKEv2 used in small groups

A combination of these approaches is also possible. For example, the GCKS may use more robust GSA_INBAND_REKEY to provide keys for some GMs (for example, those acting as senders in the group) and GSA_REKEY for the rest.

IKEv2 message semantics are preserved in that all communications consists of message request-response pairs. The exception to this rule is the GSA_REKEY pseudo-exchange, which is a single message delivering group updates to the GMs.

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

It is assumed that readers are familiar with the IPsec architecture [RFC4301], its extension for multicast [RFC5374]. This document defines an extension to the IKEv2 protocol [RFC7296], so it is assumed that readers have good understanding of this protocol.

The following key terms are used throughout this document (mostly borrowed from [RFC5374] and [RFC6407]).

Group

A set of devices that work together to protect group communications.

Group Member (GM)

An IPsec device that belongs to a group. A Group Member is authorized to be a Group Sender and/or a Group Receiver.

Group Receiver

A Group Member that is authorized to receive packets sent to a group by a Group Sender.

Group Sender

A Group Member that is authorized to send packets to a group.

Group Key Management (GKM) Protocol

A key management protocol used by a GCKS to distribute IPsec Security Association policy and keying material. A GKM protocol is used when a group of IPsec devices require the same SAs. For example, when an IPsec SA describes an IP multicast destination, the sender and all receivers need to have the group SA.

Group Controller Key Server (GCKS)

A Group Key Management (GKM) protocol server that manages IPsec state for a group. A GCKS authenticates and provides the IPsec SA policy and keying material to GKM Group Members.

Data-Security SA

The security policy distributed by a GDOI GCKS describing traffic that is expected to be protected by group members. This document described the distribution of IPsec AH and ESP Data-Security SAs.

Rekey SA

The security policy protecting Group Key Management Protocol.

Group Security Association (GSA)

A collection of Data-Security Associations (SAs) and Rekey SAs necessary for a Group Member to receive key updates. A GSA describes the working policy for a group. Refer to [RFC4046] for additional information.

Traffic Encryption Key (TEK)

The symmetric cipher key used in a Data-Security SA (e.g., IPsec ESP) to protect traffic.

Key Encryption Key (KEK)

The symmetric cipher key used in a Rekey SA to protect distribution of new keys.

Key Wrap Key (KWK)

The symmetric cipher key used to protect another key.

Group Associated Policy (GAP)

Group-wide policy not related to a particular SA.

Sender-ID (SID)

A unique identifier of a Group Sender in the context of an active GSA, used to form Initialization Vector (IV) in counter-based cipher modes.

Logical Key Hierarchy (LKH)

A group management method defined in Section 5.4 of [RFC2627].

2. G-IKEv2 Protocol

2.1. G-IKEv2 Integration into IKEv2 Protocol

G-IKEv2 is an extension to IKEv2 and uses its security mechanisms (peer authentication, confidentiality, message integrity) to ensure that only authenticated devices have access to the group policy and keys. G-IKEv2 further provides group authorization, and secure policy and key download from the GCKS to GMS.

G-IKEv2 is compatible with most IKEv2 extensions defined so far and it is believed that future IKEv2 extensions will also be possible to use with G-IKEv2. However some IKEv2 extensions require special handling if used with G-IKEv2. See Section 6 for more details.

It is assumed that readers are familiar with the IKEv2 protocol, so this document skips many details that are described in [RFC7296].

2.1.1. G-IKEv2 Transport and Port

G-IKEv2 SHOULD use UDP port 848, the same as GDOI [RFC6407], because they serve a similar function. They can use the same ports, just as IKEv1 and IKEv2 can share port 500. The version number in the IKE header distinguishes the G-IKEv2 protocol from GDOI protocol [RFC6407]. G-IKEv2 MAY also use the IKEv2 ports (500, 4500), which would provide a better integration with IKEv2. G-IKEv2 MAY also use

TCP transport for registration (unicast) IKE SA, as defined in [RFC8229].

Section 2.23 of [RFC7296] describes how IKEv2 deals with NATs. Despite the fact, that with G-IKEv2 the registration SA doesn't create any unicast IPsec SAs and thus there is no unicast ESP traffic between the GM and the GCKS to encapsulate in UDP if NAT is present, the actions described in this section concerned with the IKE SA MUST be honored. If the GM and the GCKS used UDP port 848 for the IKE_SA_INIT exchange, they MUST behave as if they used UDP port 500.

2.2. G-IKEv2 Payloads

In the following descriptions, the payloads contained in the G-IKEv2 messages are indicated by names as listed below.

Notation	Payload
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
D	Delete
GSA	Group Security Association
HDR	IKEv2 Header
IDg	Identification - Group
Idi	Identification - Initiator
Idr	Identification - Responder
KD	Key Download
KE	Key Exchange
Ni, Nr	Nonce
N	Notify
SA	Security Association
SAg	Security Association - GM Supported Transforms

Payloads defined as part of other IKEv2 extensions MAY also be included in these messages. Payloads that may optionally appear in G-IKEv2 messages will be shown in brackets, such as [CERTREQ].

G-IKEv2 defines several new payloads not used in IKEv2:

- o IDg (Group ID) -- The GM requests the GCKS for membership into the group by sending its IDg payload.
- o GSA (Group Security Association) -- The GCKS sends the group policy to the GM using this payload.
- o KD (Key Download) -- The GCKS sends the keys and the security parameters to the GMs using the KD payload.

- o SAg (Security Association -- GM Supported Transforms) -- the GM sends supported transforms, so that GCKS may select a policy appropriate for all members of the group.

The details of the contents of each payload are described in Section 4.

2.3. G-IKEv2 Member Registration and Secure Channel Establishment

The registration protocol consists of a minimum of two exchanges, IKE_SA_INIT and GSA_AUTH; member registration may have a few more messages exchanged if the EAP method, cookie challenge (for DoS protection), negotiation of Diffie-Hellman group or IKEv2 extensions based on [I-D.ietf-ipsecme-ikev2-intermediate] are used. Each exchange consists of request/response pairs. The first exchange IKE_SA_INIT is defined in IKEv2 [RFC7296]. This exchange negotiates cryptographic algorithms, exchanges nonces and computes a shared key between the GM and the GCKS.

The second exchange GSA_AUTH authenticates the previously exchanged messages, exchanges identities and certificates. The GSA_AUTH messages are encrypted and integrity protected with keys established through the previous exchanges, so the identities are hidden from eavesdroppers and all fields in all the messages are authenticated. The GCKS SHOULD authorize group members to be allowed into the group as part of the GSA_AUTH exchange. Once the GCKS accepts a group member to join a group it will download the data security keys (TEKs) and/or group key encrypting key (KEK) as part of the GSA_AUTH response message.

2.3.1. GSA_AUTH exchange

After the group member and GCKS negotiate cryptographic algorithms, exchange nonces, and compute shared keys as defined in IKEv2 [RFC7296], the GSA_AUTH exchange MUST complete before any other exchanges defined in this document can be done. GSA_AUTH is used to authenticate the previous exchanges, exchange identities and certificates. G-IKEv2 also uses this exchange for group member registration and authorization.

The GSA_AUTH exchange is identical to the IKE_AUTH exchange with the difference that its goal is to establish multicast Data-Security SAs and optionally provide GM with the keys for Rekey SA. The set of payloads in the GSA_AUTH exchange is slightly different, because policy is not negotiated between the group member and the GCKS, but instead downloaded from the GCKS to the group member. Note also, that GSA_AUTH has its own exchange type, which is different from the IKE_AUTH exchange type.

Nevertheless, the security properties of the GSA_AUTH exchange are the same as the properties of the IKE_AUTH exchange and most IKEv2 extensions to the IKE_AUTH exchange (like [RFC6467]) can also be used with the GSA_AUTH exchange.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK{IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, IDg, [SAg,] [N]}	-->

Figure 3: GSA_AUTH Request

A group member initiates a GSA_AUTH request to join a group indicated by the IDg payload. The GM MAY include an SAg payload declaring which Transforms it is willing to accept. A GM that intends to act as Group Sender SHOULD include a Notify payload status type of SENDER, which enables the GCKS to provide any additional policy necessary by group senders.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK{IDr, [CERT,] AUTH, [GSA, KD,] [N,]}

Figure 4: GSA_AUTH Normal Response

The GCKS responds with IDr, optional CERT, and AUTH payloads as if it were an IKE_AUTH. It also informs the group member of the cryptographic policies of the group in the GSA payload and the key material in the KD payload.

In addition to the IKEv2 error handling, the GCKS can reject the registration request when the IDg is invalid or authorization fails, etc. In these cases, see Section 4.7, the GSA_AUTH response will not include the GSA and KD, but will include a Notify payload indicating errors. If a GM included an SAg payload, and the GCKS chooses to evaluate it, and the GCKS detects that the group member cannot support the security policy defined for the group, then the GCKS SHOULD return a NO_PROPOSAL_CHOSEN. Other types of error notifications can be INVALID_GROUP_ID, AUTHORIZATION_FAILED or REGISTRATION_FAILED.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK{IDr, [CERT,] AUTH, N}

Figure 5: GSA_AUTH Error Response

If the group member finds the policy sent by the GCKS is unacceptable, the member SHOULD initiate GSA_REGISTRATION exchange sending IDg and the Notify NO_PROPOSAL_CHOSEN (see Section 2.3.2)).

2.3.2. GSA_REGISTRATION Exchange

When a secure channel is already established between a GM and the GCKS, the GM registration for a group can reuse the established secure channel. In this scenario the GM will use the GSA_REGISTRATION exchange. Payloads in the exchange are generated and processed as defined in Section 2.3.1.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK{IDg, [SAg,] [N]} -->	
	<-- HDR, SK{GSA,] [N,]}

Figure 6: GSA_REGISTRATION Normal Exchange

As with GSA_AUTH exchange, the GCKS can reject the registration request when the IDg is invalid or authorization fails, or GM cannot support the security policy defined for the group (which can be concluded by GCKS by evaluation of SAg payload). In this case the GCKS returns an appropriate error notification as described in Section 2.3.1.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK{IDg, [SAg,] [N]} -->	
	<-- HDR, SK{N}

Figure 7: GSA_REGISTRATION Error Exchange

This exchange can also be used if the group member finds the policy sent by the GCKS is unacceptable. The group member SHOULD notify the GCKS by sending IDg and the Notify type NO_PROPOSAL_CHOSEN, as shown below. The GCKS in this case MUST remove the GM from the group IDg.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK{IDg, N} -->	
	<-- HDR, SK{}

Figure 8: GM Reporting Errors in GSA_REGISTRATION Exchange

2.3.3. GM Registration Operations

A G-IKEv2 Initiator (GM) requesting registration contacts the GCKS using the IKE_SA_INIT exchange and receives the response from the GCKS. This exchange is unchanged from the IKE_SA_INIT in IKEv2 protocol. The IKE_SA_INIT exchange may optionally be followed by one or more the IKE_INTERMEDIATE exchanges if the GM and the GCKS negotiated using IKEv2 extensions based on this exchange.

Next the GM sends the GSA_AUTH request message with the IKEv2 payloads from IKE_AUTH (without the SAi2, TSi and TSr payloads) along with the Group ID informing the GCKS of the group the initiator wishes to join. An initiator intending to emit data traffic SHOULD send a SENDER Notify payload status. The SENDER notification not only signifies that it is a sender, but provides the initiator the ability to request Sender-ID (SID) values, in case the Data-Security SA supports a counter mode cipher. Section 2.5) includes guidance on requesting Sender-ID values.

A GM may be limited in the types of Transforms that it is able or willing to use, and may find it useful to inform the GCKS which Transforms it is willing to accept for different security protocols by including the SAg payload into the request message. Proposals for Rekey SA (with protocol GIKE_REKEY) and for Data-Security (AH [RFC4302] and/or ESP [RFC4303]) SAs may be included into SAg. Each Proposal contains a list of Transforms that the GM is able and willing to support for that protocol. Valid transform types depend on the protocol and are defined in Figure 16. Other transform types SHOULD NOT be included. The SPI length of each Proposal in an SAg is set to zero, and thus the SPI field is empty. The GCKS MUST ignore SPI length and SPI fields in the SAg payload.

Generally, a single Proposal of each type will suffice, because the group member is not negotiating Transform sets, simply alerting the GCKS to restrictions it may have. In particular, the restriction from Section 3.3 of [RFC7296] that AEAD and non-AEAD transforms must not be combined in a single proposal doesn't hold when the SAg payload is being formed. However if the GM has restrictions on combination of algorithms, this can be expressed by sending several proposals.

Proposal Num field in Proposal substructure is treated specially in SAg payload: it allows a GM to indicate that algorithms used in Rekey SA and in Data-Security (AH and/or ESP) SAs are dependent. In particular, Proposals of different types having the same value in Proposal Num field are treated as a set, so that if GCKS uses transforms from one of such Proposal for one protocol, then it MUST only use transforms from one of the Proposals with the same value in

Proposal Num field for other protocols. For example, a GM may support algorithms X and Y for both Rekey and Data-Security SAs, but with a restriction that if X is used in Rekey SA, then only X can be used in Data-Security SAs, and the same for Y. To indicate this the GM sends several Proposals marking those of them that must be used in conjunction by putting the same value in their Proposal Num field. In the simplest case when no dependency between transforms exists, all Proposals in SAg payload will have the same value in Proposal Num field.

Although the SAg payload is optional, it is RECOMMENDED for the GM to include this payload into the GSA_AUTH request to allow the GCKS to select an appropriate policy.

A GM may also indicate the support for IPcomp by inclusion one or more the IPCOMP_SUPPORTED notifications along with the SAg payload. The CPI in these notifications is set to zero and MUST be ignored by the GCKS.

Upon receiving the GSA_AUTH response, the initiator parses the response from the GCKS authenticating the exchange using the IKEv2 method, then processes the GSA and KD payloads.

The GSA payload contains the security policy and cryptographic protocols used by the group. This policy describes the optional Rekey SA (KEK), Data-security SAs (TEK), and optional Group Associated policy (GAP). If the policy in the GSA payload is not acceptable to the GM, it SHOULD notify the GCKS by initiating a GSA_REGISTRATION exchange with a NO_PROPOSAL_CHOSEN Notify payload (see Section 2.3.2). Note, that this should normally not happen if the GM includes SAg payload in the GSA_AUTH request and the GCKS takes it into account. Finally the KD payload is parsed providing the keying material for the TEK and/or KEK. The GM interprets the KD key packets, where each key packet includes the keying material for SAs distributed in the GSA payload. Keying material is matched by comparing the SPIs in the key packets to SPIs previously included in the GSA payloads. Once TEK keys and policy are matched, the GM provides them to the data security subsystem, and it is ready to send or receive packets matching the TEK policy.

The GSA KEK policy MUST include the attribute GSA_INITIAL_MESSAGE_ID with a first Message ID the GM should expect to receive if it is non-zero. The value of the attribute MUST be checked by a GM against any previously received Message ID for this group. If it is less than the previously received number, it should be considered stale and ignored. This could happen if two GSA_AUTH exchanges happened in parallel, and the Message ID changed. This attribute is used by the GM to prevent GSA_REKEY message replay attacks. The first GSA_REKEY

message that the GM receives from the GCKS must have a Message ID greater or equal to the Message ID received in the GSA_INITIAL_MESSAGE_ID attribute.

Once a GM successfully registers to the group it MUST replace any information related to this group (policy, keys) that it might have as a result of a previous registration with a new one.

Once a GM has received GIKE_REKEY policy during a registration the IKE SA may be closed. However, the GM SHOULD NOT close IKE SA, it is the GCKS who makes the decision whether to close or keep it, because depending on the policy the IKE SA may be used for inband rekeying for small groups. If inband rekeying is used, then the initial IKE SA is rekeyed (when necessary) via standard IKEv2 mechanism described in Section 1.3.2 of [RFC7296]. If for some reason this SA is teared down and no GIKE_REKEY policy was received during the registration process, the GM MUST consider itself excluded from the group. To continue participating in the group the GM should re-register.

2.3.4. GCKS Registration Operations

A G-IKEv2 GCKS passively listens for incoming requests from group members. When the GCKS receives an IKE_SA_INIT request, it selects an IKE proposal and generates a nonce and DH to include them in the IKE_SA_INIT response.

Upon receiving the GSA_AUTH request, the GCKS authenticates the group member using the same procedures as in the IKEv2 IKE_AUTH. The GCKS then authorizes the group member according to group policy before preparing to send the GSA_AUTH response. If the GCKS fails to authorize the GM, it will respond with an AUTHORIZATION_FAILED notify message. The GCKS may also respond with an INVALID_GROUP_ID notify message if the requested group is unknown to the GCKS or with an REGISTRATION_FAILED notify message if there is a problem with the requested group (for example the capacity of the group is exceeded).

The GSA_AUTH response will include the group policy in the GSA payload and keys in the KD payload. If the GCKS policy includes a group rekey option, it MUST include the GSA_INITIAL_MESSAGE_ID attribute, specifying the starting Message ID the GCKS will use when sending the GSA_REKEY message to the group members if this Message ID is non-zero. This Message ID is used to prevent GSA_REKEY message replay attacks and will be increased each time a GSA_REKEY message is sent to the group. The GCKS data traffic policy is included in the GSA TEK and keys are included in the KD TEK. The GAP MAY also be included to provide the ATD and/or DTD (Section 4.4.3.1) specifying activation and deactivation delays for SAs generated from the TEKs. If the group member has indicated that it is a sender of data traffic

and one or more Data Security SAs distributed in the GSA payload included a counter mode of operation, the GCKS responds with one or more SIDs (see Section 2.5).

[RFC5374] defines two modes of operation for multicast Data-Security SAs: transport mode and tunnel mode with address preservation. In the latter case outer source and destination addresses are taken from the inner IP packet.

If the GCKS receives a GSA_REGISTRATION exchange with a request to register a GM to a group, the GCKS will need to authorize the GM with the new group (IDg) and respond with the corresponding group policy and keys. If the GCKS fails to authorize the GM, it will respond with the AUTHORIZATION_FAILED notification. The GCKS may also respond with an INVALID_GROUP_ID or REGISTRATION_FAILED notify messages for the reasons described above.

If a group member includes an SAg in its GSA_AUTH or GSA_REGISTRATION request, the GCKS may evaluate it according to an implementation specific policy.

- o The GCKS could evaluate the list of Transforms and compare it to its current policy for the group. If the group member did not include all of the ESP or AH Transforms that match the current group policy, then the GCKS SHOULD return a NO_PROPOSAL_CHOSEN Notification.
- o The GCKS could store the list of Transforms, with the goal of migrating the group policy to a different Transforms when all of the group members indicate that they can support that Transforms.
- o The GCKS could store the list of Transforms and adjust the current group policy based on the capabilities of the devices as long as they fall within the acceptable security policy of the GCKS.

Depending on its policy, the GCKS may have no need for the IKE SA (e.g., it does not plan to initiate an GSA_INBAND_REKEY exchange). If the GM does not initiate another registration exchange or Notify (e.g., NO_PROPOSAL_CHOSEN), and also does not close the IKE SA and the GCKS is not intended to use the SA, then after a short period of time the GCKS SHOULD close the IKE SA.

2.4. Group Maintenance Channel

The GCKS is responsible for rekeying the secure group per the group policy. Rekeying is an operation whereby the GCKS provides replacement TEKs and KEK, deleting TEKs, and/or excluding group members. The GCKS may initiate a rekey message if group membership

and/or policy has changed, or if the keys are about to expire. Two forms of group maintenance channels are provided in G-IKEv2 to push new policy to group members.

GSA_REKEY The GSA_REKEY is a pseudo-exchange initiated by the GCKS, where the rekey policy is usually delivered to group members using IP multicast as a transport. This is not a real IKEv2 exchange, since no response messages are sent. This method is valuable for large and dynamic groups, and where policy may change frequently and a scalable rekeying method is required. When the GSA_REKEY is used, the IKE SA protecting the member registration exchanges is usually terminated, and group members await policy changes from the GCKS via the GSA_REKEY messages.

GSA_INBAND_REKEY The GSA_INBAND_REKEY is a normal IKEv2 exchange using the IKE SA that was setup to protecting the member registration exchange. This exchange allows the GCKS to rekey without using an independent GSA_REKEY pseudo-exchange. The GSA_INBAND_REKEY exchange provides a reliable policy delivery and is useful when G-IKEv2 is used with a small group of cooperating devices.

Depending on its policy the GCKS MAY combine these two methods. For example, it may use the GSA_INBAND_REKEY to deliver key to the GMs in the group acting as senders (as this would provide reliable keys delivery), and the GSA_REKEY for the rest GMs.

2.4.1. GSA_REKEY

The GCKS initiates the G-IKEv2 Rekey securely, usually using IP multicast. Since this rekey does not require a response and it sends to multiple GMs, G-IKEv2 rekeying MUST NOT use IKE SA windowing mechanism, described in Section 2.3 of [RFC7296]. The GCKS rekey message replaces the rekey GSA KEK or KEK array, and/or creates a new Data-Security GSA TEK. The GM_SID attribute in the Key Download payload (defined in Section 4.5.3.3) MUST NOT be part of the Rekey Exchange as this is sender specific information and the Rekey Exchange is group specific. The GCKS initiates the GSA_REKEY pseudo-exchange as following:

Members (Responder)	GCKS (Initiator)
	<-- HDR, SK{GSA, KD, [N,] [D,] [AUTH]}

Figure 9: GSA_REKEY Pseudo-Exchange

HDR is defined in Section 4.1. The Message ID in this message will start with the value the GCKS sent to the group members in the

attribute GSA_INITIAL_MESSAGE_ID or from zero if this attribute wasn't sent. The Message ID will be incremented each time a new GSA_REKEY message is sent to the group members.

The GSA payload contains the current policy for rekey and Data-Security SAs. The GSA may contain a new Rekey SA and/or a new Data-Security SAs Section 4.4.

The KD payload contains the keys for the policy included in the GSA. If the Data-Security SA is being refreshed in this rekey message, the IPsec keys are updated in the KD, and/or if the rekey SA is being refreshed in this rekey message, the rekey Key or the LKH KEK array is updated in the KD payload.

A Delete payload MAY be included to instruct the GM to delete existing SAs. See Section 4.6 for more detail.

The AUTH payload MUST be included to authenticate the GSA_REKEY message if the authentication method is based on public key signatures and MUST NOT be included if authentication is implicit. In the latter case, the fact that a GM can decrypt the GSA_REKEY message and verify its ICV proves that the sender of this message knows the current KEK, thus authenticating the sender as a member of the group. Note, that implicit authentication doesn't provide source origin authentication. For this reason using implicit authentication for GSA_REKEY is NOT RECOMMENDED unless source origin authentication is not required (for example, in a small group of highly trusted GMs). The value of the Auth Method field in the AUTH payload in the GSA_REKEY message MUST NOT be NULL Authentication.

During group member registration, the GCKS sends the authentication key in the KD payload, AUTH_KEY attribute, which the group member uses to authenticate the key server. Before the current Authentication Key expires, the GCKS will send a new AUTH_KEY to the group members in a GSA_REKEY message. The AUTH key that is sent in the rekey message may be not the same as the authentication key sent during the GM registration. If implicit authentication is used, then AUTH_KEY MUST NOT be sent to GMs.

2.4.1.1. GSA_REKEY Messages Authentication

The content of the AUTH payload depends on the authentication method and is either a digital signature or a result of prf applied to the content of the not yet encrypted GSA_REKEY message.

The authentication algorithm (prf or digital signing) is applied to the concatenation of two chunks: A and P. The chunk A lasts from the first octet of the G-IKEv2 header (not including prepended four

octets of zeros, if port 4500 is used) to the last octet of the Encrypted Payload header. The chunk P consists of the not yet encrypted content of the Encrypted payload, excluding the Initialization Vector, the Padding, the Pad Length and the Integrity Checksum Data fields (see 3.14 of [RFC7296] for description of the Encrypted payload). In other words, the P chunk is the inner payloads of the Encrypted payload in plaintext form. Figure 10 illustrates the layout of the P and A chunks in the GSA_REKEY message.

Before the AUTH payload calculation the inner payloads of Encrypted payload must be fully formed and ready for encryption except for the AUTH payload. The AUTH payload must have correct values in the Payload Header, the Auth Method and the RESERVED fields. The Authentication Data field is zeroed, but if Digital Signature authentication method is in use, then the ASN.1 Length and the AlgorithmIdentifier fields must be properly filled in, see [RFC7427].

For the purpose of the AUTH payload calculation the Length field in the IKE header and the Payload Length field in the Encrypted Payload header are adjusted so that they don't count the lengths of Initialization Vector, Integrity Checksum Data and Padding (along with Pad Length field). In other words, the Length field in the IKE header (denoted as AdjustedLen in Figure 10) is set to the sum of the lengths of A and P, and the Payload Length field in the Encrypted Payload header (denoted as AdjustedPldLen in Figure 10) is set to the length of P plus the size of the Payload header (four octets).

```
DataToAuthenticate = A | P
GsaRekeyMessage = GenIKEHDR | EncPayload
GenIKEHDR = [ four octets 0 if using port 4500 ] | AdjustedIKEHDR
AdjustedIKEHDR = SPIi | SPIr | . . . | AdjustedLen
EncPayload = AdjustedEncPldHdr | IV | InnerPlds | Pad | PadLen | ICV
AdjustedEncPldHdr = NextPld | C | RESERVED | AdjustedPldLen
A = AdjustedIKEHDR | AdjustedEncPldHdr
P = InnerPlds
```

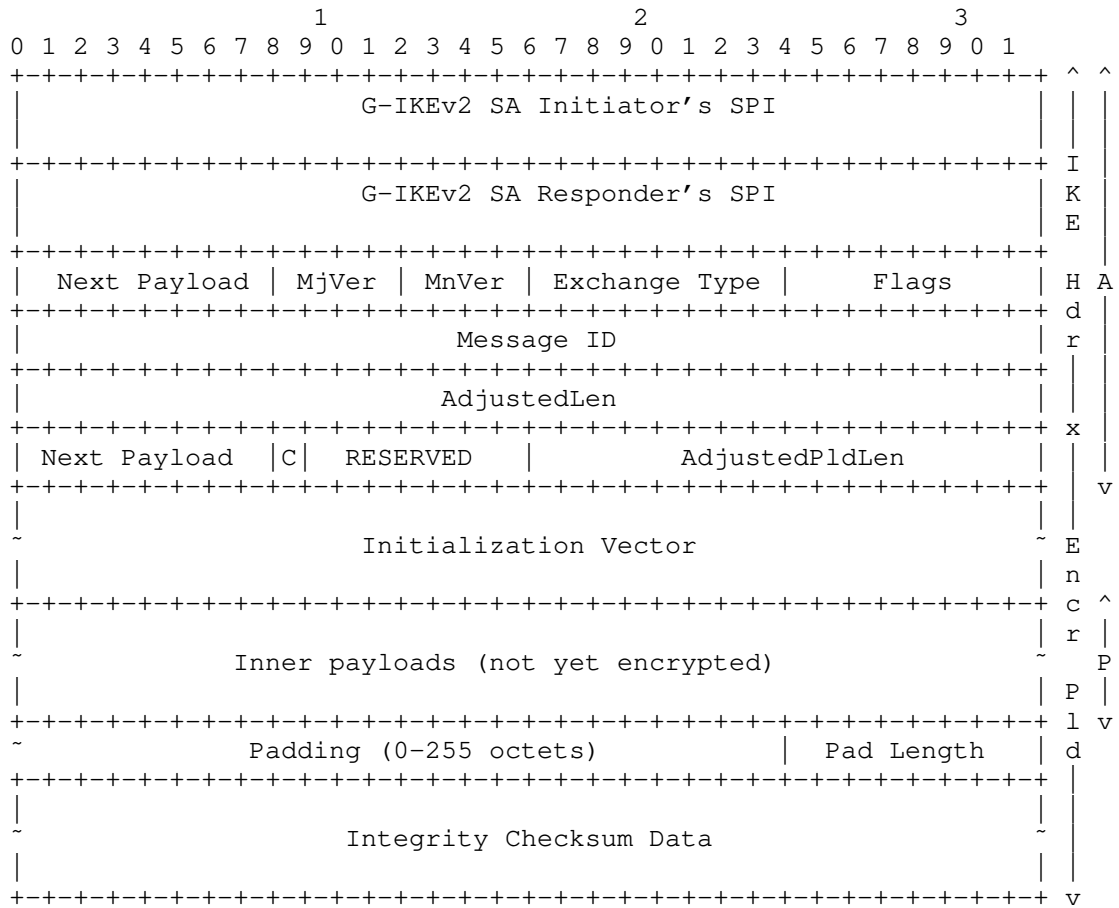


Figure 10: Data to Authenticate in the GSA_REKEY Messages

The authentication data is calculated using the authentication algorithm from the Authentication Method transform and the current authentication key provided in the AUTH_KEY attribute. Depending on the authentication method the authentication data is a digital signature or a result of applying prf from the Pseudorandom Function transform. The calculated authentication data is placed into the AUTH payload, the Length fields in the IKE Header and the Encryption Payload header are restored, the content of the Encrypted payload is encrypted and the ICV is computed using the current KEK keys.

2.4.1.2. IKE Fragmentation

IKE fragmentation [RFC7383] can be used to perform fragmentation of large GSA_REKEY messages, however when the GSA_REKEY message is emitted as an IP multicast packet there is a lack of response from the GMs. This has the following implications.

- o Policy regarding the use of IKE fragmentation is implicit. If a GCKS detects that all GMs have negotiated support of IKE fragmentation in IKE_SA_INIT, then it MAY use IKE fragmentation on large GSA_REKEY messages.
- o The GCKS must always use IKE fragmentation based on a known fragmentation threshold (unspecified in this memo), as there is no way to check if fragmentation is needed by first sending unfragmented messages and waiting for response.
- o PMTU probing cannot be performed due to lack of GSA_REKEY response message.

The calculation of authentication data MUST be applied to whole messages only, before possible IKE Fragmentation. If the message was received in fragmented form, it should be reconstructed before verifying its authenticity as if it were received unfragmented. The RESERVED field in the reconstructed Encrypted Payload header MUST be set to the value of the RESERVED field in the Encrypted Fragment payload header from the first fragment (that with Fragment Number equal to 1).

2.4.1.3. GSA_REKEY GCKS Operations

The GCKS builds the rekey message with a Message ID value that is one greater than the value included in the previous rekey message. The first message sent over a new Rekey SA must have the Message ID 0. The GSA, KD, N and D payloads follow with the same characteristics as in the GSA Registration exchange. The AUTH payload (if present) is created as defined in Section 2.4.1.1.

Because GSA_REKEY messages are not acknowledged and could be discarded by the network, one or more GMs may not receive the new policy. To mitigate such lost messages, during a rekey event the GCKS may transmit several GSA_REKEY messages with the new policy. The retransmitted messages MUST be bitwise identical and SHOULD be sent within a short time interval (a few seconds) to ensure that time-to-live would not be substantially skewed for the GMs that would receive different copies of the messages.

GCKS may also include one or several GSA_NEXT_SPI attributes specifying SPIs for the prospected rekeys, so that listening GMs are able to detect lost rekey messages and recover from this situation. See Sections Section 4.4.2.2.3 for more detail.

2.4.1.4. GSA_REKEY GM Operations

When a group member receives the Rekey Message from the GCKS it decrypts the message using the current KEK, validates its authenticity using the key retrieved in a previous G-IKEv2 exchange if AUTH payload is present, verifies the Message ID, and processes the GSA and KD payloads. The group member then downloads the new Data-Security SA and/or new Rekey SA. The parsing of the payloads is identical to the parsing done in the registration exchange.

Replay protection is achieved by a group member rejecting a GSA_REKEY message which has a Message ID smaller than the current Message ID that the GM is expecting. The GM expects the Message ID in the first GSA_REKEY message it receives to be equal or greater than the Message ID it receives in the GSA_INITIAL_MESSAGE_ID attribute. Note, that if no this attribute was received for the Rekey SA, the GM MUST assume zero as the first expected Message ID. The GM expects the Message ID in subsequent GSA_REKEY messages to be greater than the last valid GSA_REKEY message ID it received.

If the GSA payload includes a Data-Security SA using cipher in a counter-modes of operation and the receiving group member is a sender for that SA, the group member uses its current SID value with the Data-Security SAs to create counter-mode nonces. If it is a sender and does not hold a current SID value, it MUST NOT install the Data-Security SAs. It MAY initiate a GSA_REGISTRATION exchange to the GCKS in order to obtain an SID value (along with the current group policy).

Once a new Rekey SA is installed as a result of GSA_REKEY message, the current Rekey SA (over which the message was received) MUST be silently deleted after waiting DEACTIVATION_TIME_DELAY interval regardless of its expiration time. If the message includes Delete payload for existing Data-security SA, then after installing a new Data-Security SA the old one, identified by the Protocol and SPI fields in the Delete payload, MUST be silently deleted after waiting DEACTIVATION_TIME_DELAY interval regardless of its expiration time.

If a Data-Security SA is not rekeyed yet and is about to expire (a "soft lifetime" expiration is described in Section 4.4.2.1 of [RFC4301]), the GM SHOULD initiate a registration to the GCKS. This registration serves as a request for current SAs, and will result in the download of replacement SAs, assuming the GCKS policy has created

them. A GM SHOULD also initiate a registration request if a Rekey SA is about to expire and not yet replaced with a new one.

2.4.2. GSA_INBAND_REKEY Exchange

When the IKE SA protecting the member registration exchange is maintained while group member participates in the group, the GCKS can use the GSA_INBAND_REKEY exchange to individually provide policy updates to the group member.

Member (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK{GSA, KD, [N,] [D]}
HDR, SK{}	-->

Figure 11: GSA_INBAND_REKEY Exchange

Because this is a normal IKEv2 exchange, the HDR is treated as defined in [RFC7296].

2.4.2.1. GSA_INBAND_REKEY GCKS Operations

The GSA, KD, N and D payloads are built in the same manner as in a registration exchange.

2.4.2.2. GSA_INBAND_REKEY GM Operations

The GM processes the GSA, KD, N and D payloads in the same manner as if they were received in a registration exchange.

2.4.3. Deletion of SAs

There are occasions when the GCKS may want to signal to group members to delete policy at the end of a broadcast, or if group policy has changed. Deletion of SAs is accomplished by sending the G-IKEv2 Delete Payload [RFC7296], section 3.11 as part of the GSA_REKEY pseudo-exchange as shown below.

Members (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK{[GSA,] [KD,] [N,] [D,] [AUTH]}

Figure 12: SA Deletion in GSA_REKEY

If GCKS has a unicast SA with group member then it can use the GSA_INBAND_REKEY exchange to delete SAs.

Member (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK{[GSA,] [KD,] [N,] [D,]}
HDR, SK{}	-->

Figure 13: SA Deletion in GSA_INBAND_REKEY

The GCKS MAY specify the remaining active time of the policy by using the GAP_DTD attribute in the GSA GAP substructure. If a GCKS has no further SAs to send to group members, the GSA and KD payloads MUST be omitted from the message.

There may be circumstances where the GCKS may want to start over with a clean state, for example in case it runs out of available SIDs. The GCKS can signal deletion of all the Data-security SAs by sending a Delete payload with an SPI value equal to zero. For example, if the GCKS wishes to remove the Rekey SA and all the Data-security SAs, the GCKS sends a Delete payload with an SPI of zero and Protocol ID of AH or ESP, followed by another Delete payload with a SPI of zero and Protocol ID of GIKE_REKEY.

If a group member receives a Delete payload with zero SPI and protocol ID of GIKE_REKEY either via multicast Rekey SA or via unicast SA using the GSA_INBAND_REKEY exchange, it means that the group member is excluded from the group. The group member MUST re-register if it wants to continue participating in this group. The registration is performed as described in Section 2.3. Note, that if the GSA_INBAND_REKEY exchange is used to exclude a group member from the group, and thus the unicast SA between the group member and the GCKS exists, then this SA persists after this exchange and the group member may use the GSA_REGISTRATION exchange to re-register.

2.5. Counter-based modes of operation

Several counter-based modes of operation have been specified for ESP (e.g., AES-CTR [RFC3686], AES-GCM [RFC4106], AES-CCM [RFC4309], ChaCha20-Poly1305 [RFC7634], AES-GMAC [RFC4543]) and AH (e.g., AES-GMAC [RFC4543]). These counter-based modes require that no two senders in the group ever send a packet with the same Initialization Vector (IV) using the same cipher key and mode. This requirement is met in G-IKEv2 when the following requirements are met:

- o The GCKS distributes a unique key for each Data-Security SA.
- o The GCKS uses the method described in [RFC6054], which assigns each sender a portion of the IV space by provisioning each sender with one or more unique SID values.

2.5.1. Allocation of SIDs

When at least one Data-Security SA included in the group policy includes a counter-based mode of operation, the GCKS automatically allocates and distributes one SID to each group member acting in the role of sender on the Data-Security SA. The SID value is used exclusively by the group sender to which it was allocated. The group sender uses the same SID for each Data-Security SA specifying the use of a counter-based mode of operation. A GCKS MUST distribute unique keys for each Data-Security SA including a counter-based mode of operation in order to maintain unique key and nonce usage.

During registration, the group sender can choose to request one or more SID values. Requesting a value of 1 is not necessary since the GCKS will automatically allocate exactly one to the group sender. A group sender MUST request as many SIDs matching the number of encryption modules in which it will be installing the TEKs in the outbound direction. Alternatively, a group sender MAY request more than one SID and use them serially. This could be useful when it is anticipated that the group sender will exhaust their range of Data-Security SA nonces using a single SID too quickly (e.g., before the time-based policy in the TEK expires).

When the group policy includes a counter-based mode of operation, a GCKS SHOULD use the following method to allocate SID values, which ensures that each SID will be allocated to just one group sender.

1. A GCKS maintains an SID-counter, which records the SIDs that have been allocated. SIDs are allocated sequentially, with zero as the first allocated SID.
2. Each time an SID is allocated, the current value of the counter is saved and allocated to the group sender. The SID-counter is then incremented in preparation for the next allocation.
3. When the GCKS specifies a counter-based mode of operation in the Data-Security SA a group sender may request a count of SIDs during registration in a Notify payload information of type SENDER. When the GCKS receives this request, it increments the SID-counter once for each requested SID, and distributes each SID value to the group sender. The GCKS SHOULD have a policy-defined upper bound for the number of SIDs that it will return irrespective of the number requested by the GM.
4. A GCKS allocates new SID values for each registration operation by a group sender, regardless of whether the group sender had previously contacted the GCKS. In this way, the GCKS is not required to maintaining a record of which SID values it had

previously allocated to each group sender. More importantly, since the GCKS cannot reliably detect whether the group sender had sent data on the current group Data-Security SAs it does not know what Data-Security counter-mode nonce values that a group sender has used. By distributing new SID values, the key server ensures that each time a conforming group sender installs a Data-Security SA it will use a unique set of counter-based mode nonces.

5. When the SID-counter maintained by the GCKS reaches its final SID value, no more SID values can be distributed. Before distributing any new SID values, the GCKS MUST exclude all group members from the group as described in Section 2.4.3. This will result in the group members performing re-registration, during which they will receive new Data-Security SAs and group senders will additionally receive new SID values. The new SID values can safely be used because they are only used with the new Data-Security SAs.

2.5.2. GM Usage of SIDs

A GM applies the SID to Data-Security SA as follows.

- o The most significant bits NUMBER_OF_SID_BITS of the IV are taken to be the SID field of the IV.
- o The SID is placed in the least significant bits of the SID field, where any unused most significant bits are set to zero. If the SID value doesn't fit into the NUMBER_OF_SID_BITS bits, then the GM MUST treat this as a fatal error and re-register to the group.

2.6. Replay Protection for Multicast Data-Security SAs

IPsec provides replay protection as part of its security services. With multicast extension for IPsec replay protection is not always possible to achieve (see Section 6.1 of [RFC3740]). In particular, if there are many group senders for a Data-Security SA, then each of them will independently increment the Sequence Number field in the ESP header (see Section 2 of [RFC4303]) thus making impossible for the group receivers to filter out replayed packets. However, if there is only one group sender for a Data-Security SA, then it is possible to achieve replay protection with some restrictions (see Section 4.4.2.1.3). The GCKS may create several Data-Security SAs with the same traffic selectors allowing only a single group sender in each SA if it is desirable to get replay protection with multiple (but still limited number) of group senders.

In IPsec architecture assumes that it is a local matter for an IPsec receiver whether replay protection is active or not. In other words, an IPsec sender always increments the Sequence Number field in the ESP header and a receiver decides whether to check for replayed packets or not. With multicast extension for IPsec this approach generally isn't applicable, since group members don't know how many group senders exist for a particular Data-Security SA. For this reason the status or replay protection must be part of the policy downloaded to GMs by GCKS.

For this purpose this specification re-uses the Extended Sequence Numbers transform, defined in Section 3.3.2 [RFC7296]. This specification renames this transform to "Replay Protection" and adds a new value for possible Transform IDs: "Not Used" (<TBA by IANA>). The GCKS MUST include this transform in the GSA payload for every Data-Security SA. Note, that this specification prohibits using Extended Sequence Numbers (see Section 4.4.2.1.3).

3. Group Key Management and Access Control

Through the G-IKEv2 rekey, G-IKEv2 supports algorithms such as Logical Key Hierarchy (LKH) that have the property of denying access to a new group key by a member removed from the group (forward access control) and to an old group key by a member added to the group (backward access control). An unrelated notion to PFS, "forward access control" and "backward access control" have been called "perfect forward security" and "perfect backward security" in the literature [RFC2627].

Group management algorithms providing forward and backward access control other than LKH have been proposed in the literature, including OFT [OFT] and Subset Difference [NNL]. These algorithms could be used with G-IKEv2, but are not specified as a part of this document.

The Group Key Management Method transform from the GSA policy specifies how members of the group obtain group keys. This document specifies a single method for the group key management -- Wrapped Key Download. This method assumes that all group keys are sent to the GMs by the GCKS encrypted with some other keys, called Key Wrap Keys (KWK).

3.1. Key Wrap Keys

Every GM always knows at least one KWK -- the KWK that is associated with the IKE SA or multicast Rekey SA the wrapped keys are sent over. In this document it is called default KWK and is denoted as GSK_w.

The GCKS may also send other keys to GMs that will be used as Key Wrap Keys for the purpose of building key hierarchy. Each KWK is associated with an encryption algorithm from the Encryption Algorithm transform used for the SA the key is sent over. The size of a KWK MUST be of the size of the key for this Encryption Algorithm transform (taking into consideration the Key Length attribute for this transform if present). This association persists even if the key is used later in the context of another SA with possibly different Encryption Algorithm transform.

To have an ability to provide forward access control the GCKS provides each GM with a personal key at the time of registration. Besides, several intermediate keys that form a key hierarchy and are shared among several GMs may be provided by the GCKS.

3.1.1. Default Key Wrap Key

The default KWK (GSK_w) is only used in the context of a single IKE SA. Every IKE SA (unicast IKE SA or multicast Rekey SA) will have its own GSK_w. The GSK_w is used with the algorithm from the Encryption Algorithm transform for the SA the GSK_w is used in the context of. The size of GSK_w MUST be of the key size of this Encryption Algorithm transform (taking into consideration the Key Length attribute for this transform if present).

For the unicast IKE SA (used for the GM registration and for the GSA_INBAND_REKEY exchanges, if they are take place) the GSK_w is computed as follows:

GSK_w = prf+(SK_d, "Key Wrap for G-IKEv2")

where the string "Key Wrap for G-IKEv2" is 20 ASCII characters without null termination.

For the multicast Rekey SA the GSK_w is provided along with other SA keys as defined in Section 3.4.

3.2. GCKS Key Management Semantics

Wrapped Key Download method allows the GCKS to employ various key management methods

- o A simple key management methods -- when the GCKS always sends group SA keys encrypted with the GSK_w.
- o An LKH key management method -- when the GCKS provides each GM with an individual key at the time of the GM registration (encrypted with GSK_w). Then the GCKS forms an hierarchy of keys

so that the group SA keys are encrypted with other keys which are encrypted with other keys and so on, tracing back to the individual GMS' keys.

Other key policies may also be employed by the GCKS.

3.2.1. Forward Access Control Requirements

When group membership is altered using a group management algorithm new Data-Security SAs and their associated keys are usually also needed. New Data-Security SAs and keys ensure that members who were denied access can no longer participate in the group.

If forward access control is a desired property of the group, new TEK policy and the associated keys MUST NOT be included in a G-IKEv2 rekey message which changes group membership. This is required because the GSA TEK policy and the associated keys are not protected with the new KEK. A second G-IKEv2 rekey message can deliver the new GSA TEKS and their associated keys because it will be protected with the new KEK, and thus will not be visible to the members who were denied access.

If forward access control policy for the group includes keeping group policy changes from members that are denied access to the group, then two sequential G-IKEv2 rekey messages changing the group KEK MUST be sent by the GCKS. The first G-IKEv2 rekey message creates a new KEK for the group. Group members, which are denied access, will not be able to access the new KEK, but will see the group policy since the G-IKEv2 rekey message is protected under the current KEK. A subsequent G-IKEv2 rekey message containing the changed group policy and again changing the KEK allows complete forward access control. A G-IKEv2 rekey message MUST NOT change the policy without creating a new KEK.

If other methods of using LKH or other group management algorithms are added to G-IKEv2, those methods MAY remove the above restrictions requiring multiple G-IKEv2 rekey messages, providing those methods specify how the forward access control policy is maintained within a single G-IKEv2 rekey message.

3.3. GM Key Management Semantics

This specification defines a GM Key Management semantics in such a way, that it doesn't depend on the key management method employed by the GCKS. This allows having all the complexity of key management in the GCKS, which is free to implement various key management methods, such as direct transmitting of group SA keys or using some kind of

key hierarchy (e.g. LKH). For all these policies the GM behavior is the same.

Each key that a GM receives in G-IKEv2 is identified by a 32-bit number called Key ID. Zero Key ID has a special meaning -- it always contains keying material from which the keys for protecting Data-Security SAs and Rekey SA are taken.

All keys in G-IKEv2 are transmitted in encrypted form, as specified in Section 4.5.1. This format includes a Key ID (ID of a key that is encrypted) and a KWK ID (ID of a key that was used to encrypt this key). Keys may be encrypted either with default KWK (GSK_w) or with other keys, which the GM has received in the WRAP_KEY attributes. If a key was encrypted with GSK_w, then the KWK ID field is set to zero, otherwise the KWK ID field identifies the key used for encryption.

When a GM receives a message from the GCKS installing new Data-Security or Rekey SA, it will contain a KD payload with an SA_KEY attribute containing keying material for this SA. For a Data-Security SA exactly one SA_KEY attribute will be present with both Key ID and KWK ID fields set to zero. This means that the default KWK (GSK_w) should be used to extract this keying material.

For a multicast Rekey SA multiple SA_KEY attributes may be present depending on the key management method employed by the GCKS. If multiple SA_KEY attributes are present then all of them MUST contain the same keying material encrypted using different KWKs. The GM in general is unaware of the GCKS's key management method and can always use the same procedure to get the keys. In particular, the GM's task is to find a way to decrypt at least one of the SA_KEY attributes using either the GSK_w or the keys from the WRAP_KEY attributes that are present in the same message or were received in previous messages.

We will use the term "Key Path" to describe an ordered sequence of keys where each subsequent key was used to encrypt the previous one. The GM keeps its own Key Path (called Working Key Path) in the memory associated with each group it is registered to and updates it when needed. When the GSA_REKEY message is received the GM processes the received SA_KEY attributes one by one trying to construct a new key path that starts from this attributes and ends with any key in the Working Key Path or with the default KWK (GSK_w).

In the simplest case the SA_KEY attribute is encrypted with GSK_w so that the new Key Path is empty. If more complex key management methods are used then a Key Path will contain intermediate keys from the WRAP_KEY attributes received by a GM so far starting from its registration to the group. If the GM is able to construct a new Key

Path using intermediate keys it has, then it is able to decrypt the SA_KEY attribute and use its content to form new SA keys. If it is unable to build a new Key Path, then it means that the GM is excluded from the group.

Depending on the new Key Path the GM should do the following actions to be prepared for future key updates:

- o If the new Key Path is empty then no actions are needed. This may happen if no WRAP_KEY attributes from the received message were used.
- o If the new Key Path is non-empty and it ends with the default KWK (GSK_w), then the whole new Key Path is stored by the GM as the GM's Working Key Path. This situation may only happen at the time the GM is registering to the group, when the GCKS is providing it with its personal key and the other keys from the key tree that are needed for this GM. These keys form an initial Working Key Path for this GM.
- o In all other cases the new Key Path will end at some intermediate key from the GM's current Working Key Path. In this case the new Key Path is constructed by replacing a part of the GM's current Working Key Path from the beginning and up to (but not including) the key that the GM has used to decrypt the last key in the new Key Path.

Appendix A contains an example of how this algorithm works in case of LKH key management method.

3.4. SA Keys

Keys used for Data-Security SAs or Rekey SA (called here SA keys) are downloaded to GMs in the form of keying material. The keys for each algorithm employed in an SA are taken from this keying material as if they were concatenated to form it.

For a Data-Security SA the keys are taken in accordance to the third bullet from Section 2.17 of [RFC7296]. In particular, for the ESP and AH SAs the encryption key (if any) MUST be taken from the first bits of the keying material and the integrity key (if any) MUST be taken from the remaining bits.

For a Rekey SA the following keys are taken from the keying material:

GSK_e | GSK_a | GSK_w = KEYMAT

where GSK_e and GSK_a are the keys used for the Encryption Algorithm and the Integrity Algorithm transforms for the corresponding SA and GSK_w is a default KWK for this SA. Note, that GSK_w is also used with the Encryption Algorithm transform as well as GSK_e. If an AEAD algorithm is used for encryption, then SK_a key will not be used (GM can use the formula above assuming the length of SK_a is zero).

4. Header and Payload Formats

The G-IKEv2 is an IKEv2 extension and thus inherits its wire format for data structures. However, the processing of some payloads are different and several new payloads are defined: Group Identification (IDg), Group Security Association (GSA) Key Download (KD). New exchange types GSA_AUTH, GSA_REGISTRATION, GSA_REKEY and GSA_INBAND_REKEY are also added.

This section describes new payloads and the differences in processing of existing IKEv2 payloads.

4.1. G-IKEv2 Header

G-IKEv2 uses the same IKE header format as specified in [RFC7296] section 3.1. Major Version is 2 and Minor Version is 0 as in IKEv2. IKE SA Initiator's SPI, IKE SA Responder's SPI, Flags, Message ID, and Length are as specified in [RFC7296].

4.2. Group Identification Payload

The Group Identification (IDg) payload allows the group member to indicate which group it wants to join. The payload is constructed by using the IKEv2 Identification Payload (section 3.5 of [RFC7296]). ID type ID_KEY_ID MUST be supported. ID types ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, ID_IPV6_ADDR SHOULD be supported. ID types ID_DER_ASN1_DN and ID_DER_ASN1_GN are not expected to be used. The Payload Type for the Group Identification payload is fifty (50).

4.3. Security Association - GM Supported Transforms Payload

The Security Association - GM Supported Transforms Payload (SAG) payload declares which Transforms a GM is willing to accept. The payload is constructed using the format of the IKEv2 Security Association payload (section 3.3 of [RFC7296]). The Payload Type for SAG is identical to the SA Payload Type -- thirty-three (33).

Specifying multiple GSA TEKs allows multiple related data streams (e.g., video, audio, and text) to be associated with a session, but each protected with an individual security association policy.

A GAP allows for the distribution of group-wise policy, such as instructions for when to activate and de-activate SAs.

Policies are distributed in substructures to the GSA payload. The format of the substructures is defined below in Section 4.4.2 (for GSA policy) and in Section 4.4.3 (for GA policy). The first octet of the substructure unambiguously determines its type -- it is zero for GAP and non-zero (actually, it is a security protocol ID) for GSA policies.

4.4.2. Group Security Association Policy Substructure

The GSA policy substructure contains parameters for the SA used with this group. Depending on the security protocol the SA is either a Rekey SA or a Data-Security SA (ESP and AH). It is NOT RECOMMENDED that the GCKS distribute both ESP and AH policies for the same set of Traffic Selectors.

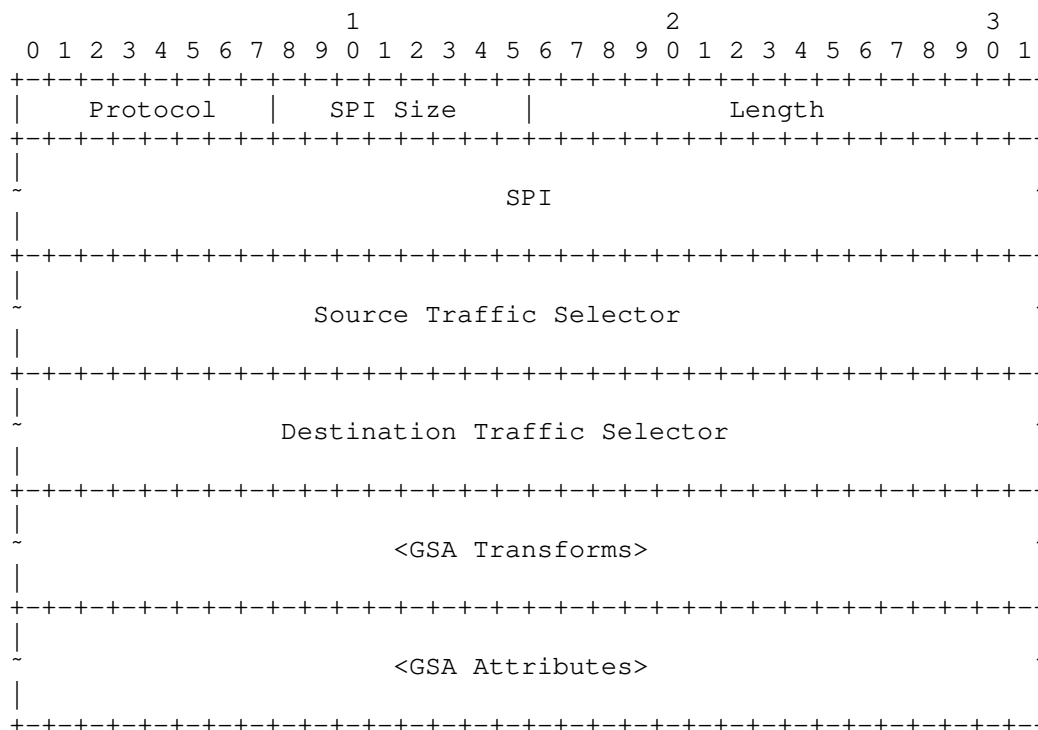


Figure 15: GSA Policy Substructure Format

The GSA policy fields are defined as follows:

- o Protocol (1 octet) -- Identifies the security protocol for this group SA. The values are defined in the IKEv2 Security Protocol Identifiers in [IKEV2-IANA]. The valid values for this field are: <TBA> (GIKE_REKEY) for Rekey SA and 2 (AH) or 3 (ESP) for Data-Security SAs.
- o SPI Size (1 octet) -- Size of Security Parameter Index (SPI) for the SA. SPI size depends on the SA protocol. For GIKE_REKEY it is 16 octets, while for AH and ESP it is 4 octets.
- o Length (2 octets, unsigned integer) -- Length of this substructure including the header.
- o SPI (variable) -- Security Parameter Index for the group SA. The size of this field is determined by the SPI Size field. As described above, these SPIs are assigned by the GCKS. In case of GIKE_REKEY the SPI must be the IKEv2 Header SPI pair where the first 8 octets become the "Initiator's SPI" field in the G-IKEv2

rekey message IKEv2 HDR, and the second 8 octets become the "Responder's SPI" in the same HDR. When selecting SPI the GCKS MUST make sure that the sole first 8 octets (corresponding to "Initiator's SPI" field in the IKEv2 header) uniquely identify the Rekey SA.

- o Source & Destination Traffic Selectors - (variable) -- Substructures describing the source and destination of the network identities. The format for these substructures is defined in IKEv2 [RFC7296], section 3.13.1. For the Rekey SA (with the GIKE_REKEY protocol) the destination traffic selectors MUST define a single multicast IP address, an IP protocol (assumed to be UDP) and a single port the GSA_REKEY messages will be destined to. The source traffic selector in this case MUST either define a single IP address, an IP protocol (assumed to be UDP) and a single port the GSA_REKEY messages will be originated from or be a wildcard selector. For the Data-Security (AH and ESP) SAs the destination traffic selectors SHOULD define a single multicast IP address. The source traffic selector in this case SHOULD define a single IP address or be a wildcard selector. IP protocol and ports define the characteristics of traffic protected by this Data-Security SA. If the Data-Security SAs are created in tunnel mode, then it MUST BE tunnel mode with address preservation (see [RFC5374]). UDP encapsulation [RFC3948] is not used for the multicast Data-Security SAs.
- o GSA Transforms (variable) -- A list of Transform Substructures specifies the policy information for the SA. The format is defined in IKEv2 [RFC7296], section 3.3.2. The Last Substruc value in each Transform Substructure will be set to 3 except for the last one in the list, which is set to 0. Section 4.4.2.1 describes using IKEv2 transforms in GSA policy substructure.
- o GSA Attributes (variable) -- Contains policy attributes associated with the group SA. The following sections describe the possible attributes. Any or all attributes may be optional, depending on the protocol and the group policy. Section 4.4.2.2 defines attributes used in GSA policy substructure.

4.4.2.1. GSA Transforms

GSA policy is defined by means of transforms in the GSA policy substructure. For this purpose the transforms defined in [RFC7296] are used. In addition, new transform types are defined for using in G-IKEv2: Authentication Method (AUTH) and Group Key Management Method (GKM), see Section 8.

Valid Transform Types depend on the SA protocol and are summarized in the table below.

Protocol	Mandatory Types	Optional Types
GIKE_REKEY	ENCR, INTEG*, PRF, AUTH**, GKM**	
ESP	ENCR	INTEG, RP
AH	INTEG	RP

Figure 16: Valid Transform Types

(*) If AEAD encryption algorithm is used, then INTEG transform MUST NOT be specified, otherwise it MUST be specified.

(**) May only appear at the time of a GM registration, (in the GSA_aUTH and GSA_REGISTRATION exchanges).

4.4.2.1.1. Authentication Method Transform

The Authentication Method (AUTH) transform is used in the GIKE_REKEY policy to convey information of how GCKS will authenticate the GSA_REKEY messages. This values are from the IKEv2 Authentication Method registry [IKEV2-IANA]. Note, that this registry defines only values in a range 0-255, so even that Transform ID field in the Transform substructure allows for 65536 possible values, in case of the Authentication Method transform the values 256-65535 MUST NOT appear.

Among the currently defined authentication methods in the IKEv2 Authentication Method registry, only the following are allowed to be used in the Authentication Method transform: NULL Authentication and Digital Signature. Other currently defined authentication methods MUST NOT be used. The following semantics is associated with each of the allowed methods.

NULL Authentication -- No additional authentication of the GSA_REKEY messages will be provided by the GCKS besides the ability for the GMs to correctly decrypt them and verify their ICV. In this case the GCKS MUST NOT include the AUTH_KEY attribute into the KD payload. Additionally, the AUTH payload MUST NOT be included in the GIKE_REKEY messages.

Digital Signature -- Digital signatures will be used by the GCKS to authenticate the GSA_REKEY messages. In this case the GCKS MUST include the AUTH_KEY attribute containing the public key into the KD payload at the time the GM is registered to the group. To specify the details of the signature algorithm a new attribute Algorithm Identifier (<TBA by IANA>) is defined.

This attribute contains DER-encoded ASN.1 object AlgorithmIdentifier, which would specify the signature algorithm and the hash function that the GCKS will use for authentication. The AlgorithmIdentifier object is defined in section 4.1.1.2 of [RFC5280], see also [RFC7427] for the list of common AlgorithmIdentifier values used in IKEv2. In case of using digital signature the GCKS MUST include the Algorithm Identifier attribute in the Authentication Method transform.

The authentication method MUST NOT change as a result of rekey operations. This means that the Authentication Method transform may not appear in the rekey messages, it may only appear in the registration exchange (either GSA_AUTH or GSA_REGISTRATION).

The type of the Authentication Method Transform is <TBA by IANA>.

4.4.2.1.2. Group Key Management Method Transform

The Group Key Management Method (GKM) transform is used in the GIKE_REKEY policy to convey information of how GCKS will manage the group keys to provide forward and backward access control (i.e., used to exclude group members). Possible key management methods are defined in a new IKEv2 registry "Transform Type <TBA> -- Group Key Management Methods" (see Section 8). This document defines one values for this registry:

Wrapped Key Download (<TBA by IANA>) -- Keys are downloaded by GCKS to the GMs in encrypted form. This algorithm may provide forward and backward access control if some form of key hierarchy is used and each GM is provided with a personal key at the time of registration. Otherwise no access control is provided.

The group key management method MUST NOT change as a result of rekey operations. This means that the Group Key Management Method transform may not appear in the rekey messages, it may only appear in the registration exchange (either GSA_AUTH or GSA_REGISTRATION).

The type of the Group Key Management Method transform is <TBA by IANA>.

4.4.2.1.3. Replay Protection Transform

The "Extended Sequence Number (ESN)" Transform is defined in [RFC7296]. This specification renames this transform to "Replay Protection (RP)". This transform allows to specify whether the 64-bit Extended Sequence Numbers (ESN) are to be used in ESP and AH.

Since both AH [RFC4302] and ESP [RFC4303] are defined in such a way, that high-order 32 bits of extended sequence numbers are never transmitted, it makes using ESN in multicast Data-Security SAs problematic, because GMs that join group long after it is created will have to somehow learn the current high order 32 bits of ESN for each sender in the group. The algorithm for doing this described in [RFC4302] and [RFC4303] is resource-consuming and is only suitable when a receiver is able to guess the high-order 32 bits close enough to its real value, which is not the case for multicast SAs. For this reason extended sequence numbers MUST NOT be used for multicast Data-Security SAs and thus the value "Extended Sequence Numbers" (1) for the Replay Protection transform type MUST NOT be used in the GSA Payload. The GCKS MUST estimate the data rate and rekey Data-Security SAs frequently enough so that Sequence Numbers (SN) don't wrap.

4.4.2.2. GSA Attributes

GSA attributes are generally used to provide GMs with additional parameters for the GSA policy. Unlike security parameters distributed via transforms, which are expected not to change over time (unless policy changes), the parameters distributed via GSA attributes may depend on the time the provision takes place, on the existence of others group SAs or on other conditions.

This document creates a new IKEv2 IANA registry for the types of the GSA attributes which is initially filled as described in Section 8. In particular, the following attributes are initially added.

GSA Attributes	Value	Type	Multiple	Protocol
Reserved	0			
GSA_KEY_LIFETIME	1	V	N	GIKE_REKEY, AH, ESP
GSA_INITIAL_MESSAGE_ID	2	V	N	GIKE_REKEY
GSA_NEXT_SPI	3	V	Y	GIKE_REKEY, AH, ESP

The attributes must follow the format defined in the IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V).

4.4.2.2.1. GSA_KEY_LIFETIME Attribute

The GSA_KEY_LIFETIME attribute (1) specifies the maximum time for which the SA is valid. The value is a 4 octet unsigned integer in a network byte order, specifying a valid time period in seconds. A single attribute of this type MUST be included into any GSA policy substructure.

When the lifetime expires, the group security association and all associated keys MUST be deleted. The GCKS may delete the SA at any time before the end of the validity period.

This attribute SHOULD NOT be used if inband rekeying (via the GSA_INBAND_REKEY exchange) is employed by the GCKS for the GM.

4.4.2.2.2. GSA_INITIAL_MESSAGE_ID Attribute

The GSA_INITIAL_MESSAGE_ID attribute (2) defines the initial Message ID to be used by the GCKS in the GSA_REKEY messages. The Message ID is a 4 octet unsigned integer in network byte order.

A single attribute of this type MUST be included into the GSA KEK policy substructure if the initial Message ID of the Rekey SA is non-zero. Note, that it is always the case if GMs join the group after some multicast rekey operations have already taken place, so in these cases this attribute will be included into the GSA policy at the time of GMs' registration.

This attribute MUST NOT be used if inband rekeying (via the GSA_INBAND_REKEY exchange) is employed by the GCKS for the GM.

4.4.2.2.3. GSA_NEXT_SPI Attribute

The optional GSA_NEXT_SPI attribute (3) contains SPI that the GCKS reserved for the next Rekey SA or Data-Security SAs replacing the current ones. The length of the attribute data is determined by the SPI Size field in the GSA Policy substructure the attribute resides in (see Section 4.4.2), and the attribute data contains SPI as it would appear on the network. Multiple attributes of this type MAY be included, meaning that any of the supplied SPIs can be used in the replacement group SA.

The GM MAY store these values and if later the GM starts receiving messages with one of these SPIs without seeing a rekey message over the current Rekey SA, this may be used as an indication, that the rekey message got lost on its way to this GM. In this case the GM SHOULD re-register to the group.

Note, that this method of detecting lost rekey messages can only be used by group receivers. Additionally there is no point to include this attribute in the GSA_INBAND_REKEY messages, since they use reliable transport. Note also, that the GCKS is free to forget its promises and not to use the SPIs it sent in the GSA_NEXT_SPI attributes before (e.g. in case of the GCKS is rebooted), so the GM must only treat these information as a "best effort" made by the GCKS to prepare for future rekeys.

This attribute MUST NOT be used if inband rekeying (via the GSA_INBAND_REKEY exchange) is employed by the GCKS for the GM.

4.4.3. Group Associated Policy Substructure

Group specific policy that does not belong to any SA policy can be distributed to all group member using Group Associated Policy (GAP) substructure.

The GAP substructure is defined as follows:

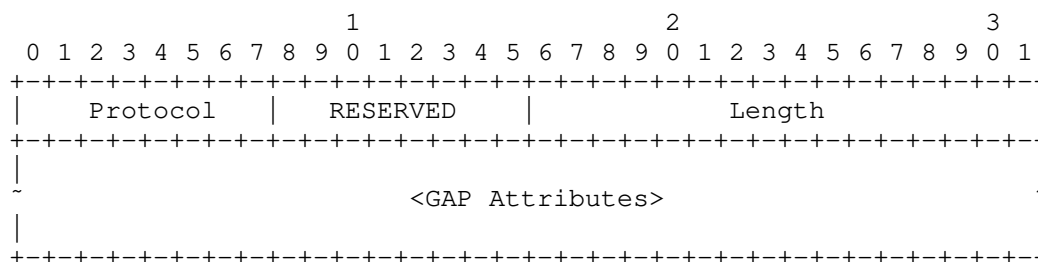


Figure 17: GAP Substructure Format

The GAP substructure fields are defined as follows:

- o Protocol (1 octet) -- MUST be zero. This value is reserved in Section 8 and is never used for any security protocol, so it is used here to indicate that this substructure contains policy not related to any specific protocol.
- o RESERVED (octet) -- MUST be zero on transmission, MUST be ignored on receipt.
- o Length (2 octets, unsigned integer) -- Length of this substructure including the header.
- o GAP Attributes (variable) -- Contains policy attributes associated with no specific SA. The following sections describe possible attributes. Any or all attributes may be optional, depending on the group policy.

This document creates a new IKEv2 IANA registry for the types of the GAP attributes which is initially filled as described in Section 8. In particular, the following attributes are initially added.

GAP Attributes	Value	Type	Multiple
Reserved	0		
GAP_ATD	1	B	N
GAP_DTD	2	B	N
GAP_SID_BITS	3	B	N

The attributes must follow the format defined in the IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V).

4.4.3.1. GAP_ATD And GAP_DTD Attributes

Section 4.2.1 of [RFC5374] specifies a key rollover method that requires two values be provided to group members -- Activation Time Delay (ATD) and Deactivation Time Delay (DTD).

The GAP_ATD attribute (1) allows a GCKS to set the Activation Time Delay for Data-Security SAs of the group. The ATD defines how long active members of the group (those who sends traffic) should wait after receiving new SAs before starting sending traffic over them. Note, that to achieve smooth rollover passive members of the group should activate the SAs immediately once they receive them.

The GAP_DTD attribute (2) allows the GCKS to set the Deactivation Time Delay for previously distributed SAs. The DTD defines how long after receiving a request to delete Data-Security SAs passive group members should wait before actually deleting them. Note that active members of the group should stop sending traffic over these old SAs once new replacement SAs are activated (after time specified in the GAP_ATD attribute).

The GAP_ATD and GAP_DTD attributes contain 16 bit unsigned integer in a network byte order, specifying the delay in seconds. These attributes are OPTIONAL. If one of them or both are not sent by the GCKS, then no corresponding delay should be employed.

4.4.3.2. GAP_SID_BITS Attribute

The GAP_SID_BITS attribute (3) declares how many bits of the cipher nonce are taken to represent an SID value. The bits are applied as the most significant bits of the IV, as shown in Figure 1 of [RFC6054] and specified in Section 2.5.2. Guidance for a GCKS choosing the NUMBER_OF_SID_BITS is provided in Section 3 of [RFC6054]. This value is applied to each SID value distributed in the KD payload.

The GCKS MUST include this attribute if there are more than one sender in the group and any of the Data-Security SAs use counter-based cipher mode. The number of SID bits is represented as 16 bit unsigned integer in network byte order.

4.5. Key Download Payload

The Key Download (KD) payload contains the group keys for the SAs specified in the GSA Payload. The Payload Type for the Key Download payload is fifty-two (52).

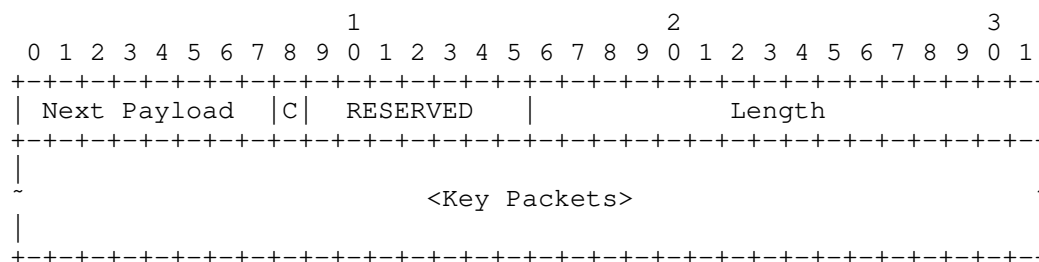


Figure 18: Key Download Payload Format

The Key Download payload fields are defined as follows:

- o Next Payload, C, RESERVED, Payload Length fields comprise the IKEv2 Generic Payload Header and are defined in Section 3.2. of [RFC7296].
- o Key Packets (variable) -- Contains Group Key Packet and Member Key Packet substructures. Each Key Packet contains keys for a single group rekey or Data-Security SA or a keys and security parameters for a GM.

Two types of Key Packets are used -- Group Key Packet and Member Key Packet.

4.5.1. Wrapped Key Format

The symmetric keys in G-IKEv2 are never sent in clear. They are always encrypted with other keys using the format called Wrapped Key that is shown below (Figure 19).

The keys are encrypted using algorithm that is used to encrypt the message the keys are sent in. It means, that in case of unicast IKE SA (used for GMs registration and rekeying using GSA_INBAND_REKEY) the encryption algorithm will be the one negotiated during the IKE SA establishment, while for a GSA_REKEY message the algorithm will be

provided by the GCKS in the Encryption Algorithm transform in the GSA payload when this multicast SA was being established.

If AEAD mode is used for encryption, then for the purpose of key encryption the authentication tag **MUST NOT** be used (both not calculated and not verified), since the G-IKEv2 provides authentication of all its messages. In addition there is no AAD in this case. If encryption algorithm requires padding, then the encrypted key **MUST** be padded before encryption to have the required size. If the encryption algorithm doesn't define the padding content, then the following scheme **SHOULD** be used: the Padding bytes are initialized with a series of (unsigned, 1-byte) integer values. The first padding byte appended to the plaintext is numbered 1, with subsequent padding bytes making up a monotonically increasing sequence: 1, 2, 3, The length of the padding is not transmitted and is implicitly determined, since the length of the key is known.

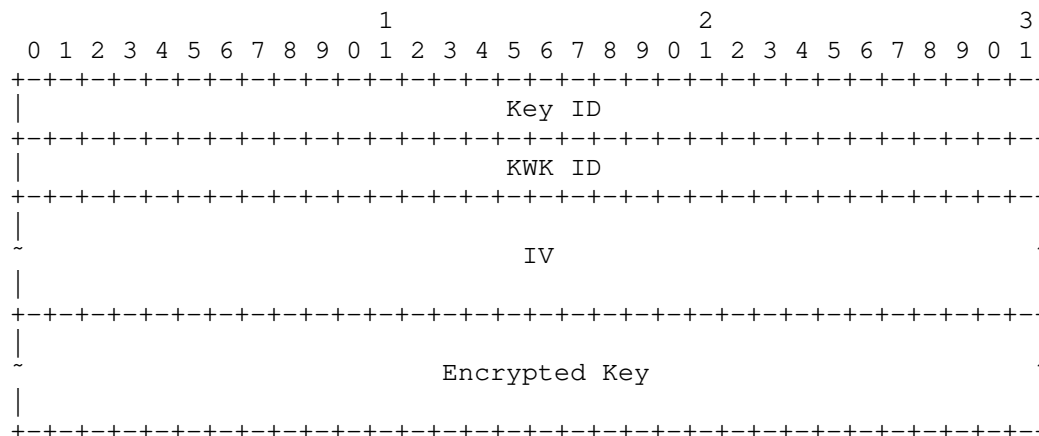


Figure 19: Wrapped Key Format

The Wrapped Key fields are defined as follows:

- o Key ID (4 octets) -- ID of the encrypted key. The value zero means that the encrypted key contains SA keys (in the form of keying material, see Section 3.4)), otherwise it contains some intermediate key.
- o KWK ID (4 octets) -- ID of the key that was used to encrypt key with specified Key ID. The value zero means that the default KWK was used to encrypt the key, otherwise some intermediate key was used.

- o IV (variable) -- Initialization Vector used for encryption. The size and the content of IV is defined by the employed encryption transform.
- o Encrypted Key (variable) -- The encrypted key bits. These bits may comprise either a single encrypted key or a result of encryption of a concatenation of keys (key material) for several algorithms.

4.5.2. Group Key Packet Substructure

Group Key Packet substructure contains SA key information. This key information is associated with some group SAs: either with Data-Security SAs or with group Rekey SA.

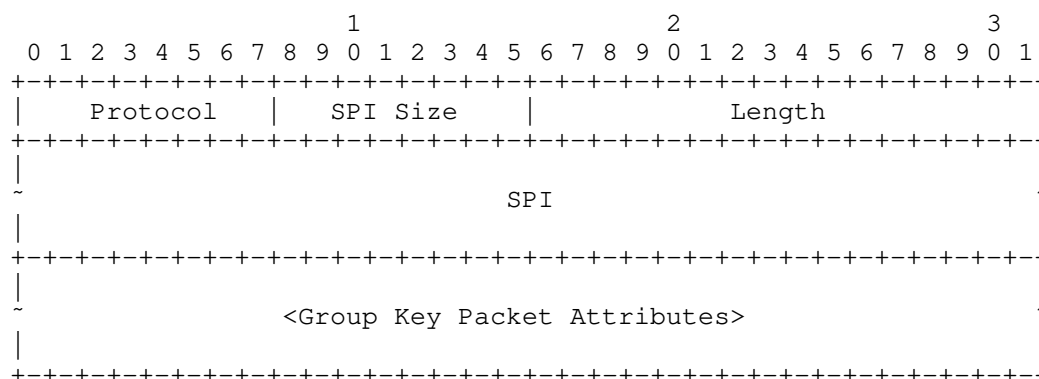


Figure 20: Group Key Packet Substructure Format

- o Protocol (1 octet) -- Identifies the security protocol for this key packet. The values are defined in the IKEv2 Security Protocol Identifiers in [IKEV2-IANA]. The valid values for this field are: <TBA> (GIKE_REKEY) for KEK Key packet and 2 (AH) or 3 (ESP) for TEK key packet.
- o SPI Size (1 octet) -- Size of Security Parameter Index (SPI) for the corresponding SA. SPI size depends on the security protocol. For GIKE_REKEY it is 16 octets, while for AH and ESP it is 4 octets.
- o Length (2 octets, unsigned integer) -- Length of this substructure including the header.
- o SPI (variable) -- Security Parameter Index for the corresponding SA. The size of this field is determined by the SPI Size field. In case of GIKE_REKEY the SPI must be the IKEv2 Header SPI pair

where the first 8 octets become the "Initiator's SPI" field in the G-IKEv2 rekey message IKEv2 HDR, and the second 8 octets become the "Responder's SPI" in the same HDR. When selecting SPI the GCKS MUST make sure that the sole first 8 octets (corresponding to "Initiator's SPI" field in the IKEv2 header) uniquely identify the Rekey SA.

- o Group Key Packet Attributes (variable length) -- Contains Key information for the corresponding SA.

This document creates a new IKEv2 IANA registry for the types of the Group Key Packet attributes which is initially filled as described in Section 8. In particular, the following attributes are initially added.

Group Key Packet Attributes	Value	Type	Multiple	Protocol
Reserved	0			
SA_KEY	1	V	N/Y* N	GIKE_REKEY, AH, ESP

(*) Multiple SA_KEY attributes may only appear for the GIKE_REKEY protocol in the GSA_REKEY exchange if the GCKS uses the Group Key Management method that allows excluding GMs from the group (like LKH).

The attributes must follow the format defined in the IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V).

4.5.2.1. SA_KEY Attribute

The SA_KEY attribute (1) contains a keying material for the corresponding SA. The content of the attribute is formatted according to Section 4.5.1 with a precondition that the Key ID field MUST always be zero. The size of the keying material MUST be equal to the total size of the keys needed to be taken from this keying material (see Section 3.4) for the corresponding SA.

If the Key Packet is for a Data-Security SA (AH or ESP protocols), then exactly one SA_KEY attribute MUST be present with both Key ID and KWK ID fields set to zero.

If the Key Packet is for a Rekey SA (GIKE_REKEY protocol), then in the GSA_AUTH, GSA_REGISTRATION and GSA_INBAND_REKEY exchanges exactly one SA_KEY attribute MUST be present. In the GSA_REKEY exchange at

least one SA_KEY attribute MUST be present, and more attributes MAY be present (depending on the key management method employed by the GCKS).

4.5.3. Member Key Packet Substructure

The Member Key Packet substructure contains keys and other parameters that are specific for a member of the group and are not associated with any particular group SA.

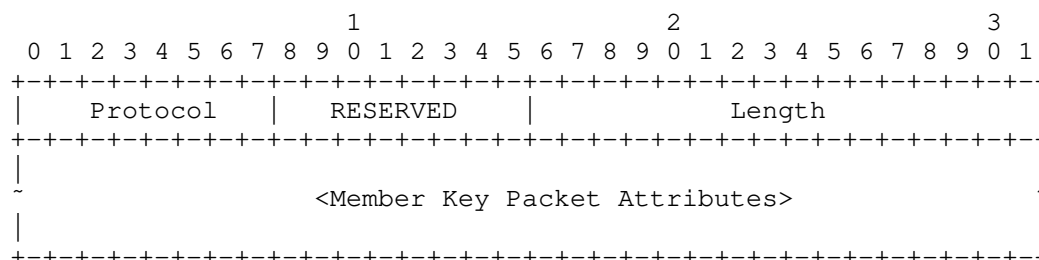


Figure 21: Member Key Packet Substructure Format

The Member Key Packet substructure fields are defined as follows:

- o Protocol (1 octet) -- MUST be zero. This value is reserved in Section 8 and is never used for any security protocol, so it is used here to indicate that this Key Packet is not associated with any particular SA.
- o RESERVED (octet) -- MUST be zero on transmission, MUST be ignored on receipt.
- o Length (2 octets, unsigned integer) -- Length of this substructure including the header.
- o Member Key Packet Attributes (variable length) -- Contains Key information and other parameters exclusively for a particular member of the group.

Member Key Packet substructure contains sensitive information for a single GM, for this reason it MUST NOT be sent in GSA_REKEY messages and MUST only be sent via unicast SA at the time the GM registers to the group (in either GSA_AUTH or GSA_REGISTRATION exchanges).

This document creates a new IKEv2 IANA registry for the types of the Member Key Packet attributes which is initially filled as described in Section 8. In particular, the following attributes are initially added.

Member Key Packet Attributes	Value	Type	Multiple
Reserved	0		
WRAP_KEY	1	V	Y
AUTH_KEY	2	V	N
GM_SID	3	V	Y

The attributes must follow the format defined in the IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V).

4.5.3.1. WRAP_KEY Attribute

The WRAP_KEY attribute (1) contains a key that is used to encrypt other keys. One or more these attributes are sent to GMs if the GCKS key management method relies on some key hierarchy (e.g. LKH). This attribute MUST NOT be used if inband rekeying (via the GSA_INBAND_REKEY exchange) is employed by the GCKS for the GM.

The content of the attribute has a format defined in Section 4.5.1 with a precondition that the Key ID field MUST NOT be zero. The algorithm associated with the key is from the Encryption Transform for the SA the WRAP_KEY attributes was sent in. The size of the key MUST be equal to the key size for this algorithm.

Multiple instances of the WRAP_KEY attributes MAY be present in the key packet.

4.5.3.2. AUTH_KEY Attribute

The AUTH_KEY attribute (2) contains the key that is used to authenticate the GSA_REKEY messages. The content of the attribute depends on the authentication method the GCKS specified in the Authentication Method transform in the GSA payload.

- o If digital signatures are used for the GSA_REKEY messages authentication then the content of the AUTH_KEY attribute is a public key used for digital signature authentication. The public key MUST be represented as DER-encoded ASN.1 object SubjectPublicKeyInfo, defined in section 4.1.2.7 of [RFC5280]. The signature algorithm that will use this key was specified in the Algorithm Identifier attribute of the Authentication Method transform. The key MUST be compatible with this algorithm. An RSA public key format is defined in [RFC8017], Section A.1. DSS public key format is defined in [RFC3279] Section 2.3.2. For ECDSA Public keys, use format described in [RFC5480] Section 2.

Other algorithms added to the IKEv2 Authentication Method registry are also expected to include a format of the SubjectPublicKeyInfo object included in the algorithm specification.

Multiple instances of the AUTH_KEY attributes MUST NOT be sent. This attribute MUST NOT appear in the rekey operations (in the GSA_REKEY or GSA_INBAND_REKEY exchanges).

4.5.3.3. GM_SID Attribute

The GM_SID attribute (3) is used to download one or more Sender-ID values for the exclusive use of a group member. One or more of this attributes MUST be sent by the GCKS if the GM informed the GCKS that it would be a sender (by inclusion the SENDER notification to the request) and at least one of the Data-Security SAs included in the GSA payload uses counter-based mode of encryption.

If the GMs has requested multiple SID values in the SENDER notification, then the GCKS SHOULD provide it with the requested number of SIDs by sending multiple instances of the GM_SID attribute. The GCKS MAY send fewer SIDs than requested by the GM (e.g. if it is running out of SIDs), but it MUST NOT send more than requested.

This attribute MUST NOT appear in the rekey operations (in the GSA_REKEY or GSA_INBAND_REKEY exchanges).

4.6. Delete Payload

Delete payload is used in G-IKEv2 when the GCKS wants to delete Data-Security and Rekey SAs. The interpretation of the Protocol field in the Delete payload is extended, so that zero protocol indicates deletion of whole Group SA (i.e. all Data-Security SAs and Rekey SA). See Section 2.4.3 for detail.

4.7. Notify Payload

G-IKEv2 uses the same Notify payload as specified in [RFC7296], section 3.10.

There are additional Notify Message types introduced by G-IKEv2 to communicate error conditions and status (see Section 8).

- o INVALID_GROUP_ID (45) -- error type notification that indicates that the group ID sent during the registration process is invalid. The Protocol ID and SPI Size fields in the Notify payload MUST be zero. There is no data associated with this notification and the content of the Notification Data field MUST be ignored on receipt.

- o AUTHORIZATION_FAILED (46) -- error type notification that is sent in the response to a GSA_AUTH or GSA_REGISTRATION message when authorization failed. The Protocol ID and SPI Size fields in the Notify payload MUST be zero. There is no data associated with this notification and the content of the Notification Data field MUST be ignored on receipt.
- o REGISTRATION_FAILED (<TBA>) -- error type notification that is sent by the GCKS when the GM registration request cannot be satisfied for the reasons not related to this particular GM, for example if the capacity of the group is exceeded. The Protocol ID and SPI Size fields in the Notify payload MUST be zero. There is no data associated with this notification and the content of the Notification Data field MUST be ignored on receipt.
- o SENDER (16429) -- status type notification that is sent in the GSA_AUTH or the GSA_REGISTRATION exchanges to indicate that the GM intends to be sender of data traffic. The data includes a count of how many SID values the GM desires. The count MUST be 4 octets long and contain the big endian representation of the number of requested SIDs. The Protocol ID and SPI Size fields in the Notify payload MUST be zero.
- o REKEY_IS_NEEDED (<TBA>) -- status type notification that is sent in the GSA_AUTH response message to indicate that the GM must perform an immediate rekey of IKE SA to make it secure against quantum computers and then start a registration request over. The Protocol ID and SPI Size fields in the Notify payload MUST be zero. There is no data associated with this notification and the content of the Notification Data field MUST be ignored on receipt.

4.7.1. USE_TRANSPORT_MODE Notification

This specification uses the USE_TRANSPORT_MODE notification defined in section 3.10.1 of [RFC7296] to specify the mode Data-Security SAs should be created in. The GCKS MUST include the USE_TRANSPORT_MODE notification in a message containing the GSA payload if Data-Security SAs are to be created in transport mode and MUST NOT include if they are to be created in tunnel mode.

Note, that it is not possible with this specification to create a group where some Data-Security SAs use transport mode and the others use tunnel mode. If such a configuration is needed two different groups must be defined.

4.8. Authentication Payload

G-IKEv2 uses the same Authentication payload as specified in [RFC7296], section 3.8, to authenticate the rekey message. However, if it is used in the GSA_REKEY messages the content of the payload is computed differently, as described in Section 2.4.1.1.

5. Using G-IKEv2 Attributes

G-IKEv2 defines a number of attributes, that are used to convey information from GCKS to GMs. There are some restrictions on where and when these attributes can appear in G-IKEv2 messages, which are defined when the attributes are introduced. For convenience these restrictions are summarized in Table 1 (for multicast rekey operations) and Table 2 (for inband rekey operations) below.

The following notation is used:

- S A single attribute of this type must be present
- M Multiple attributes of this type may be present
- [] Attribute is optional
- Attribute must not be present

Note, that the restrictions are defined per a substructure corresponding attributes are defined for and not per whole G-IKEv2 message.

Attributes	GSA_AUTH GSA_REGISTRATION	GSA_REKEY
GSA_KEY_LIFETIME	S	S
GSA_INITIAL_MESSAGE_ID	[S]	[S]
GSA_NEXT_SPI	[M]	[M]
GAP_ATD	[S]	[S]
GAP_DTD	[S]	[S]
GAP_SID_BITS	S*	-
SA_KEY	S	S/[M]**
WRAP_KEY	[M]**	[M]**
AUTH_KEY	S***	[S]****
GM_SID	S*/[M]*	-

Table 1: Using attributes in G-IKEv2 exchanges when multicast rekey is used

- * The GAP_SID_BITS attribute must be present if the GCKS policy includes at least one cipher in counter mode of operation and the GM included the SENDER notify into the registration request. Otherwise it must not be present. At least one GM_SID attribute must be present in the former case (and more may be present if the GM requested more SIDs) and no GM_SID attributes must be present in the latter case.
- ** The WRAP_KEY attributes may be present if the GCKS employs key management method that relies on key tree (like LKH).
- *** The AUTH_KEY attribute must be present if the GCKS employs authentication method other than NULL Authentication.
- *** The AUTH_KEY attribute may be present if the GCKS employs authentication method based on digital signatures and wants to change the public key for the following multicast rekey operations.

Attributes	GSA_AUTH GSA_REGISTRATION	GSA_INBAND_REKEY
GSA_KEY_LIFETIME	[S]	[S]
GSA_INITIAL_MESSAGE_ID	-	-
GSA_NEXT_SPI	-	-
GAP_ATD	[S]	[S]
GAP_DTD	[S]	[S]
GAP_SID_BITS	S*	-
SA_KEY	S	S
WRAP_KEY	-	-
AUTH_KEY	-	-
GM_SID	S*/[M]*	-

Table 2: Using attributes in G-IKEv2 exchanges when inband rekey is used

- * The GAP_SID_BITS attribute must be present if the GCKS policy includes at least one cipher in counter mode of operation and the GM included the SENDER notify into the registration request. Otherwise it must not be present. At least one GM_SID attribute must be present in the former case (and more may be present if the GM requested more SIDs) and no GM_SID attributes must be present in the latter case.

6. Interaction with other IKEv2 Protocol Extensions

A number of IKEv2 extensions is defined that can be used to extend protocol functionality. G-IKEv2 is compatible with most of them. In particular, EAP authentication defined in [RFC7296] can be used to establish registration IKE SA, as well as EAP-only authentication [RFC5998] and Secure Password authentication [RFC6467]. G-IKEv2 is compatible with and can use IKEv2 Redirect Mechanism [RFC5685] and IKEv2 Session Resumption [RFC5723]. G-IKEv2 is also compatible with Multiple Key Exchanges in IKEv2 framework, defined in [I-D.ietf-ipsecme-ikev2-multiple-ke].

The above list of compatible IKEv2 extensions is not exhaustive, however some IKEv2 extensions require special handling if used in G-IKEv2.

6.1. Mixing Preshared Keys in IKEv2 for Post-quantum Security

G-IKEv2 can take advantage of the protection provided by Postquantum Preshared Keys (PPK) for IKEv2 [RFC8784]. However, the use of PPK leaves the initial IKE SA susceptible to quantum computer (QC) attacks. While group SA keys are protected with the default KWK (GSK_w), which is derived from SK_d and thus cannot be broken even by attacker equipped with a QC, authentication of these keys relies on authentication of IKE SA messages, which is not secure against QC until the initial IKE SA is rekeyed. In addition, the other content of IKE SA messages may also be visible to an attacker with a QC. See Section 6 of [RFC8784] for details.

For these reasons the GCKS MUST NOT send GSA and KD payloads in the GSA_AUTH response message and MUST return a new notification REKEY_IS_NEEDED instead. Upon receiving this notification in the GSA_AUTH response the GM MUST perform an IKE SA rekey and then initiate a new GSA_REGISTRATION request for the same group. Below are possible scenarios involving using PPK.

The GM starts the IKE_SA_INIT exchange requesting using PPK, and the GCKS responds with agreement to do it, or aborts according to its "mandatory_or_not" flag:

Initiator (Member)	Responder (GCKS)
<hr/>	
HDR, SAi1, KEi, Ni, N(USE_PPK)	-->
	<-- DR, SAr1, KEr, Nr, [CERTREQ], N(USE_PPK)

Figure 22: IKE_SA_INIT Exchange requesting using PPK

The GM then starts the GSA_AUTH exchange with the PPK_ID; if using PPK is not mandatory for the GM, the NO_PPK_AUTH notification is included in the request:

Initiator (Member)	Responder (GCKS)
<hr/>	
HDR, SK{IDi, AUTH, IDg, N(PPK_IDENTITY), N(NO_PPK_AUTH)}	-->

Figure 23: GSA_AUTH Request using PPK

If the GCKS has no such PPK and using PPK is not mandatory for it and the NO_PPK_AUTH is included, then the GCKS continues without PPK; in this case no rekey is needed:

```

Initiator (Member)                Responder (GCKS)
-----
                                <-- HDR, SK{IDr, AUTH, GSA, KD}

```

Figure 24: GSA_AUTH Response using no PPK

If the GCKS has no such PPK and either the NO_PPK_AUTH is missing or using PPK is mandatory for the GCKS, the GCKS aborts the exchange:

```

Initiator (Member)                Responder (GCKS)
-----
                                <-- HDR, SK{N(AUTHENTICATION_FAILED)}

```

Figure 25: GSA_AUTH Error Response

Assuming the GCKS has the proper PPK it continues with a request to the GM to immediately perform a rekey by sending the REKEY_IS_NEEDED notification:

```

Initiator (Member)                Responder (GCKS)
-----
                                <-- HDR, SK{IDr, AUTH, N(PPK_IDENTITY),
                                N(REKEY_IS_NEEDED) }

```

Figure 26: GSA_AUTH Response using PPK

The GM initiates the CREATE_CHILD_SA exchange to rekey the initial IKE SA and then makes a new registration request for the same group over the new IKE SA:

```

Initiator (Member)                Responder (GCKS)
-----
HDR, SK{SA, Ni, KEi}  -->
                                <-- HDR, SK{SA, Nr, KEr}
HDR, SK{IDg}  --->
                                <-- HDR, SK{GSA, KD}

```

Figure 27: Rekeying IKE SA followed by GSA_REGISTRATION Exchange

Note, that [I-D.smyslov-ipsecme-ikev2-gr-alt] MAY be used to make the initial IKE SA secure against QC.

7. Security Considerations

7.1. GSA Registration and Secure Channel

G-IKEv2 registration exchange uses IKEv2 IKE_SA_INIT protocols, inheriting all the security considerations documented in [RFC7296] section 5 Security Considerations, including authentication, confidentiality, protection against man-in-the-middle, protection against replay/reflection attacks, and denial of service protection. The GSA_AUTH and GSA_REGISTRATION exchanges also take advantage of those protections. In addition, G-IKEv2 brings in the capability to authorize a particular group member regardless of whether they have the IKEv2 credentials.

7.2. GSA Maintenance Channel

The GSA maintenance channel is cryptographically and integrity protected using the cryptographic algorithm and key negotiated in the GSA member registration exchange.

7.2.1. Authentication/Authorization

The authentication key is distributed during the GM registration, and the receiver of the rekey message uses that key to verify the message came from the authorized GCKS. An implicit authentication can also be used, in which case the ability of the GM to decrypt and to verify ICV of the received message proved that a sender of the message is a member of the group. However, implicit authentication doesn't provide source origin authentication, so the GM cannot be sure that the message came from the GCKS. For this reason using implicit authentication is NOT RECOMMENDED unless in a small group of trusted parties.

7.2.2. Confidentiality

Confidentiality is provided by distributing a confidentiality key as part of the GSA member registration exchange.

7.2.3. Man-in-the-Middle Attack Protection

GSA maintenance channel is integrity protected by using a digital signature.

7.2.4. Replay/Reflection Attack Protection

The GSA_REKEY message includes a monotonically increasing sequence number to protect against replay and reflection attacks. A group member will recognize a replayed message by comparing the Message ID

number to that of the last received rekey message, any rekey message containing a Message ID number less than or equal to the last received value MUST be discarded. Implementations should keep a record of recently received GSA rekey messages for this comparison.

8. IANA Considerations

8.1. New Registries

A new set of registries is created for G-IKEv2 on IKEv2 parameters page [IKEV2-IANA]. The terms Reserved, Expert Review and Private Use are to be applied as defined in [RFC8126].

This document creates a new IANA registry "Transform Type <TBA> - Group Key Management Methods". The initial values of the new registry are:

Value	Group Key Management Method
Reserved	0
Wrapped Key Download	1
Unassigned	2-1023
Private Use	1024-65535

Changes and additions to the unassigned range of this registry are by the Expert Review Policy [RFC8126].

This document creates a new IANA registry "GSA Attributes". The initial values of the new registry are:

GSA Attributes	Value	Type	Multiple	Protocol
Reserved	0			
GSA_KEY_LIFETIME	1	V	N	GIKE_REKEY, AH, ESP
GSA_INITIAL_MESSAGE_ID	2	V	N	GIKE_REKEY
GSA_NEXT_SPI	3	V	Y	GIKE_REKEY, AH, ESP
Unassigned	5-16383			
Private Use	16384-32767			

Changes and additions to the unassigned range of this registry are by the Expert Review Policy [RFC8126].

This document creates a new IANA registry "GAP Attributes". The initial values of the new registry are:

GAP Attributes	Value	Type	Multiple
Reserved	0		
GAP_ATD	1	B	N
GAP_DTD	2	B	N
GAP_SID_BITS	3	B	N
Unassigned	4-16383		
Private Use	16384-32767		

Changes and additions to the unassigned range of this registry are by the Expert Review Policy [RFC8126].

This document creates a new IANA registry "Group Key Packet Attributes". The initial values of the new registry are:

Group Key Packet Attributes	Value	Type	Multiple	Protocol
Reserved	0			
SA_KEY	1	V	Y N	GIKE_REKEY, AH, ESP
Unassigned	2-16383			
Private Use	16384-32767			

Changes and additions to the unassigned range of this registry are by the Expert Review Policy [RFC8126].

This document creates a new IANA registry "Member Key Packet Attributes". The initial values of the new registry are:

Member Key Packet Attributes	Value	Type	Multiple
Reserved	0		
WRAP_KEY	1	V	Y
AUTH_KEY	2	V	N
GM_SID	3	V	Y
Unassigned	4-16383		
Private Use	16384-32767		

Changes and additions to the unassigned range of this registry are by the Expert Review Policy [RFC8126].

8.2. Changes in the Existing IKEv2 Registries

This document defines new Exchange Types in the "IKEv2 Exchange Types" registry:

Value	Exchange Type
39	GSA_AUTH
40	GSA_REGISTRATION
41	GSA_REKEY
<TBA>	GSA_INBAND_REKEY

This document defines new Payload Types in the "IKEv2 Payload Types" registry:

Value	Next Payload Type	Notation
50	Group Identification	IDg
51	Group Security Association	GSA
52	Key Download	KD

This document makes the following changes to the "Transform Type Values" registry:

- o Defines two new transform types -- "Authentication Method (AUTH)" and "Group Key Management Method (GKM)";
- o Renames existing transform type "Extended Sequence Numbers (ESN)" to "Replay Protection (RP)";
- o Changes the "Used In" column for the existing allocations as follows;

Type	Description	Used In
1	Encryption Algorithm (ENCR)	IKE, GIKE_REKEY and ESP
2	Pseudo-random Function (PRF)	IKE, GIKE_REKEY
3	Integrity Algorithm (INTEG)	IKE, GIKE_REKEY, AH, optional in ESP
4	Diffie-Hellman Group (D-H)	IKE, optional in AH, ESP
5	Replay Protection (RP)	AH and ESP
<TBA>	Authentication Method (AUTH)	GIKE_REKEY
<TBA>	Group Key Management Method (GKM)	GIKE_REKEY

This document defines a new Attribute Type in the "IKEv2 Transform Attribute Types" registry:

Value	Attribute Type	Format
<TBA>	Algorithm Identifier	TLV

This document renames the "Transform Type 5 - Extended Sequence Numbers Transform IDs" registry to "Transform Type 5 - Replay

Protection Transform IDs" and also adds a new value into this registry:

Number	Name

<TBA>	Not Used

This document defines new Notify Message Types in the "Notify Message Types - Error Types" registry:

Value	Notify Messages - Error Types

45	INVALID_GROUP_ID
46	AUTHORIZATION_FAILED
<TBA>	REGISTRATION_FAILED

This document defines new Notify Message Types in the "Notify Message Types - Status Types" registry:

Value	Notify Messages - Status Types

16429	SENDER

The Notify type with the value 16429 was allocated earlier in the development of G-IKEv2 document with the name SENDER_REQUEST_ID. This specification changes its name to SENDER.

This document defines a new Security Protocol Identifier in the "IKEv2 Security Protocol Identifiers" registry:

Protocol ID	Protocol

<TBA>	GIKE_REKEY

9. Acknowledgements

The authors thank Lakshminath Dondeti and Jing Xiang for first exploring the use of IKEv2 for group key management and providing the basis behind the protocol. Mike Sullenberger and Amjad Inamdar were instrumental in helping resolve many issues in several versions of the document.

The authors are grateful to Tero Kivinen for his careful review and valuable proposals how to improve the document.

10. Contributors

The following individuals made substantial contributions to early versions of this memo.

Sheela Rowles
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-527-7677
Email: sheela@cisco.com

Aldous Yeung
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-853-2032
Email: cyyeung@cisco.com

Paulina Tran
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-526-8902
Email: ptran@cisco.com

Yoav Nir
Dell EMC
9 Andrei Sakharov St
Haifa 3190500
Israel

Email: ynir.ietf@gmail.com

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [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, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, DOI 10.17487/RFC6054, November 2010, <<https://www.rfc-editor.org/info/rfc6054>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <<https://www.rfc-editor.org/info/rfc7427>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

11.2. Informative References

- [I-D.ietf-ipsecme-ikev2-intermediate]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-10 (work in progress), March 2022.
- [I-D.ietf-ipsecme-ikev2-multiple-ke]
Tjhai, C., Tomlinson, M., Bartlett, G., Fluhrer, S., Geest, D. V., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in IKEv2", draft-ietf-ipsecme-ikev2-multiple-ke-05 (work in progress), March 2022.
- [I-D.smyslov-ipsecme-ikev2-qr-alt]
Smyslov, V., "Alternative Approach for Mixing Preshared Keys in IKEv2 for Post-quantum Security", draft-smyslov-ipsecme-ikev2-qr-alt-04 (work in progress), August 2021.
- [IKEV2-IANA]
IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <<http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7>>.
- [NNL]
Naor, D., Noal, M., and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", Advances in Cryptology, Crypto '01, Springer-Verlag LNCS 2139, 2001, pp. 41-62, 2001, <<http://www.wisdom.weizmann.ac.il/~naor/PAPERS/2nl.pdf>>.
- [OFT]
McGrew, D. and A. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", Manuscript, submitted to IEEE Transactions on Software Engineering, 1998, <<https://pdfs.semanticscholar.org/d24c/7b41f7bcc2b6690e1b4d80eaf8c3e1cc5ee5.pdf>>.
- [RFC2409]
Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC2627]
Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, DOI 10.17487/RFC2627, June 1999, <<https://www.rfc-editor.org/info/rfc2627>>.
- [RFC3279]
Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <<https://www.rfc-editor.org/info/rfc3279>>.

- [RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <<https://www.rfc-editor.org/info/rfc3686>>.
- [RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, DOI 10.17487/RFC3740, March 2004, <<https://www.rfc-editor.org/info/rfc3740>>.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <<https://www.rfc-editor.org/info/rfc3948>>.
- [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, DOI 10.17487/RFC4046, April 2005, <<https://www.rfc-editor.org/info/rfc4046>>.
- [RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <<https://www.rfc-editor.org/info/rfc4106>>.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <<https://www.rfc-editor.org/info/rfc4309>>.
- [RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <<https://www.rfc-editor.org/info/rfc4543>>.
- [RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <<https://www.rfc-editor.org/info/rfc5374>>.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <<https://www.rfc-editor.org/info/rfc5480>>.
- [RFC5685] Devarapalli, V. and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5685, DOI 10.17487/RFC5685, November 2009, <<https://www.rfc-editor.org/info/rfc5685>>.

- [RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, DOI 10.17487/RFC5723, January 2010, <<https://www.rfc-editor.org/info/rfc5723>>.
- [RFC5998] Eronen, P., Tschofenig, H., and Y. Sheffer, "An Extension for EAP-Only Authentication in IKEv2", RFC 5998, DOI 10.17487/RFC5998, September 2010, <<https://www.rfc-editor.org/info/rfc5998>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, DOI 10.17487/RFC6467, December 2011, <<https://www.rfc-editor.org/info/rfc6467>>.
- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.
- [RFC7634] Nir, Y., "ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec", RFC 7634, DOI 10.17487/RFC7634, August 2015, <<https://www.rfc-editor.org/info/rfc7634>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.
- [RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <<https://www.rfc-editor.org/info/rfc8784>>.

Appendix A. Use of LKH in G-IKEv2

Section 5.4 of [RFC2627] describes the LKH architecture, and how a GCKS uses LKH to exclude group members. This section clarifies how the LKH architecture is used with G-IKEv2.

A.1. Notation

In this section we will use the notation $X\{Y\}$ where a key with ID Y is encrypted with the key with ID X . The notation $GSK_w\{Y\}$ means that the default wrap key GSK_w (with zero KWK ID) is used to encrypt key Y , and the notation $X\{K_sa\}$ means key X is used to encrypt the SA key K_sa (which always has zero Key ID). Note, that $GSK_w\{K_sa\}$ means that the SA key is encrypted with the default wrap key, in which case both KWK ID and Key ID are zero. For simplicity we will assume that

The content of the KD payload will be shown as a sequence of Key Packets. The Group Key Packet substructure will be denoted as $GP(SAn())$, when n is an SPI for the SA, and the Member Key Packet substructure will be denoted as $MP()$. The content of the Key Packets is shown as SA_KEY and $WRAP_KEY$ attributes with the notation described above. For simplicity the type of the attribute will not be shown, because it is implicitly defined by the type of Key Packet.

Here is the example of KD payload.

$$KD(GP1(X\{K_sa\}), MP(Y\{X\}, Z\{Y\}, GSK_w\{Z\}))$$

For simplicity any other attributes in the KD payload are omitted.

We will also use the notation $X \rightarrow Y \rightarrow Z$ to describe the Key Path. In this case key Y is needed to decrypt key X and key Z is needed to decrypt key Y . In the example above the keys had the following relation: $K_sa \rightarrow X \rightarrow Y \rightarrow Z \rightarrow GSK_w$.

A.2. Group Creation

When a GCKS forms a group, it creates a key tree as shown in the figure below. The key tree contains logical keys (which are represented as the values of their Key IDs in the figure) and a private key shared with only a single GM (the GMs are represented as letters followed by the corresponding key ID in parentheses in the figure). The root of the tree contains the multicast Rekey SA key (which is represented as $SAn(K_san)$). The figure below assumes that the Key IDs are assigned sequentially; this is not a requirement and only used for illustrative purposes. The GCKS may create a complete tree as shown, or a partial tree which is created on demand as members join the group.

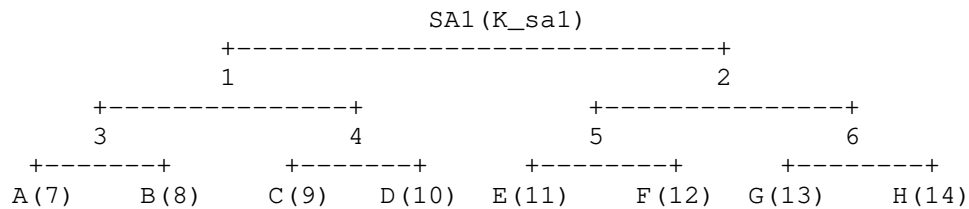


Figure 28: Initial LKH tree

When GM A joins the group, the GCKS provides it with the keys in the KD payload of the GSA_AUTH or GSA_REGISTRATION exchange. Given the tree shown in figure above, the KD payload will be:

$KD(GP(SA1)(1\{K_{sa1}\}), MP(3\{1\}, 7\{3\}, GSK_w\{7\}))$

KD Payload for the Group Member A

From these attributes the GM A will construct the Key Path $K_{sa1} \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow GSK_w$ and since it ends up with GSK_w , it will use all the WRAP_KEY attributes present in the path as its Working Key Path: $1 \rightarrow 3 \rightarrow 7$.

Similarly, when other GMs will be joining the group they will be provided with the corresponding keys, so after all the GMs will have the following Working Key Paths:

A: $1 \rightarrow 3 \rightarrow 7$	B: $1 \rightarrow 3 \rightarrow 8$	C: $1 \rightarrow 4 \rightarrow 9$,	D: $1 \rightarrow 4 \rightarrow 10$
E: $2 \rightarrow 5 \rightarrow 11$	F: $2 \rightarrow 5 \rightarrow 12$	G: $2 \rightarrow 6 \rightarrow 13$	H: $2 \rightarrow 6 \rightarrow 14$

A.3. Simple Group SA Rekey

If the GCKS performs a simple SA rekey without changing group membership, it will only send Group Key Packet in the KD payload with a new SA key encrypted with the default KWK.

$KD(GP(SA2)(GSK_w\{K_{sa2}\}))$

KD Payload for the Simple Group SA Rekey

All the GMs will be able to decrypt it and no changes in their Working Key Paths will happen.

A.4. Group Member Exclusion

If the GKCS has reason to believe that a GM should be excluded, then it can do so by sending a GSA_REKEY message that includes a set of

GM_KEY attributes which would allow all GMs except for the excluded one to get a new SA key.

In the example below the GCKS excludes GM F. For this purpose it changes the key tree as follows, replacing the key 2 with the key 15 and the key 5 with the key 16. It also generates a new SA key for a new SA3.

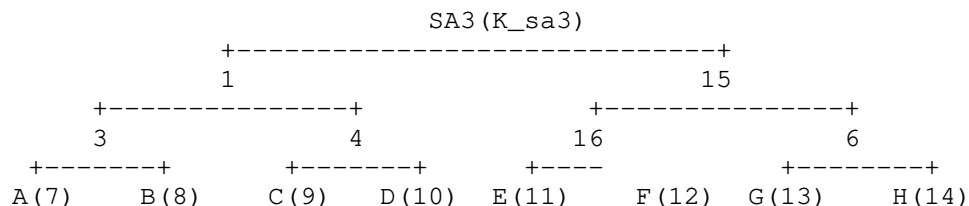


Figure 29: LKH tree after F has been excluded

Then it sends the following KD payload for the new Rekey SA3:

KD(GP(SA3)(1{K_sa3},15{K_sa3}),MP(6{15},16{15},11{16}))

KD Payload for the Group Member F

While processing this KD payload:

- o GMs A, B, C and D will be able to decrypt the SA_KEY attribute 1{K_sa3} by using the "1" key from their key path. Since no new GM_KEY attributes are in the new Key Path, they won't update their Working Key Paths.
- o GMs G and H will construct new Key Path 15->6 and will be able to decrypt the intermediate key 15 using the key 6 from their Working Key Paths. So, they will update their Working Key Paths replacing their beginnings up to the key 6 with the new Key Path (thus replacing the key 2 with the key 15).
- o GM E will construct new Key Path 16->15->11 and will be able to decrypt the intermediate key 16 using the key 11 from its Working Key Path. So, it will update its Working Key Path replacing its beginnings up to the key 11 with the new Key Path (thus replacing the key 2 with the key 15 and the key 5 with the key 16).
- o GM F won't be able to construct any Key Path leading to any key he possesses, so it will be unable to decrypt the new SA key for the SA3 and thus it will be excluded from the group once the SA3 is used.

Finally, the GMs will have the following Working Key Paths:

A: 1->3->7	B: 1->3->8	C: 1->4->9,	D: 1->4->10
E: 15->16->11	F: excluded	G: 15->6->13	H: 15->6->14

Authors' Addresses

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
Russian Federation

Phone: +7 495 276 0211
Email: svan@elvis.ru

Brian Weis
Independent
USA

Email: bew.stds@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 12, 2022

C. Hopps
LabN Consulting, L.L.C.
November 8, 2021

IP-TFS: Aggregation and Fragmentation Mode for ESP and its Use for IP
Traffic Flow Security
draft-ietf-ipsecme-iptfs-12

Abstract

This document describes a mechanism for aggregation and fragmentation of IP packets when they are being encapsulated in ESP payload. This new payload type can be used for various purposes such as decreasing encapsulation overhead for small IP packets; however, the focus in this document is to enhance IPsec traffic flow security (IP-TFS) by adding Traffic Flow Confidentiality (TFC) to encrypted IP encapsulated traffic. TFC is provided by obscuring the size and frequency of IP traffic using a fixed-sized, constant-send-rate IPsec tunnel. The solution allows for congestion control as well as non-constant send-rate usage.

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 <https://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 May 12, 2022.

Copyright Notice

Copyright (c) 2021 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 (<https://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. Terminology & Concepts	4
2. The AGGFRAG Tunnel	4
2.1. Tunnel Content	4
2.2. Payload Content	5
2.2.1. Data Blocks	6
2.2.2. End Padding	6
2.2.3. Fragmentation, Sequence Numbers and All-Pad Payloads	6
2.2.4. Empty Payload	8
2.2.5. IP Header Value Mapping	8
2.2.6. IP Time-To-Live (TTL) and Tunnel errors	9
2.2.7. Effective MTU of the Tunnel	9
2.3. Exclusive SA Use	10
2.4. Modes of Operation	10
2.4.1. Non-Congestion Controlled Mode	10
2.4.2. Congestion Controlled Mode	10
2.5. Summary of Receiver Processing	12
3. Congestion Information	12
3.1. ECN Support	14
4. Configuration of AGGFRAG Tunnels for IP-TFS	14
4.1. Bandwidth	14
4.2. Fixed Packet Size	14
4.3. Congestion Control	14
5. IKEv2	15
5.1. USE_AGGFRAG Notification Message	15
6. Packet and Data Formats	15
6.1. AGGFRAG_PAYLOAD Payload	15
6.1.1. Non-Congestion Control AGGFRAG_PAYLOAD Payload Format	16
6.1.2. Congestion Control AGGFRAG_PAYLOAD Payload Format	17
6.1.3. Data Blocks	19
6.1.4. IKEv2 USE_AGGFRAG Notification Message	20
7. IANA Considerations	21
7.1. AGGFRAG_PAYLOAD Sub-Type Registry	21
7.2. USE_AGGFRAG Notify Message Status Type	21
8. Security Considerations	22
9. References	22
9.1. Normative References	22
9.2. Informative References	22
Appendix A. Example Of An Encapsulated IP Packet Flow	24

Appendix B. A Send and Loss Event Rate Calculation	25
Appendix C. Comparisons of IP-TFS	26
C.1. Comparing Overhead	26
C.1.1. IP-TFS Overhead	26
C.1.2. ESP with Padding Overhead	26
C.2. Overhead Comparison	27
C.3. Comparing Available Bandwidth	28
C.3.1. Ethernet	28
Appendix D. Acknowledgements	30
Appendix E. Contributors	30
Author's Address	31

1. Introduction

Traffic Analysis ([RFC4301], [AppCrypt]) is the act of extracting information about data being sent through a network. While directly obscuring the data with encryption [RFC4303], the traffic pattern itself exposes information due to variations in its shape and timing ([RFC8546], [AppCrypt]). Hiding the size and frequency of traffic is referred to as Traffic Flow Confidentiality (TFC) per [RFC4303].

[RFC4303] provides for TFC by allowing padding to be added to encrypted IP packets and allowing for transmission of all-pad packets (indicated using protocol 59). This method has the major limitation that it can significantly under-utilize the available bandwidth.

This document defines an aggregation and fragmentation (AGGFRAG) mode for ESP, and its use for IP Traffic Flow Security (IP-TFS). This solution provides for full TFC without the aforementioned bandwidth limitation. This is accomplished by using a constant-send-rate IPsec [RFC4303] tunnel with fixed-sized encapsulating packets; however, these fixed-sized packets can contain partial, whole or multiple IP packets to maximize the bandwidth of the tunnel. A non-constant send-rate is allowed, but the confidentiality properties of its use are outside the scope of this document.

For a comparison of the overhead of IP-TFS with the RFC4303 prescribed TFC solution see Appendix C.

Additionally, IP-TFS provides for operating fairly within congested networks [RFC2914]. This is important for when the IP-TFS user is not in full control of the domain through which the IP-TFS tunnel path flows.

The mechanisms, such as the AGGFRAG mode, defined in this document are generic with the intent of allowing for non-TFS uses, but such uses are outside the scope of this document.

1.1. Terminology & Concepts

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document assumes familiarity with IP security concepts including TFC as described in [RFC4301].

2. The AGGFRAG Tunnel

As mentioned in Section 1, AGGFRAG mode utilizes an IPsec [RFC4303] tunnel as its transport. For the purpose of IP-TFS, fixed-sized encapsulating packets are sent at a constant rate on the AGGFRAG tunnel.

The primary input to the tunnel algorithm is the requested bandwidth to be used by the tunnel. Two values are then required to provide for this bandwidth use, the fixed size of the encapsulating packets, and rate at which to send them.

The fixed packet size MAY either be specified manually or be determined through other methods such as the Packetization Layer MTU Discovery (PLMTUD) ([RFC4821], [RFC8899]) or Path MTU discovery (PMTUD) ([RFC1191], [RFC8201]). PMTUD is known to have issues so PLMTUD is considered the more robust option. For PLMTUD, congestion control payloads can be used as in-band probes (see Section 6.1.2 and [RFC8899]).

Given the encapsulating packet size and the requested bandwidth to be used, the corresponding packet send rate can be calculated. The packet send rate is the requested bandwidth to be used divided by the size of the encapsulating packet.

The egress (receiving) side of the AGGFRAG tunnel MUST allow for and expect the ingress (sending) side of the AGGFRAG tunnel to vary the size and rate of sent encapsulating packets, unless constrained by other policy.

2.1. Tunnel Content

As previously mentioned, one issue with the TFC padding solution in [RFC4303] is the large amount of wasted bandwidth as only one IP packet can be sent per encapsulating packet. In order to maximize bandwidth, IP-TFS breaks this one-to-one association by introducing an AGGFRAG mode for ESP.

AGGFRAG mode aggregates as well as fragments the inner IP traffic flow into encapsulating IPsec tunnel packets. For IP-TFS, the IPsec encapsulating tunnel packets are a fixed size. Padding is only added to the the tunnel packets if there is no data available to be sent at the time of tunnel packet transmission, or if fragmentation has been disabled by the receiver.

This is accomplished using a new Encapsulating Security Payload (ESP, [RFC4303]) Next Header field value AGGFRAG_PAYLOAD (Section 6.1).

Other non-IP-TFS uses of this AGGFRAG mode have been suggested, such as increased performance through packet aggregation, as well as handling MTU issues using fragmentation. These uses are not defined here, but are also not restricted by this document.

2.2. Payload Content

The AGGFRAG_PAYLOAD payload content defined in this document is comprised of a 4 or 24 octet header followed by either a partial datablock, a full datablock, or multiple partial or full datablocks. The following diagram illustrates this payload within the ESP packet. See Section 6.1 for the exact formats of the AGGFRAG_PAYLOAD payload.

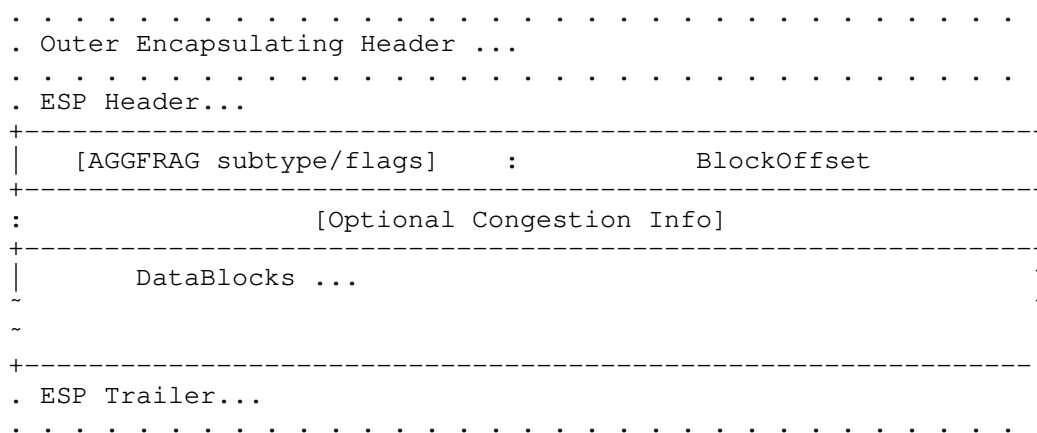


Figure 1: Layout of an AGGFRAG mode IPsec Packet

The "BlockOffset" value is either zero or some offset into or past the end of the "DataBlocks" data.

If the "BlockOffset" value is zero it means that the "DataBlocks" data begins with a new data block.

Conversely, if the "BlockOffset" value is non-zero it points to the start of the new data block, and the initial "DataBlocks" data belongs to the data block that is still being re-assembled.

If the "BlockOffset" points past the end of the "DataBlocks" data then the next data block occurs in a subsequent encapsulating packet.

Having the "BlockOffset" always point at the next available data block allows for recovering the next inner packet in the presence of outer encapsulating packet loss.

An example AGGFRAG mode packet flow can be found in Appendix A.

2.2.1. Data Blocks

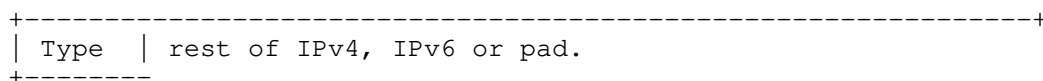


Figure 2: Layout of a DataBlock

A data block is defined by a 4-bit type code followed by the data block data. The type values have been carefully chosen to coincide with the IPv4/IPv6 version field values so that no per-data block type overhead is required to encapsulate an IP packet. Likewise, the length of the data block is extracted from the encapsulated IPv4's "Total Length" or IPv6's "Payload Length" fields.

2.2.2. End Padding

Since a data block's type is identified in its first 4-bits, the only time padding is required is when there is no data to encapsulate. For this end padding a "Pad Data Block" is used.

2.2.3. Fragmentation, Sequence Numbers and All-Pad Payloads

In order for a receiver to reassemble fragmented inner-packets, the sender MUST send the inner-packet fragments back-to-back in the logical outer packet stream (i.e., using consecutive ESP sequence numbers). However, the sender is allowed to insert "all-pad" payloads (i.e., payloads with a "BlockOffset" of zero and a single pad "DataBlock") in between the packets carrying the inner-packet fragment payloads. This interleaving of all-pad payloads allows the sender to always send a tunnel packet, regardless of the encapsulation computational requirements.

When a receiver is reassembling an inner-packet, and it receives an "all-pad" payload, it increments the expected sequence number that the next inner-packet fragment is expected to arrive in.

Given the above, the receiver will need to handle out-of-order arrival of outer ESP packets prior to reassembly processing. ESP already provides for optionally detecting replay attacks. Detecting replay attacks normally utilizes a window method. A similar sequence number based sliding window can be used to correct re-ordering of the outer packet stream. Receiving a larger (newer) sequence number packet advances the window, and received older ESP packets whose sequence numbers the window has passed by are dropped. A good choice for the size of this window depends on the amount of misordering the user may normally experience.

As the amount of misordering that may be present is hard to predict, the window size SHOULD be configurable by the user. Implementations MAY also dynamically adjust the reordering window based on actual misordering seen in arriving packets.

Please note when IP-TFS sends a continuous stream of packets, there is no requirement for an explicit lost packet timer; however, using a lost packet timer is RECOMMENDED. If an implementation does not use a lost packet timer and only considers an outer packet lost when the reorder window moves by it, the inner traffic can be delayed by up to the reorder window size times the per packet send rate. This amount of delay could be significant for slower send rates or when larger reorder window sizes are in use. As the lost packet timer affects delay of inner packet delivery, one could choose to set it proportionate to the tunnel rate.

While ESP guarantees an increasing sequence number with subsequently sent packets, it does not actually require the sequence numbers to be generated with no gaps (e.g., sending only even numbered sequence numbers would be allowed as long as they are always increasing). Gaps in the sequence numbers will not work for this document so the sequence number stream MUST increase monotonically by 1 for each subsequent packet.

When using the AGGFRAG_PAYLOAD in conjunction with replay detection, the window size for both MAY be reduced to the smaller of the two window sizes. This is because packets outside of the smaller window but inside the larger would still be dropped by the mechanism with the smaller window size. However, there is also no requirement to make these values the same. Indeed, in some cases, such as slow tunnels where a very small or zero reorder window size is appropriate, the user may still want a large replay detection window to log replayed packets. Additionally, large replay windows can be

implemented with very little overhead compared to large reorder windows.

Finally, as sequence numbers are reset when switching SAs (e.g., when re-keying a child SA), senders MUST NOT send initial fragments of an inner packet using one SA and subsequent fragments in a different SA.

2.2.3.1. Optional Extra Padding

When the tunnel bandwidth is not being fully utilized, a sender MAY pad-out the current encapsulating packet in order to deliver an inner packet un-fragmented in the following outer packet. The benefit would be to avoid inner-packet fragmentation in the presence of a bursty offered load (non-bursty traffic will naturally not fragment). Senders MAY also choose to allow for a minimum fragment size to be configured (e.g., as a percentage of the AGGFRAG_PAYLOAD payload size) to avoid fragmentation at the cost of tunnel bandwidth. The cost with these methods is complexity and added delay of inner traffic. The main advantage to avoiding fragmentation is to minimize inner packet loss in the presence of outer packet loss. When this is worthwhile (e.g., how much loss and what type of loss is required, given different inner traffic shapes and utilization, for this to make sense), and what values to use for the allowable/added delay may be worth researching, but is outside the scope of this document.

While use of padding to avoid fragmentation does not impact interoperability, used inappropriately it can reduce the effective throughput of a tunnel. Senders implementing either of the above approaches will need to take care to not reduce the effective capacity, and overall utility, of the tunnel through the overuse of padding.

2.2.4. Empty Payload

To support reporting of congestion control information (described later) using a non-AGGFRAG_PAYLOAD enabled SA, it is allowed to send an AGGFRAG_PAYLOAD payload with no data blocks (i.e., the ESP payload length is equal to the AGGFRAG_PAYLOAD header length). This special payload is called an empty payload.

Currently this situation is only applicable in non-IKEv2 use cases.

2.2.5. IP Header Value Mapping

[RFC4301] provides some direction on when and how to map various values from an inner IP header to the outer encapsulating header, namely the Don't-Fragment (DF) bit ([RFC0791] and [RFC8200]), the Differentiated Services (DS) field [RFC2474] and the Explicit

Congestion Notification (ECN) field [RFC3168]. Unlike [RFC4301], AGGFRAG mode may and often will be encapsulating more than one IP packet per ESP packet. To deal with this, these mappings are restricted further.

2.2.5.1. DF bit

AGGFRAG mode never maps the inner DF bit as it is unrelated to the AGGFRAG tunnel functionality; AGGFRAG mode never needs to IP fragment the inner packets and the inner packets will not affect the fragmentation of the outer encapsulation packets.

2.2.5.2. ECN value

The ECN value need not be mapped as any congestion related to the constant-send-rate IP-TFS tunnel is unrelated (by design) to the inner traffic flow. The sender MAY still set the ECN value of inner packets based on the normal ECN specification [RFC3168].

2.2.5.3. DS field

By default the DS field SHOULD NOT be copied, although a sender MAY choose to allow for configuration to override this behavior. A sender SHOULD also allow the DS value to be set by configuration.

2.2.6. IP Time-To-Live (TTL) and Tunnel errors

[RFC4301] specifies how to modify the inner packet TTL [RFC0791].

Any errors (e.g., ICMP errors arriving back at the tunnel ingress due to tunnel traffic) are handled the same as with non-AGGFRAG IPsec tunnels.

2.2.7. Effective MTU of the Tunnel

Unlike [RFC4301], there is normally no effective MTU (EMTU) on an AGGFRAG tunnel as all IP packet sizes are properly transmitted without requiring IP fragmentation prior to tunnel ingress. That said, a sender MAY allow for explicitly configuring an MTU for the tunnel.

If fragmentation has been disabled on the AGGFRAG tunnel, then the tunnel's EMTU and behaviors are the same as normal IPsec tunnels [RFC4301].

2.3. Exclusive SA Use

This document does not specify mixed use of an AGGFRAG_PAYLOAD enabled SA. A sender **MUST** only send AGGFRAG_PAYLOAD payloads over an SA configured for AGGFRAG mode.

2.4. Modes of Operation

Just as with normal IPsec/ESP tunnels, AGGFRAG tunnels are unidirectional. Bidirectional IP-TFS functionality is achieved by setting up 2 AGGFRAG tunnels, one in either direction.

An AGGFRAG tunnel used for IP-TFS can operate in 2 modes, a non-congestion controlled mode and congestion controlled mode.

2.4.1. Non-Congestion Controlled Mode

In the non-congestion controlled mode, IP-TFS sends fixed-sized packets over an AGGFRAG tunnel at a constant rate. The packet send rate is constant and is not automatically adjusted regardless of any network congestion (e.g., packet loss).

For similar reasons as given in [RFC7510] the non-congestion controlled mode should only be used where the user has full administrative control over the path the tunnel will take. This is required so the user can guarantee the bandwidth and also be sure as to not be negatively affecting network congestion [RFC2914]. In this case packet loss should be reported to the administrator (e.g., via syslog, YANG notification, SNMP traps, etc) so that any failures due to a lack of bandwidth can be corrected.

Non-congestion control mode is also appropriate if ESP over TCP is in use [RFC8229].

2.4.2. Congestion Controlled Mode

With the congestion controlled mode, IP-TFS adapts to network congestion by lowering the packet send rate to accommodate the congestion, as well as raising the rate when congestion subsides. Since overhead is per packet, by allowing for maximal fixed-size packets and varying the send rate transport overhead is minimized.

The output of the congestion control algorithm will adjust the rate at which the ingress sends packets. While this document does not require a specific congestion control algorithm, best current practice **RECOMMENDS** that the algorithm conform to [RFC5348]. Congestion control principles are documented in [RFC2914] as well. [RFC4342] provides an example of the [RFC5348] algorithm which

matches the requirements of IP-TFS (i.e., designed for fixed-size packet and send rate varied based on congestion).

The required inputs for the TCP friendly rate control algorithm described in [RFC5348] are the receiver's loss event rate and the sender's estimated round-trip time (RTT). These values are provided by IP-TFS using the congestion information header fields described in Section 3. In particular, these values are sufficient to implement the algorithm described in [RFC5348].

At a minimum, the congestion information **MUST** be sent, from the receiver and from the sender, at least once per RTT. Prior to establishing an RTT the information **SHOULD** be sent constantly from the sender and the receiver so that an RTT estimate can be established. Not receiving this information over multiple consecutive RTT intervals should be considered a congestion event that causes the sender to adjust its sending rate lower. For example, [RFC4342] calls this the "no feedback timeout" and it is equal to 4 RTT intervals. When a "no feedback timeout" has occurred [RFC4342] halves the sending rate.

An implementation **MAY** choose to always include the congestion information in its AGGFRAG payload header if sending on an IP-TFS enabled SA. Since IP-TFS normally will operate with a large packet size, the congestion information should represent a small portion of the available tunnel bandwidth. An implementation choosing to always send the data **MAY** also choose to only update the "LossEventRate" and "RTT" header field values it sends every "RTT" though.

When choosing a congestion control algorithm (or a selection of algorithms) note that IP-TFS is not providing for reliable delivery of IP traffic, and so per packet ACKs are not required and are not provided.

It is worth noting that the variable send-rate of a congestion controlled AGGFRAG tunnel, is not private; however, this send-rate is being driven by network congestion, and as long as the encapsulated (inner) traffic flow shape and timing are not directly affecting the (outer) network congestion, the variations in the tunnel rate will not weaken the provided inner traffic flow confidentiality.

2.4.2.1. Circuit Breakers

In addition to congestion control, implementations **MAY** choose to define and implement circuit breakers [RFC8084] as a recovery method of last resort. Enabling circuit breakers is also a reason a user may wish to enable congestion information reports even when using the

non-congestion controlled mode of operation. The definition of circuit breakers are outside the scope of this document.

2.5. Summary of Receiver Processing

An AGGFRAG enabled SA receiver has a few tasks to perform.

The receiver MAY process incoming AGGFRAG_PAYLOAD payloads as soon as they arrive as much as it can. I.e., if the incoming AGGFRAG_PAYLOAD packet contains complete inner packet(s), the receiver should extract and transmit them immediately. For partial packets the receiver needs to keep the partial packets in the memory until the they fall out from the reordering window, or until the missing parts of the packets are received, in which case it will reassemble and transmit them. If AGGFRAG_PAYLOAD payload contains multiple packets they SHOULD be sent out in the order they are in the AGGFRAG_PAYLOAD (i.e., keep the original order they were received on the other end). The cost of using this method is that an amplification of out-of-order delivery of inner packets can occur due to inner packet aggregation.

Instead of the method described in the previous paragraph, the receiver MAY reorder out-of-order AGGFRAG_PAYLOAD payloads received into in-sequence-order AGGFRAG_PAYLOAD payloads (Section 2.2.3), and only after it has in-order AGGFRAG_PAYLOAD payload stream, the receiver transmits the inner-packets. Using this method will make sure the packets are sent in-order, i.e., there is no reordering possible, but the cost is that a lost packet will cause delay of up to the lost packet timer interval (or the full reorder window if no lost packet timer is used), and there will be extra burstiness in the output stream (when lost packet is dropped out from the re-order window, all outer packets received after that are then immediately processed, and sent out back to back).

Additionally, if congestion control is enabled, the receiver sends congestion control data (Section 6.1.2) back to the sender as described in Section 2.4.2 and Section 3.

3. Congestion Information

In order to support the congestion control mode, the sender needs to know the loss event rate and to approximate the RTT [RFC5348]. In order to obtain these values, the receiver sends congestion control information on it's SA back to the sender. Thus, to support congestion control the receiver must have a paired SA back to the sender (this is always the case when the tunnel was created using IKEv2). If the SA back to the sender is a non-AGGFRAG_PAYLOAD

enabled SA then an AGGFRAG_PAYLOAD empty payload (i.e., header only) is used to convey the information.

In order to calculate a loss event rate compatible with [RFC5348], the receiver needs to have a round-trip time estimate. Thus the sender communicates this estimate in the "RTT" header field. On startup this value will be zero as no RTT estimate is yet known.

In order for the sender to estimate its "RTT" value, the sender places a timestamp value in the "TVal" header field. On first receipt of this "TVal", the receiver records the new "TVal" value along with the time it arrived locally, subsequent receipt of the same "TVal" MUST NOT update the recorded time.

When the receiver sends its CC header it places this latest recorded "TVal" in the "TEcho" header field, along with 2 delay values, "Echo Delay" and "Transmit Delay". The "Echo Delay" value is the time delta from the recorded arrival time of "TVal" and the current clock in microseconds. The second value, "Transmit Delay", is the receiver's current transmission delay on the tunnel (i.e., the average time between sending packets on its half of the AGGFRAG tunnel).

When the sender receives back its "TVal" in the "TEcho" header field it calculates 2 RTT estimates. The first is the actual delay found by subtracting the "TEcho" value from its current clock and then subtracting "Echo Delay" as well. The second RTT estimate is found by adding the received "Transmit Delay" header value to the senders own transmission delay (i.e., the average time between sending packets on its half of the AGGFRAG tunnel). The larger of these 2 RTT estimates SHOULD be used as the "RTT" value.

The two RTT estimates are required to handle different combinations of faster or slower tunnel packet paths with faster or slower fixed tunnel rates. Choosing the larger of the two values guarantees that the "RTT" is never considered faster than the aggregate transmission delay based on the IP-TFS send rate (the second estimate), as well as never being considered faster than the actual RTT along the tunnel packet path (the first estimate).

The receiver also calculates, and communicates in the "LossEventRate" header field, the loss event rate for use by the sender. This is slightly different from [RFC4342] which periodically sends all the loss interval data back to the sender so that it can do the calculation. See Appendix B for a suggested way to calculate the loss event rate value. Initially this value will be zero (indicating no loss) until enough data has been collected by the receiver to update it.

3.1. ECN Support

In addition to normal packet loss information AGGFRAG mode supports use of the ECN bits in the encapsulating IP header [RFC3168] for identifying congestion. If ECN use is enabled and a packet arrives at the egress (receiving) side with the Congestion Experienced (CE) value set, then the receiver considers that packet as being dropped, although it does not drop it. The receiver MUST set the E bit in any AGGFRAG_PAYLOAD payload header containing a "LossEventRate" value derived from a CE value being considered.

As noted in [RFC3168] the ECN bits are not protected by IPsec and thus may constitute a covert channel. For this reason, ECN use SHOULD NOT be enabled by default.

4. Configuration of AGGFRAG Tunnels for IP-TFS

IP-TFS is meant to be deployable with a minimal amount of configuration. All IP-TFS specific configuration should be specified at the unidirectional tunnel ingress (sending) side. It is intended that non-IKEv2 operation is supported, at least, with local static configuration.

4.1. Bandwidth

Bandwidth is a local configuration option. For non-congestion controlled mode, the bandwidth SHOULD be configured. For congestion controlled mode, the bandwidth can be configured or the congestion control algorithm discovers and uses the maximum bandwidth available. No standardized configuration method is required.

4.2. Fixed Packet Size

The fixed packet size to be used for the tunnel encapsulation packets MAY be configured manually or can be automatically determined using other methods such as PLMTUD ([RFC4821], [RFC8899]) or PMTUD ([RFC1191], [RFC8201]). As PMTUD is known to have issues, PLMTUD is considered the more robust option. No standardized configuration method is required.

4.3. Congestion Control

Congestion control is a local configuration option. No standardized configuration method is required.

5. IKEv2

5.1. USE_AGGFRAG Notification Message

As mentioned previously AGGFRAG tunnels utilize ESP payloads of type AGGFRAG_PAYLOAD.

When using IKEv2, a new "USE_AGGFRAG" Notification Message enables the AGGFRAG_PAYLOAD payload on a child SA pair. The method used is similar to how USE_TRANSPORT_MODE is negotiated, as described in [RFC7296].

To request use of the AGGFRAG_PAYLOAD payload on the Child SA pair, the initiator includes the USE_AGGFRAG notification in an SA payload requesting a new Child SA (either during the initial IKE_AUTH or during CREATE_CHILD_SA exchanges). If the request is accepted then the response MUST also include a notification of type USE_AGGFRAG. If the responder declines the request the child SA will be established without AGGFRAG_PAYLOAD payload use enabled. If this is unacceptable to the initiator, the initiator MUST delete the child SA.

As the use of the AGGFRAG_PAYLOAD payload is currently only defined for non-transport mode tunnels, the USE_AGGFRAG notification MUST NOT be combined with USE_TRANSPORT notification.

The USE_AGGFRAG notification contains a 1 octet payload of flags that specify requirements from the sender of the notification. If any requirement flags are not understood or cannot be supported by the receiver then the receiver SHOULD NOT enable use of AGGFRAG_PAYLOAD (either by not responding with the USE_AGGFRAG notification, or in the case of the initiator, by deleting the child SA if the now established non-AGGFRAG_PAYLOAD using SA is unacceptable).

The notification type and payload flag values are defined in Section 6.1.4.

6. Packet and Data Formats

The packet and data formats defined below are generic with the intent of allowing for non-IP-TFS uses, but such uses are outside the scope of this document.

6.1. AGGFRAG_PAYLOAD Payload

ESP Next Header value: 0x5

An AGGFRAG payload is identified by the ESP Next Header value AGGFRAG_PAYLOAD which has the value 0x5. The value 5 was chosen to not conflict with other used values. The first octet of this payload indicates the format of the remaining payload data.

```

  0 1 2 3 4 5 6 7
+-+--+--+--+--+--+--+
|  Sub-type    | ...
+-+--+--+--+--+--+--+

```

Sub-type:

An 8-bit value indicating the payload format.

This document defines 2 payload sub-types. These payload formats are defined in the following sections.

6.1.1. Non-Congestion Control AGGFRAG_PAYLOAD Payload Format

The non-congestion control AGGFRAG_PAYLOAD payload is comprised of a 4 octet header followed by a variable amount of "DataBlocks" data as shown below.

```

                                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
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  Sub-Type (0) |  Reserved   |          BlockOffset          |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|          DataBlocks ...
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

Sub-type:

An octet indicating the payload format. For this non-congestion control format, the value is 0.

Reserved:

An octet set to 0 on generation, and ignored on receipt.

BlockOffset:

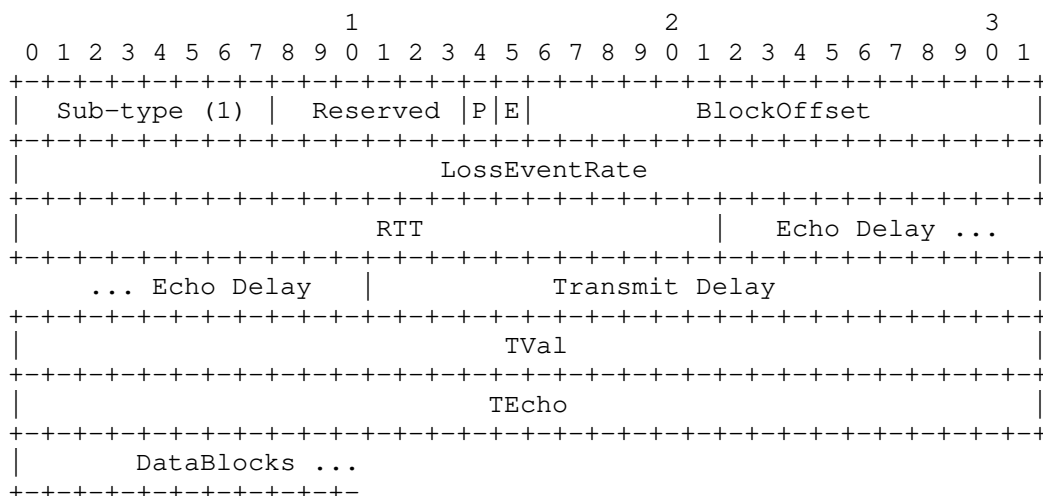
A 16-bit unsigned integer counting the number of octets of "DataBlocks" data before the start of a new data block. If the start of a new data block occurs in a subsequent payload the "BlockOffset" will point past the end of the "DataBlocks" data. In this case all the "DataBlocks" data belongs to the current data block being assembled. When the "BlockOffset" extends into subsequent payloads it continues to only count "DataBlocks" data (i.e., it does not count subsequent packets non-"DataBlocks" data such as header octets).

DataBlocks:

Variable number of octets that begins with the start of a data block, or the continuation of a previous data block, followed by zero or more additional data blocks.

6.1.2. Congestion Control AGGFRAG_PAYLOAD Payload Format

The congestion control AGGFRAG_PAYLOAD payload is comprised of a 24 octet header followed by a variable amount of "DataBlocks" data as shown below.

**Sub-type:**

An octet indicating the payload format. For this congestion control format, the value is 1.

Reserved:

A 6-bit field set to 0 on generation, and ignored on receipt.

P:

A 1-bit value if set indicates that PLMTUD probing is in progress. This information can be used to avoid treating missing packets as loss events by the CC algorithm when running the PLMTUD probe algorithm.

E:

A 1-bit value if set indicates that Congestion Experienced (CE) ECN bits were received and used in deriving the reported "LossEventRate".

BlockOffset:

The same value as the non-congestion controlled payload format value.

LossEventRate:

A 32-bit value specifying the inverse of the current loss event rate as calculated by the receiver. A value of zero indicates no loss. Otherwise the loss event rate is "1/LossEventRate".

RTT:

A 22-bit value specifying the sender's current round-trip time estimate in microseconds. The value MAY be zero prior to the sender having calculated a round-trip time estimate. The value SHOULD be set to zero on non-AGGFRAG_PAYLOAD enabled SAs. If the value is equal to or larger than "0x3FFFFFF" it MUST be set to "0x3FFFFFF".

Echo Delay:

A 21-bit value specifying the delay in microseconds incurred between the receiver first receiving the "TVal" value which it is sending back in "TEcho". If the value is equal to or larger than "0x1FFFFFF" it MUST be set to "0x1FFFFFF".

Transmit Delay:

A 21-bit value specifying the transmission delay in microseconds. This is the fixed (or average) delay on the receiver between it sending packets on the IPTFS tunnel. If the value is equal to or larger than "0x1FFFFFF" it MUST be set to "0x1FFFFFF".

TVal:

An opaque 32-bit value that will be echoed back by the receiver in later packets in the "TEcho" field, along with an "Echo Delay" value of how long that echo took.

TEcho:

The opaque 32-bit value from a received packet's "TVal" field. The received "TVal" is placed in "TEcho" along with an "Echo Delay" value indicating how long it has been since receiving the "TVal" value.

DataBlocks:

Variable number of octets that begins with the start of a data block, or the continuation of a previous data block, followed by zero or more additional data blocks. For the special case of sending congestion control information on a non-IP-TFS enabled SA this value MUST be empty (i.e., be zero octets long).

6.1.3. Data Blocks

```

          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
+-----+-----+-----+
| Type   | IPv4, IPv6 or pad...
+-----+-----+-----+

```

Type:

A 4-bit field where 0x0 identifies a pad data block, 0x4 indicates an IPv4 data block, and 0x6 indicates an IPv6 data block.

6.1.3.1. IPv4 Data Block

```

          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
+-----+-----+-----+
| 0x4   | IHL   | TypeOfService | TotalLength |
+-----+-----+-----+
| Rest of the inner packet ...
+-----+-----+-----+

```

These values are the actual values within the encapsulated IPv4 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:

A 4-bit value of 0x4 indicating IPv4 (i.e., first nibble of the IPv4 packet).

TotalLength:

The 16-bit unsigned integer "Total Length" field of the IPv4 inner packet.

6.1.3.2. IPv6 Data Block

```

          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
+-----+-----+-----+
| 0x6   | TrafficClass | FlowLabel |
+-----+-----+-----+
| PayloadLength | Rest of the inner packet ...
+-----+-----+-----+

```

These values are the actual values within the encapsulated IPv6 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:

A 4-bit value of 0x6 indicating IPv6 (i.e., first nibble of the IPv6 packet).

PayloadLength:

The 16-bit unsigned integer "Payload Length" field of the inner IPv6 inner packet.

6.1.3.3. Pad Data Block

```

                                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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x0 | Padding ... |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type:

A 4-bit value of 0x0 indicating a padding data block.

Padding:

Extends to end of the encapsulating packet.

6.1.4. IKEv2 USE_AGGFRAG Notification Message

As discussed in Section 5.1, a notification message USE_AGGFRAG is used to negotiate use of the ESP AGGFRAG_PAYLOAD Next Header value.

The USE_AGGFRAG Notification Message State Type is (TBD2).

The notification payload contains 1 octet of requirement flags. There are currently 2 requirement flags defined. This may be revised by later specifications.

```

+---+---+---+---+---+---+---+---+
| 0 | 0 | 0 | 0 | 0 | 0 | C | D |
+---+---+---+---+---+---+---+---+

```

0:

6 bits - reserved, MUST be zero on send, unless defined by later specifications.

C:

Congestion Control bit. If set, then the sender is requiring that congestion control information MUST be returned to it periodically as defined in Section 3.

D:

Don't Fragment bit. If set, indicates the sender of the notify message does not support receiving packet fragments (i.e., inner packets MUST be sent using a single "Data Block"). This value only applies to what the sender is capable of receiving; the sender MAY still send packet fragments unless similarly restricted by the receiver in its USE_AGGFRAG notification.

7. IANA Considerations

7.1. AGGFRAG_PAYLOAD Sub-Type Registry

This document requests IANA create a registry called "AGGFRAG_PAYLOAD Sub-Type Registry" under a new category named "ESP AGGFRAG_PAYLOAD Parameters". The registration policy for this registry is "Expert Review" ([RFC8126] and [RFC7120]).

Name:

AGGFRAG_PAYLOAD Sub-Type Registry

Description:

AGGFRAG_PAYLOAD Payload Formats.

Reference:

This document

This initial content for this registry is as follows:

Sub-Type	Name	Reference
0	Non-Congestion Control Format	This document
1	Congestion Control Format	This document
3-255	Reserved	

7.2. USE_AGGFRAG Notify Message Status Type

This document requests a status type USE_AGGFRAG be allocated from the "IKEv2 Notify Message Types - Status Types" registry.

Value:

TBD2

Name:

USE_AGGFRAG

Reference:

This document

8. Security Considerations

This document describes an aggregation and fragmentation mechanism and its use to add TFC to IP traffic. The use described is expected to increase the security of the traffic being transported. Other than the additional security afforded by using this mechanism, IP-TFS utilizes the security protocols [RFC4303] and [RFC7296] and so their security considerations apply to IP-TFS as well.

As noted in (Section 3.1) the ECN bits are not protected by IPsec and thus may constitute a covert channel. For this reason, ECN use SHOULD NOT be enabled by default.

As noted previously in Section 2.4.2, for TFC to be fully maintained the encapsulated traffic flow should not be affecting network congestion in a predictable way, and if it would be then non-congestion controlled mode use should be considered instead.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [AppCrypt] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 11 2017.

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, DOI 10.17487/RFC4342, March 2006, <<https://www.rfc-editor.org/info/rfc4342>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <<https://www.rfc-editor.org/info/rfc5348>>.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <<https://www.rfc-editor.org/info/rfc7120>>.

- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.
- [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <<https://www.rfc-editor.org/info/rfc8084>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.
- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.
- [RFC8546] Trammell, B. and M. Kuehlewind, "The Wire Image of a Network Protocol", RFC 8546, DOI 10.17487/RFC8546, April 2019, <<https://www.rfc-editor.org/info/rfc8546>>.
- [RFC8899] Fairhurst, G., Jones, T., Tuexen, M., Ruengeler, I., and T. Voelker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <<https://www.rfc-editor.org/info/rfc8899>>.

Appendix A. Example Of An Encapsulated IP Packet Flow

Below an example inner IP packet flow within the encapsulating tunnel packet stream is shown. Notice how encapsulated IP packets can start and end anywhere, and more than one or less than 1 may occur in a single encapsulating packet.

```

Offset: 0      Offset: 100    Offset: 2900    Offset: 1400
[ ESP1  (1500) ][ ESP2  (1500) ][ ESP3  (1500) ][ ESP4  (1500) ]
[--800--][--800--][60][--240--][--4000-----][pad]

```

Figure 3: Inner and Outer Packet Flow

The encapsulated IP packet flow (lengths include IP header and payload) is as follows: an 800 octet packet, an 800 octet packet, a 60 octet packet, a 240 octet packet, a 4000 octet packet.

The "BlockOffset" values in the 4 AGGFRAG payload headers for this packet flow would thus be: 0, 100, 2900, 1400 respectively. The first encapsulating packet ESP1 has a zero "BlockOffset" which points at the IP data block immediately following the AGGFRAG header. The following packet ESP2s "BlockOffset" points inward 100 octets to the start of the 60 octet data block. The third encapsulating packet ESP3 contains the middle portion of the 4000 octet data block so the offset points past its end and into the forth encapsulating packet. The fourth packet ESP4s offset is 1400 pointing at the padding which follows the completion of the continued 4000 octet packet.

Appendix B. A Send and Loss Event Rate Calculation

The current best practice indicates that congestion control SHOULD be done in a TCP friendly way. A TCP friendly congestion control algorithm is described in [RFC5348]. For this IP-TFS use case (as with [RFC4342]) the (fixed) packet size is used as the segment size for the algorithm. The main formula in the algorithm for the send rate is then as follows:

$$X = \frac{1}{R * (\text{sqrt}(2*p/3) + 12*\text{sqrt}(3*p/8)*p*(1+32*p^2))}$$

Where "X" is the send rate in packets per second, "R" is the round trip time estimate and "p" is the loss event rate (the inverse of which is provided by the receiver).

In addition the algorithm in [RFC5348] also uses an "X_recv" value (the receiver's receive rate). For IP-TFS one MAY set this value according to the sender's current tunnel send-rate ("X").

The IP-TFS receiver, having the RTT estimate from the sender can use the same method as described in [RFC5348] and [RFC4342] to collect the loss intervals and calculate the loss event rate value using the weighted average as indicated. The receiver communicates the inverse of this value back to the sender in the AGGFRAG_PAYLOAD payload header field "LossEventRate".

The IP-TFS sender now has both the "R" and "p" values and can calculate the correct sending rate. If following [RFC5348] the sender should also use the slow start mechanism described therein when the IP-TFS SA is first established.

Appendix C. Comparisons of IP-TFS

C.1. Comparing Overhead

For comparing overhead the overhead of ESP for both normal and AGGFRAG tunnel packets must be calculated, and so an algorithm for encryption and authentication must be chosen. For the data below AES-GCM-256 was selected. This leads to an IP+ESP overhead of 54.

$$54 = 20 \text{ (IP)} + 8 \text{ (ESPH)} + 2 \text{ (ESPF)} + 8 \text{ (IV)} + 16 \text{ (ICV)}$$

Additionally, for IP-TFS, non-congestion control AGGFRAG_PAYLOAD headers were chosen which adds 4 octets for a total overhead of 58.

C.1.1. IP-TFS Overhead

For comparison the overhead of AGGFRAG payload is 58 octets per outer packet. Therefore the octet overhead per inner packet is 58 divided by the number of outer packets required (fractional allowed). The overhead as a percentage of inner packet size is a constant based on the Outer MTU size.

$$\begin{aligned} \text{OH} &= 58 / \text{Outer Payload Size} / \text{Inner Packet Size} \\ \text{OH \% of Inner Packet Size} &= 100 * \text{OH} / \text{Inner Packet Size} \\ \text{OH \% of Inner Packet Size} &= 5800 / \text{Outer Payload Size} \end{aligned}$$

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	518	1442	8942
<hr/>			
40	11.20%	4.02%	0.65%
576	11.20%	4.02%	0.65%
1500	11.20%	4.02%	0.65%
9000	11.20%	4.02%	0.65%

Figure 4: IP-TFS Overhead as Percentage of Inner Packet Size

C.1.2. ESP with Padding Overhead

The overhead per inner packet for constant-send-rate padded ESP (i.e., traditional IPsec TFC) is 36 octets plus any padding, unless fragmentation is required.

When fragmentation of the inner packet is required to fit in the outer IPsec packet, overhead is the number of outer packets required to carry the fragmented inner packet times both the inner IP overhead (20) and the outer packet overhead (54) minus the initial inner IP overhead plus any required tail padding in the last encapsulation packet. The required tail padding is the number of required packets times the difference of the Outer Payload Size and the IP Overhead minus the Inner Payload Size. So:

Inner Payload Size = IP Packet Size - IP Overhead

Outer Payload Size = MTU - IPsec Overhead

$$NF0 = \frac{\text{Inner Payload Size}}{\text{Outer Payload Size} - \text{IP Overhead}}$$

NF = CEILING(NF0)

$$\begin{aligned} OH &= NF * (\text{IP Overhead} + \text{IPsec Overhead}) \\ &\quad - \text{IP Overhead} \\ &\quad + NF * (\text{Outer Payload Size} - \text{IP Overhead}) \\ &\quad - \text{Inner Payload Size} \end{aligned}$$

$$\begin{aligned} OH &= NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ &\quad - (\text{IP Overhead} + \text{Inner Payload Size}) \end{aligned}$$

$$\begin{aligned} OH &= NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ &\quad - \text{Inner Packet Size} \end{aligned}$$

C.2. Overhead Comparison

The following tables collect the overhead values for some common L3 MTU sizes in order to compare them. The first table is the number of octets of overhead for a given L3 MTU sized packet. The second table is the percentage of overhead in the same MTU sized packet.

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
L3 MTU	576	1500	9000	576	1500	9000
PSize	522	1446	8946	518	1442	8942
<hr/>						
40	482	1406	8906	4.5	1.6	0.3
128	394	1318	8818	14.3	5.1	0.8
256	266	1190	8690	28.7	10.3	1.7
518	4	928	8428	58.0	20.8	3.4
576	576	870	8370	64.5	23.2	3.7
1442	286	4	7504	161.5	58.0	9.4
1500	228	1500	7446	168.0	60.3	9.7
8942	1426	1558	4	1001.2	359.7	58.0
9000	1368	1500	9000	1007.7	362.0	58.4

Figure 5: Overhead comparison in octets

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000	576	1500	9000
PSize	522	1446	8946	518	1442	8942
<hr/>						
40	1205.0%	3515.0%	22265.0%	11.20%	4.02%	0.65%
128	307.8%	1029.7%	6889.1%	11.20%	4.02%	0.65%
256	103.9%	464.8%	3394.5%	11.20%	4.02%	0.65%
518	0.8%	179.2%	1627.0%	11.20%	4.02%	0.65%
576	100.0%	151.0%	1453.1%	11.20%	4.02%	0.65%
1442	19.8%	0.3%	520.4%	11.20%	4.02%	0.65%
1500	15.2%	100.0%	496.4%	11.20%	4.02%	0.65%
8942	15.9%	17.4%	0.0%	11.20%	4.02%	0.65%
9000	15.2%	16.7%	100.0%	11.20%	4.02%	0.65%

Figure 6: Overhead as Percentage of Inner Packet Size

C.3. Comparing Available Bandwidth

Another way to compare the two solutions is to look at the amount of available bandwidth each solution provides. The following sections consider and compare the percentage of available bandwidth. For the sake of providing a well understood baseline normal (unencrypted) Ethernet as well as normal ESP values are included.

C.3.1. Ethernet

In order to calculate the available bandwidth the per packet overhead is calculated first. The total overhead of Ethernet is 14+4 octets of header and CRC plus and additional 20 octets of framing (preamble, start, and inter-packet gap) for a total of 38 octets. Additionally the minimum payload is 46 octets.

Size	E + P	E + P	E + P	PTFS	PTFS	PTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	92	92	92	96	96	96	38	74
<hr/>								
40	614	1538	9038	47	42	40	84	114
128	614	1538	9038	151	136	129	166	202
256	614	1538	9038	303	273	258	294	330
518	614	1538	9038	614	552	523	574	610
576	1228	1538	9038	682	614	582	614	650
1442	1842	1538	9038	1709	1538	1457	1498	1534
1500	1842	3076	9038	1777	1599	1516	1538	1574
8942	11052	10766	9038	10599	9537	9038	8998	9034
9000	11052	10766	18076	10667	9599	9096	9038	9074

Figure 7: L2 Octets Per Packet

Size	E + P	E + P	E + P	PTFS	PTFS	PTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	92	92	92	96	96	96	38	74
<hr/>								
40	2.0M	0.8M	0.1M	26.4M	29.3M	30.9M	14.9M	11.0M
128	2.0M	0.8M	0.1M	8.2M	9.2M	9.7M	7.5M	6.2M
256	2.0M	0.8M	0.1M	4.1M	4.6M	4.8M	4.3M	3.8M
518	2.0M	0.8M	0.1M	2.0M	2.3M	2.4M	2.2M	2.1M
576	1.0M	0.8M	0.1M	1.8M	2.0M	2.1M	2.0M	1.9M
1442	678K	812K	138K	731K	812K	857K	844K	824K
1500	678K	406K	138K	703K	781K	824K	812K	794K
8942	113K	116K	138K	117K	131K	138K	139K	138K
9000	113K	116K	69K	117K	130K	137K	138K	137K

Figure 8: Packets Per Second on 10G Ethernet

Size	E + P	E + P	E + P	PTFS	PTFS	PTFS	Enet	ESP
	590	1514	9014	590	1514	9014	any	any
	92	92	92	96	96	96	38	74
<hr/>								
40	6.51%	2.60%	0.44%	84.36%	93.76%	98.94%	47.62%	35.09%
128	20.85%	8.32%	1.42%	84.36%	93.76%	98.94%	77.11%	63.37%
256	41.69%	16.64%	2.83%	84.36%	93.76%	98.94%	87.07%	77.58%
518	84.36%	33.68%	5.73%	84.36%	93.76%	98.94%	93.17%	87.50%
576	46.91%	37.45%	6.37%	84.36%	93.76%	98.94%	93.81%	88.62%
1442	78.28%	93.76%	15.95%	84.36%	93.76%	98.94%	97.43%	95.12%
1500	81.43%	48.76%	16.60%	84.36%	93.76%	98.94%	97.53%	95.30%
8942	80.91%	83.06%	98.94%	84.36%	93.76%	98.94%	99.58%	99.18%
9000	81.43%	83.60%	49.79%	84.36%	93.76%	98.94%	99.58%	99.18%

Figure 9: Percentage of Bandwidth on 10G Ethernet

A sometimes unexpected result of using an AGGFRAG tunnel (or any packet aggregating tunnel) is that, for small to medium sized packets, the available bandwidth is actually greater than native Ethernet. This is due to the reduction in Ethernet framing overhead. This increased bandwidth is paid for with an increase in latency. This latency is the time to send the unrelated octets in the outer tunnel frame. The following table illustrates the latency for some common values on a 10G Ethernet link. The table also includes latency introduced by padding if using ESP with padding.

	ESP+Pad 1500	ESP+Pad 9000	IP-TFS 1500	IP-TFS 9000
<hr/>				
40	1.12 us	7.12 us	1.17 us	7.17 us
128	1.05 us	7.05 us	1.10 us	7.10 us
256	0.95 us	6.95 us	1.00 us	7.00 us
518	0.74 us	6.74 us	0.79 us	6.79 us
576	0.70 us	6.70 us	0.74 us	6.74 us
1442	0.00 us	6.00 us	0.05 us	6.05 us
1500	1.20 us	5.96 us	0.00 us	6.00 us

Figure 10: Added Latency

Notice that the latency values are very similar between the two solutions; however, whereas IP-TFS provides for constant high bandwidth, in some cases even exceeding native Ethernet, ESP with padding often greatly reduces available bandwidth.

Appendix D. Acknowledgements

We would like to thank Don Fedyk for help in reviewing and editing this work. We would also like to thank Michael Richardson, Sean Turner, Valery Smyslov and Tero Kivinen for reviews and many suggestions for improvements, as well as Joseph Touch for the transport area review and suggested improvements.

Appendix E. Contributors

The following people made significant contributions to this document.

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Author's Address

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 4, 2022

V. Smyslov
ELVIS-PLUS
August 31, 2021

Announcing Supported Authentication Methods in IKEv2
draft-smyslov-ipsecme-ikev2-auth-announce-04

Abstract

This specification defines a mechanism that allows the Internet Key Exchange version 2 (IKEv2) implementations to indicate the list of supported authentication methods to their peers while establishing IKEv2 Security Association (SA). This mechanism improves interoperability when IKEv2 partners are configured with multiple different credentials to authenticate each other.

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 <https://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 March 4, 2022.

Copyright Notice

Copyright (c) 2021 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 (<https://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	2
2. Terminology and Notation	3
3. Protocol Details	3
3.1. Exchanges	3
3.2. SUPPORTED_AUTH_METHODS Notify	4
3.2.1. 2-octet Announcement	5
3.2.2. 3-octet Announcement	6
3.2.3. Multi-octet Announcement	7
4. Security Considerations	8
5. IANA Considerations	8
6. References	8
6.1. Normative References	8
6.2. Informative References	9
Author's Address	9

1. Introduction

The Internet Key Exchange version 2 (IKEv2) protocol, defined in [RFC7296], performs authenticated key exchange in IPsec. IKEv2, unlike its predecessor IKEv1, defined in [RFC2409], doesn't include a mechanism to negotiate an authentication method that the peers would use to authenticate each other. It is assumed that each peer selects whatever authentication method it thinks is appropriate, depending on authentication credentials it has.

This approach generally works well when there is no ambiguity in selecting authentication credentials. The problem may arise when there are several credentials of different type configured on one peer, while only some of them are supported on the other peer. Another problem situation is when a single credential may be used to produce different types of authentication tokens (e.g. signatures of different formats). Emerging post-quantum signature algorithms may bring additional challenges for implementations, especially if so called hybrid schemes are used (e.g. see [I-D.ounsworth-pq-composite-sigs]).

This specification defines an extension to the IKEv2 protocol that allows peers to announce their supported authentication methods, thus decreasing risks of SA establishment failure in situations when there are several ways for the peers to authenticate themselves.

2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Protocol Details

The idea is that each party sends a list of authentication methods it supports to its peer. In addition, the sending party may optionally specify that some of the authentication methods are only to be used with particular trust anchors. Upon receiving this information the peer may take it into account while selecting an algorithm for its authentication if several methods are available.

3.1. Exchanges

If the responder is willing to use this extension, it includes a new notification SUPPORTED_AUTH_METHODS in a response message of the IKE_SA_INIT exchange. This notification contains a list of authentication methods supported by the responder.

Initiator	Responder
-----	-----
HDR, SAI1, KEi, Ni -->	<-- HDR, SAR1, KEr, Nr, [CERTREQ,] [N(SUPPORTED_AUTH_METHODS)]

Figure 1: IKE_SA_INIT Exchange

If the initiator doesn't support this extension, it will ignore the received notification as an unknown status notify. Otherwise, it MAY send the SUPPORTED_AUTH_METHODS notification in the IKE_AUTH request message, with a list of authentication methods supported by the initiator.

Initiator	Responder
-----	-----
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAI2, TSi, TSr, [N(SUPPORTED_AUTH_METHODS)] } -->	<-- HDR, SK {IDr, [CERT,] AUTH, SAR2, TSi, TSr }

Figure 2: IKE_AUTH Exchange

Since the responder sends the SUPPORTED_AUTH_METHODS notification in the IKE_SA_INIT exchange, it must take care that the size of the response message wouldn't grow too much so that IP fragmentation takes place. If the following conditions are met:

- o the SUPPORTED_AUTH_METHODS notification to be included is so large, that the responder suspects that IP fragmentation of the resulting IKE_SA_INIT response message may happen;
- o both peers support the IKE_INTERMEDIATE exchange, defined in [I-D.ietf-ipsecme-ikev2-intermediate] (i.e. the responder has received and is going to send the INTERMEDIATE_EXCHANGE_SUPPORTED notification);

then the responder may choose not to send actual list of the supported authentication methods in the IKE_SA_INIT exchange and instead ask the initiator to start the IKE_INTERMEDIATE exchange for the list to be sent in. In this case the responder includes SUPPORTED_AUTH_METHODS notification containing no data in the IKE_SA_INIT response.

If the initiator receives the empty SUPPORTED_AUTH_METHODS notification in the IKE_SA_INIT exchange, it means that the responder is going to send the list of the supported authentication methods in the IKE_INTERMEDIATE exchange. If this exchange is to be initiated anyway for some other reason, then the responder MUST use it to send the SUPPORTED_AUTH_METHODS notification. Otherwise, the initiator MAY start the IKE_INTERMEDIATE exchange just for this sole purpose by sending an empty request message.

Initiator	Responder
-----	-----
HDR, SK {...} -->	<-- HDR, SK {...
	[N(SUPPORTED_AUTH_METHODS)] }

Figure 3: IKE_INTERMEDIATE Exchange

Note, that sending the SUPPORTED_AUTH_METHODS notification and using information obtained from it is optional for both the initiator and the responder.

3.2. SUPPORTED_AUTH_METHODS Notify

The format of the SUPPORTED_AUTH_METHODS notification is shown below.

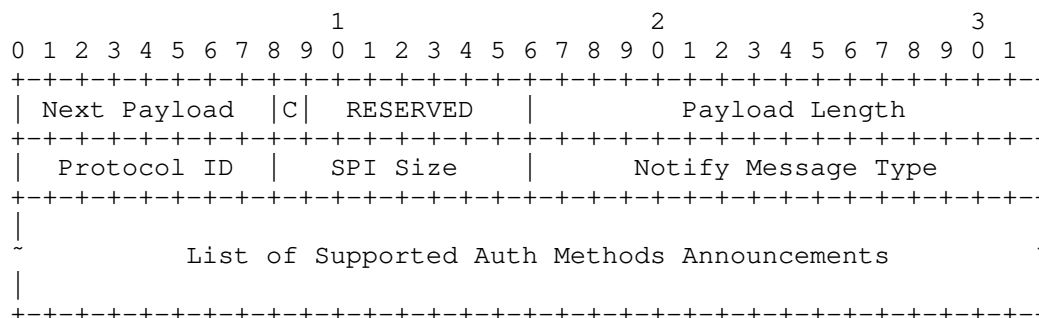


Figure 4: SUPPORTED_AUTH_METHODS Notify

The Notify payload format is defined in Section 3.10 of [RFC7296]. When a Notify payload of type SUPPORTED_AUTH_METHODS is sent, the Protocol ID field is set to 0, the SPI Size is set to 0, meaning there is no SPI field, and the Notify Message Type is set to <TBA by IANA>.

The Notification Data field contains the list of supported authentication methods announcements. Each individual announcement is a variable-size data blob, which format depends on the announced authentication method. The blob always starts with an octet containing the length of the blob followed by an octet containing the authentication method. Authentication methods are represented as values from the "IKEv2 Authentication Method" registry defined in [IKEV2-IANA]. The meaning of the remaining octets of the blob, if any, depends on the authentication method and is defined below. Note, that for the currently defined authentication methods the length octet fully defines both the format and the semantics of the blob.

If more authentication methods are defined in future, the corresponding documents must describe the semantics of the announcements for these methods. Implementations MUST skip announcements which semantics they don't understand.

3.2.1. 2-octet Announcement

If the announcement contains an authentication method that is not concerned with public key cryptography, then the following format is used.

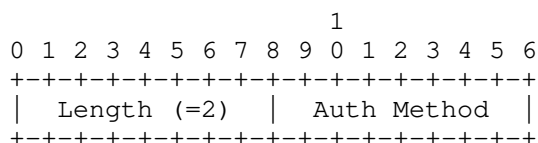


Figure 5: Supported Authentication Method

- o Length - Length of the blob, must be 2 for this case.
- o Auth Method - Announced authentication method.

This format is applicable for the authentication methods "Shared Key Message Integrity Code" (2) and "NULL Authentication" (13). Note, that authentication method "Generic Secure Password Authentication Method" (12) would also fall in this category, however it is negotiated separately (see [RFC6467] and for this reason there is no point to announce it via this mechanism.

3.2.2. 3-octet Announcement

If the announcement contains an authentication method that is concerned with public key cryptography, then the following format is used. This format allows to link the announcement with a particular trust anchor from the Certificate Request payload.

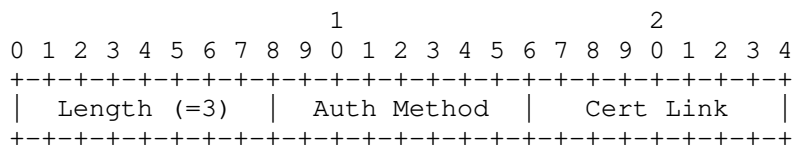


Figure 6: Supported Authentication Method

- o Length - Length of the blob, must be 3 for this case.
- o Auth Method - Announced authentication method.
- o Cert Link - Link this announcement to a particular CA.

If the Cert Link field contains non-zero value N, it means that the announced authentication method is intended to be used only with the N-th trust anchor (CA certificate) from the Certificate Request payload(s) sent by this peer. If it is zero, then this authentication method may be used with any of CAs, that are not linked to any other announcement. If multiple CERTREQ payloads were sent, the CAs from all of them are treated as a single list for the purpose of the linking. If no Certificate Request payload were

receives, the content of this field MUST be ignored and treated as zero.

This format is applicable for the authentication methods "RSA Digital Signature" (1), "DSS Digital Signature" (3), "ECDSA with SHA-256 on the P-256 curve" (9), "ECDSA with SHA-384 on the P-384 curve" (10) and "ECDSA with SHA-512 on the P-512 curve" (11). Note however, that these authentication methods are currently superseded by the "Digital Signature" (14) authentication method, which has a different announcement format, described below.

3.2.3. Multi-octet Announcement

The following format is currently used only with the "Digital Signature" (14) authentication method.

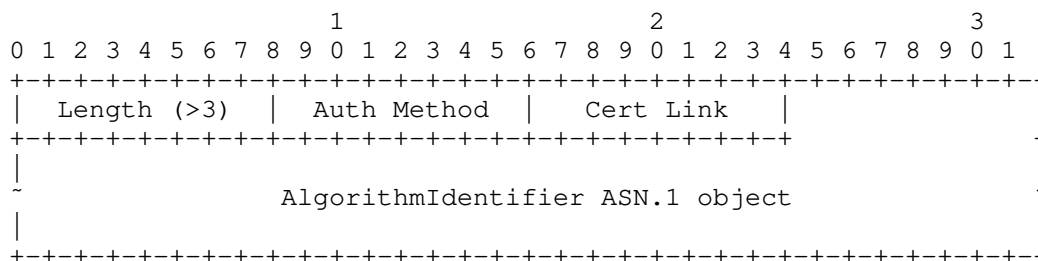


Figure 7: Supported Authentication Method

- o Length - Length of the blob, must be greater than 3 for this case.
- o Auth Method - Announced authentication method, currently may only be 14 ("Digital Signature").
- o Cert Link - Link this announcement to a particular CA; see Section 3.2.2 for details.
- o AlgorithmIdentifier ASN.1 object - DER-encoded ASN.1 object AlgorithmIdentifier.

The "Digital Signature" authentication method, defined in [RFC7427], supersedes previously defined signature authentication methods. In this case the real authentication algorithm is identified via AlgorithmIdentifier ASN.1 object. Appendix A in [RFC7427] contains examples of Commonly Used ASN.1 Objects.

4. Security Considerations

Security considerations for IKEv2 protocol are discussed in [RFC7296]. It is assumed that this extension of the IKEv2 doesn't add new vulnerabilities to the protocol.

5. IANA Considerations

This document defines a new Notify Message Types in the "Notify Message Types - Status Types" registry:

<TBA> SUPPORTED_AUTH_METHODS

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <<https://www.rfc-editor.org/info/rfc7427>>.
- [I-D.ietf-ipsecme-ikev2-intermediate] Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-07 (work in progress), August 2021.
- [IKEV2-IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <<http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7>>.

6.2. Informative References

- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, DOI 10.17487/RFC6467, December 2011, <<https://www.rfc-editor.org/info/rfc6467>>.
- [I-D.ounsworth-pq-composite-sigs] Ounsworth, M. and M. Pala, "Composite Signatures For Use In Internet PKI", draft-ounsworth-pq-composite-sigs-05 (work in progress), July 2021.

Author's Address

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru