

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

D. Carrel
B. Weis
Cisco Systems
July 2, 2018

IPsec Key Exchange using a Controller
draft-carrel-ipsecme-controller-ike-00

Abstract

This document presents a key exchange method allowing devices managed by a controller (e.g., an SDN management station) to create private pair-wise IPsec SAs without IKEv2 or any other direct peer-to-peer session establishment messages.

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 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 |
| 1.1. | Requirements Language | 4 |
| 2. | Overview | 4 |
| 3. | Generating Initial IPsec SAs | 5 |
| 4. | Rekey of IPsec SAs | 6 |
| 4.1. | Single IPsec Device Rekey | 7 |
| 4.2. | Simultaneous Rekey of IPsec Devices | 9 |
| 5. | IPsec Database Generation | 12 |
| 5.1. | The Security Policy Database (SPD) | 12 |
| 5.2. | Security Association Database (SAD) | 12 |
| 5.2.1. | Generating Keying Material for IPsec SAs | 12 |
| 5.3. | Peer Authorization Database (PAD) | 14 |
| 6. | Policy distributed by the Controller | 14 |
| 6.1. | IPsec policy negotiation | 15 |
| 7. | Security Considerations | 16 |
| 8. | IANA Considerations | 17 |
| 9. | Acknowledgements | 17 |
| 10. | References | 17 |
| 10.1. | Normative References | 17 |
| 10.2. | Informative References | 18 |
| Appendix A. | Example Controller protocols | 18 |
| A.1. | Example: I2NSF | 18 |
| A.2. | Example: Network Controller | 19 |
| A.3. | Additional controller protocol considerations | 19 |
| A.3.1. | Peer-to-peer distribution of IPsec policy | 20 |
| A.3.2. | Ordering of messages distributed to a controller | 21 |
| Appendix B. | Differences between IKEv2 and Controller based methods | 21 |
| | Authors' Addresses | 21 |

[1. Introduction](#)

Network architectures typically have included network devices directly communicating using network control protocols such as routing and signaling protocols. Additionally, secured communications between these network devices are usually accomplished with a key agreement protocol such as IKEv2 [[RFC7296](#)], in which the network devices directly authenticate each other and agree upon security policy and keying material to protect communications between themselves. However, controller-based network architectures (sometimes called "Software-Defined Networking") are now being defined [[RFC7426](#)] [[RFC8192](#)] and implemented. In controller-based network architectures, control protocols --including key exchange protocols -- are not implemented directly between the network devices. Software-Defined Networks utilize the controller based network design while maximizing the scalability that it provides.

The result is a significantly different trust model; rather than apply a peer-to-peer trust model, the network applies a device-to-controller trust model.

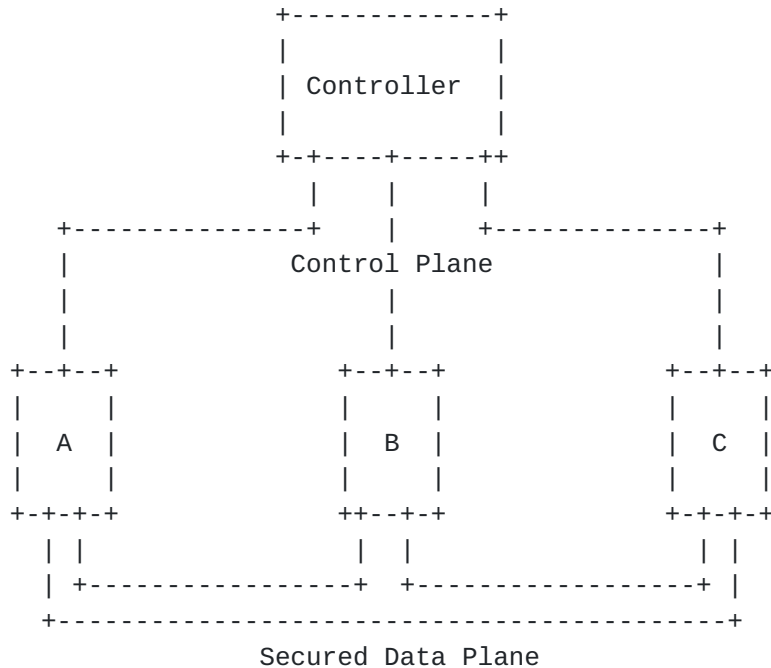


Figure 1: Controller-based Secured Communications

Figure 1 shows an example controller based network design. Three network devices (labeled A, B, and C) setup a protected control plane connection to a Controller. The Controller distributes policy to the network devices, which enable them to securely communicate between them in the data plane.

When a controller participates in a key exchange protocol, it is tempting to give it the task of generating and distributing session keys directly to network devices. However, such a design has several security considerations. Because the controller has all session keys, it can become an active participant or a passive monitor to the secured communications. Also, for scalability reasons controller-distributed session keys tend to be group keys, either a single group key or a set of group keys that devices use to protect communications between them.

Many key exchange methods (such as IKEv2) use a Diffie-Hellman (DH) algorithm to derive keys. When combined with an authentication method, the key exchange method allows two network devices to generate private pair-wise keys with each other. This document presents a key exchange method making use of the device-to-controller trust model, where a controller is used to distribute keying material

and policy between network devices, also resulting in the devices generating private pair-wise keys with each other. DH public values are provided to controllers from IPsec devices, where the controller relays the DH public values to authorized peers of that IPsec device as defined by a centralized policy. Network devices then create and install private pair-wise IPsec session keys to be used to secure communications with their peers.

Controller-based key exchange methods can be used to create a Gateway-to-Gateway VPN [[RFC7018](#)] in either a Full-Mesh Topology or Dynamic Full-Mesh Topology.

Although IKEv2 is not used in this approach, interfaces between IKEv2 and IPsec defined in [RFC 7296](#) are maintained as much as possible.

1.1. Requirements Language

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.

2. Overview

A controller acts as a proxy relaying policy and keying material between IPsec devices. All algorithms are selected by the controller, for example by a management station. The combination of a Controller and a set of IPsec devices comprises a cooperating group of devices that make up a VPN, where each IPsec device is authorized to communicate with other IPsec devices in the group (either all devices, or a subset depending on policy).

DH public values are distributed to the controller from each IPsec device and redistributed from the controller to each IPsec device. Each IPsec device creates and maintains a DH pair, which it uses to communicate with other members of the VPN. This distribution of DH public values (and other related values) is intended to be embedded into an existing network device/controller protocol. In particular, Controller-IKE provides a mechanism for secure key management and only key management. It does not provide policy information or configuration as that is assumed to be provided by the controller. One such protocol is being developed at this time in the IETF I2NSF working group.

3. Generating Initial IPsec SAs

When an IPsec device begins operation, it generates a DH pair, using an algorithm defined in the IKEv2 Diffie-Hellman Group Transform IDs [[IKEV2-IANA](#)]. If the device does not have any active peers it simply distributes its DH public value to the Controller, along with a nonce used during SA creation. Whenever a DH pair is created, a new nonce MUST also be created. Whenever DH public values are transmitted, they are transmitted with the corresponding nonce. Whenever a DH private or DH public value is used, it is used along with the corresponding nonce. However, in the diagrams and descriptions below, the nonces are often left out for simplicity and clarity sake.

Upon receiving a peer's DH public value and nonce, the receiver creates IPsec SAs (as described in [Section 5.2](#)). For each peer, a pair of IPsec SAs are created combining the IPsec device's own DH private value with the DH public number received from the Controller.

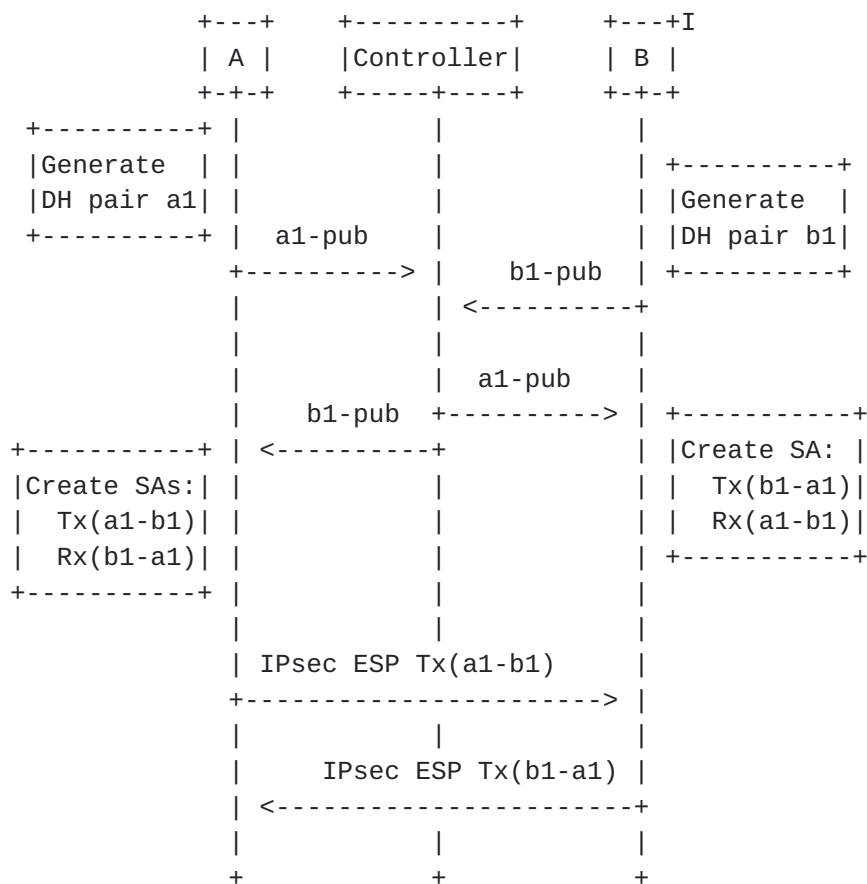


Figure 2: Generation of Initial IPsec SAs between two peers

Figure 2 shows IPsec SA generation between a pair of IPsec devices. Two IPsec devices (A and B shown in Figure 1) join the network. Each

creates its own DH pair (labelled "a1" on A and "b1" on B), and distributes the DH public value (labelled a1-pub and b1-pub) to the Controller. The controller forwards the DH public value to all authorized peers, although for simplicity of exposition the figure only shows the two IPsec devices.

When each device receives the peer's DH public value, a pair of IPsec SAs are generated: one outbound and one inbound. As shown in the figure, A generates an outbound SA labeled Tx(a1-b1), representing that it has been generated using A's DH pair labeled a1 and B's DH pair labeled b1. B generates the same IPsec SA as an inbound SA, which is labeled Rx(a1-b1). Similarly, A generates an inbound IPsec SA labelled Rx(b1-a1), which is the same IPsec SA on B labelled Tx(b1-a1).

This process repeats on both A and B as they discover other IPsec devices with which they are authorized to communicate.

4. Rekey of IPsec SAs

Any IPsec device may initiate a rekey at any time. Common reasons to perform a rekey include a local time or volume based policy, or may be the result of a cipher counter mode Initialization Vector (IV) counter nearing its final value. The rekey process is performed individually for each remote peer. If rekeying is performed with multiple peers simultaneously, then the decision process and rules described in this rekey are performed independently for each peer.

A decision process choosing an outbound IPsec SA is followed when certain events occur, as described in the rules below. The same decision process is followed regardless of whether the device is performing a rekey or responding to a peer's rekey. The decision process is:

1. Determine the outbound SAs with the remote peer's most recently distributed DH public value.
2. Determine which of those outbound SAs are "live". A "live" outbound SA is one built from a DH value from the local peer for which it has observed inbound traffic using any SA based on the same local DH pair. This proves that the remote peer is prepared to receive traffic protected by that DH pair.
3. Choose the "live" outbound SA built from the local peer's most recent DH public value.

A rekey operation follows these four basic rules.

Rule 1 When an IPsec device needs to perform a rekey with a remote peer, it creates a new pair of IPsec SAs by combining the new DH private value with the peer's DH public values. If the remote peer is also in the midst of a rollover and its DH public value has already been received, then this may result in creating two sets of SAs: one pair with the remote peer's old DH public value, and one pair with the remote peer's new DH public value.

Rule 2 When an IPsec device receives a new remote peer's DH public value from the controller it creates and installs a new pair of IPsec SAs by combining the remote peer's new DH public value with its own current local DH private values. If both devices are in the midst of a rollover, this may result in creating two sets of SAs with the remote peer's new DH public value: one with the local old DH private value, and one with the local new DH private value. The outbound SA decision process is performed.

Rule 3 The first IPsec packet received by a rekeying IPsec device on an inbound SA using its new DH pair causes it to perform the outbound SA decision process. It may also shorten the lifetime of IPsec SAs using its own old DH pair that are shared with this peer, as they are no longer in use (other than the inbound SA might receive packets in transit).

Rule 4 The first IPsec packet received from a remote rekeying IPsec device using the remote peer's new DH pair allows the IPsec device to shorten the lifetime of IPsec SAs shared with this peer using unused remote DH pairs.

Two examples follow: a single IPsec device performing a rekey with its peers, and two IPsec devices performing a simultaneous rekey. The same rekey operations described above are exhibited in both cases.

4.1. Single IPsec Device Rekey

When a single IPsec device begins a rekey, it first generates a new DH pair and generates new IPsec SA pairs for each peer with which it is communicating. It does this by combining the new DH private value with each peer's existing DH public value. Only when the new IPsec SAs have been installed and the device is prepared to receive on those new SAs does it then distribute the new DH public value to the Controller, which forwards it to its authorized peers. It continues to transmit on the old SAs for each peer until it observes that peer begin to transmit on the new SAs.

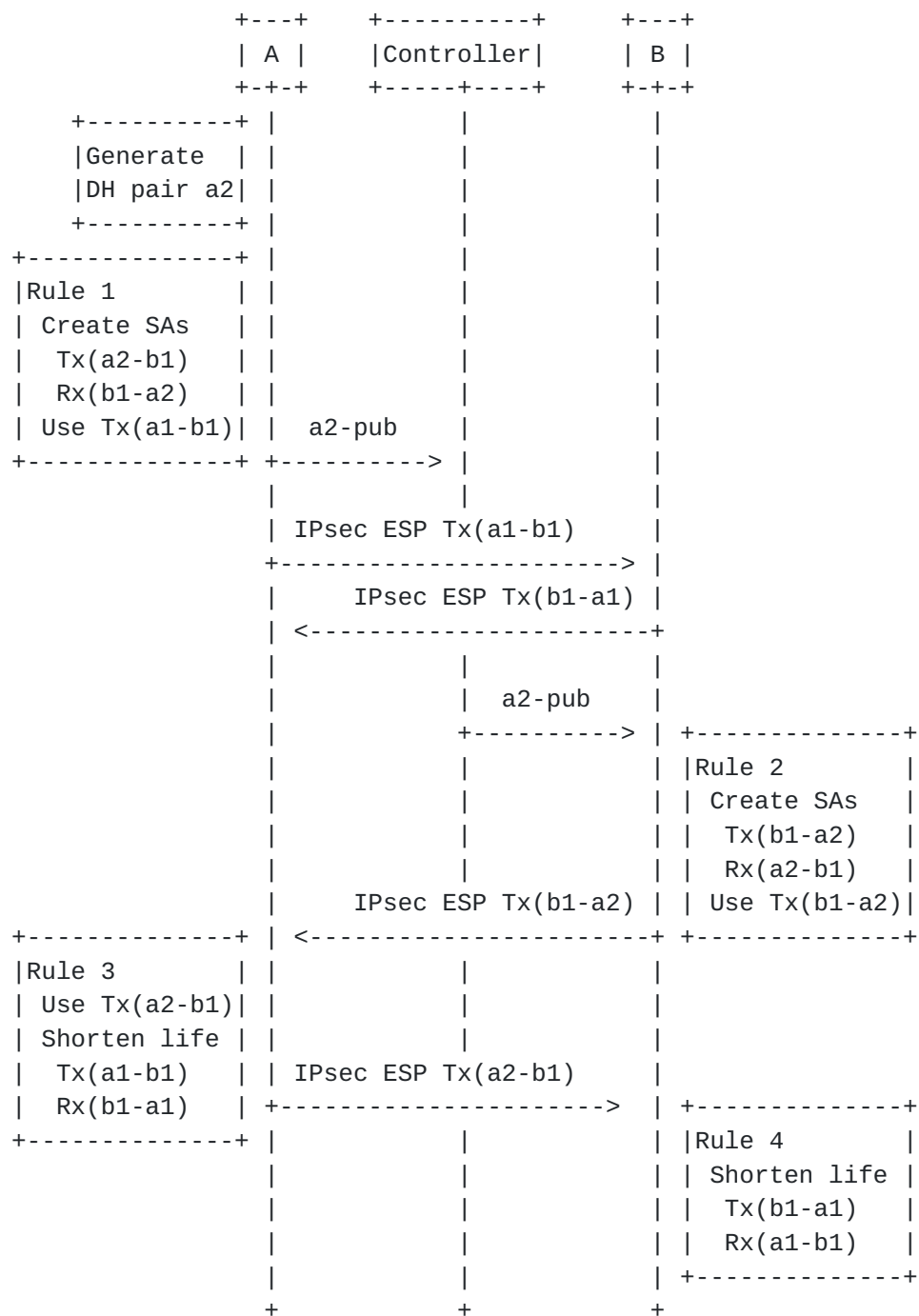


Figure 3: Single IPsec Device Rekey Protocol Flow

In Figure 3, device A is shown as performing a rekey, and it creates a DH pair labelled "a2". The following steps are followed.

1. Rule 1 requires creating new IPsec SAs for each peer. In this example, A creates a new outbound IPsec SA to communicate with B labelled Tx(a2-b1), and a new inbound IPsec SA labelled

Rx(b1-a2). A continues to transmit on Tx(a1-b1) (generated as shown in Figure 2).

2. A distributes the new public value (a2-pub) to the Controller who forwards it to A's authorized peers, which includes B. During this time, both A and B continue to use the initial IPsec SAs setup between them using a1 and b1.
3. When B receives a2 from the controller, B follows Rule 2 by creating Tx(b1-a2), Rx(a2-b1). B also follows the outbound SA decision process, which causes it to change its outbound IPsec SA to A to Tx(b1-a2).
4. When A receives a packet protected by Rx(b1-a2), it follows Rule 3 and performs the outbound SA decision process. This causes it to change its outbound IPsec SA to Use Tx(a2-b1). It also optionally shortens the lifetime of the old IPsec SAs shared with this peer.
5. When B receives a packet protected by Tx(a2-b1), it follows Rule 4, in which it may shorten the lifetime of the old IPsec SAs shared with this peer using DH pairs that are no longer in use.

At the end of the rekey, both A and B retain a single DH pair, and a single set of IPsec SAs between them.

4.2. Simultaneous Rekey of IPsec Devices

When two or more IPsec device simultaneously begin a rekey, they each follow the rekeying method described in the previous section. Every rekeying IPsec device generates a new DH pair and generates new IPsec SA pairs for each peer with which it is communicating by combining their new DH private value with each peer's existing DH public value. When this completes on a particular IPsec device, it distributes the new DH public value to the Controller, which forwards it to its authorized peers. Each continues to transmit on the existing SAs for each peer until it observes that peer transmitting on the new SAs. During a simultaneous rekey up to four pairs of IPsec SAs may be temporarily created, but the four rules ensure that they converge on a single new set of IPsec SAs.

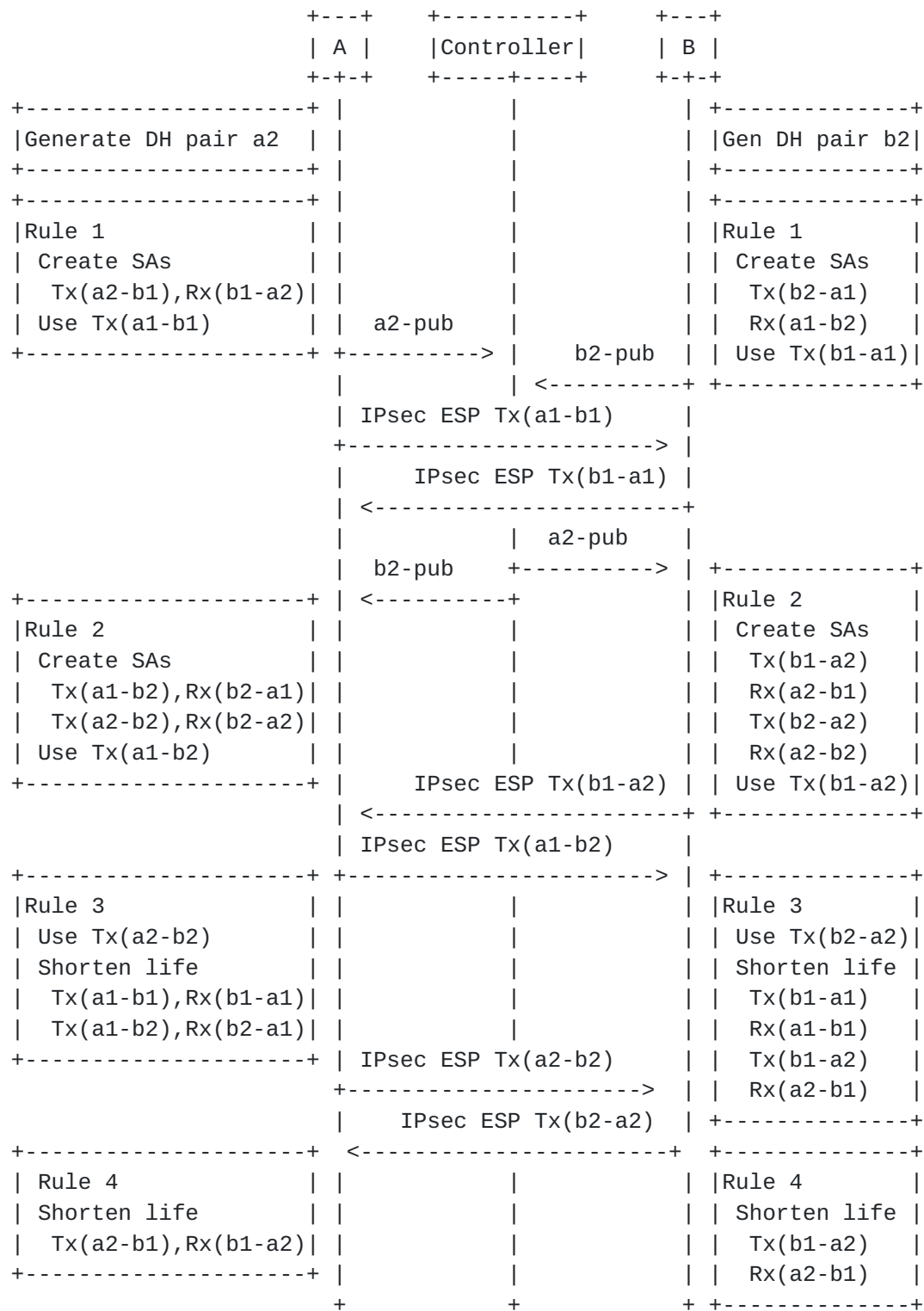


Figure 4: Simultaneous IPsec Device Rekey Protocol Flow

In Figure 4, device A and device B are both shown as performing a rekey. Their initial state corresponds to the final state shown in

Figure 2 (i.e., they are communicating using a single pair of IPsec SAs created from DH pairs "a1" and "b1").

1. A and B follow Rule 1, which includes creating new IPsec SAs for each peer. In this example, A creates a new outbound IPsec SA to communicate with B labelled Tx(a2-b1), and a new inbound IPsec SA labelled Rx(b1-a2). B creates a new outbound IPsec SA to communicate with A labelled Tx(a1-b2), and a new inbound IPsec SA labelled Rx(b2-a1). A and B continue to transmit on IPsec SAs previously created from DH pairs "a1" and "b1".
2. A distributes the new public value (a2-pub) to the Controller who forwards it to A's authorized peers, which includes B. B also distributes the new public value (b2-pub) to the Controller who forwards it to B's authorized peers, which includes A.
3. When A and B receive each other's new peer DH public value from the controller they follow Rule 2. But because now there are four DH values that could be in used between A and B, they must be prepared to use IPsec SAs using each permutation of DH values: a1-b1, a1-b2, a2-b1, a2-b2. Prior to implementing Rule 2, each has already created sets of IPsec SAs matching two of the permutations, so just two more sets must be generated during Rule 2.
 - * One pair is created using the IPsec device's old DH pair with the peer's new DH pair. This is necessary, because the peer may transmit on this pair.
 - * One pair is created using the IPsec device's new DH pair with the peer's new DH pair. This is the set of IPsec SAs that will be used at the end of the rekey process.

Each peer begins transmitting on an IPsec SA that combines the remote peer's new DH pair and its own old DH pair, which is the most recent "live" SA on which it can transmit. I.e., A begins transmitting on Tx(a1-b2) and B begins transmitting on Tx(b1-a2).

4. When A receives a packet protected by Rx(b1-a2), it understands that the remote peer has received its new DH public value. A also understands that because of Rule 2 that B must have created IPsec SAs using a2-b2. This allows A to follow Rule 3 and change its outbound IPsec SA to Use Tx(a2-b2). Similarly, when B receives a packet protected by Rx(a1-b2), B recognizes that it can also begin to transmit using Tx(b2-a2). Note that it is also possible that A will receive a packet protected by Rx(b2-a2) or B will receive a packet protected by Rx(a2-b2), and then knows it can transmit on an IPsec SA using both of the new DH pairs.

5. Also in Rule 3, Both A and B optionally shorten the lifetime of older IPsec SAs shared with this peer derived from unused DH pairs to be cleaned up. A shortens the lifetime of SAs based on a1. B shortens the lifetime of SAs based on b1.
6. When A and B receive a packet protected by the remote peer's latest DH pair, they shorten the lifetime of SAs based on the remote peer's unused DH pair.

5. IPsec Database Generation

The PAD, SPD, and SAD all need to be setup as defined in the IPsec Security Architecture [[RFC4301](#)].

5.1. The Security Policy Database (SPD)

The SPD is implemented using methods outside the scope of this document. The SPD describes the type of traffic that will be protected between IPsec devices and the policy (e.g., ciphers) used to create SAs.

5.2. Security Association Database (SAD)

The SAD is constructed from IPsec policy (e.g., ciphers) obtained (depending on the controller protocol method) either from the controller or distributed by a peer (see [Section 6](#)).

Keying Material is generated following the method defined in IKEv2, and depends on SPIs, nonces, and the Diffie-Hellman shared secret.

The following sections describe how the necessary values are determined.

5.2.1. Generating Keying Material for IPsec SAs

5.2.1.1. g^{air}

A DH public value is distributed from the peer.

A DH shared secret (g^{air}) is computed using the peer's public value, and the device's private value. The DH group to be used must be known by the device. Options include distribution by an SDN controller, or distribution by the peer with the DH public value (see [Section 6](#)).

5.2.1.2. Nonces

Nonces are distributed with a DH public value, and are used only with that value. It is RECOMMENDED that nonces are generated as described in [Section 2.10 of \[RFC7296\]](#).

IKEv2 Key derivation specifies an initiator's nonce (Ni) and a responder's nonce (Nr). While neither peer is truly initiating a session), in order to fit the IKE key material models the roles must be assigned. The initiator is chosen as the peer with the larger nonce and the responder is the peer with the smaller. This does mean that the roles can change for each rekey and for each SA within a rekey.

5.2.1.3. SPIs

SPI values that are unique to each generation of keying material need to be determined. While each peer could distribute its own inbound SA value, the SPI value would be used by many peers. Although this is not a problem for an SA lookup (lookup can include the source and destination IP addresses), experience has shown that this is sub-optimal for some hardware SA lookup algorithms. Instead, this specification proposes generating values that are unpredictable and indistinguishable from randomly-generated SPI values.

SPI values are generated using the IKEv2 prf+ function, where nonces are used as the input to the prf. However, in some circumstances additional values may also need to be added to the prf for peers to ensure that they have implemented the same policy. [Appendix A.3.1](#) includes a discussion of when this might be needed.

$$\{\text{SPI}_i \mid \text{SPI}_r\} = \text{prf}+(\text{Ni} \mid \text{Nr}, \text{"SPI generation"})$$

The SPI_i is taken by the device providing Ni, where SPI_r is taken by the other device.

Because a device is not choosing its inbound SPI, its SA lookup process needs to be aware that duplicates could occur. In that case, the inbound SA Lookup SHOULD include a source IP addresses in addition to the SPI value (see [Section 4.1 of \[RFC4301\]](#)).

5.2.1.4. IPsec key generation

As described in previous sections, a DH public value and a nonce are distributed by peers. These are used to generate IPsec keys following the method defined in the IKEv2. SKEYSEED is generated following [Section 2.14 of \[RFC7296\]](#):

$$\text{SKEYSEED} = \text{prf}(\text{Ni} \parallel \text{Nr}, g^{\text{air}})$$

KEYMAT can be similarly derived as defined by IKEv2 ([Section 2.17 of \[RFC7296\]](#)), although only SK_d is required to be generated (shown in [Section 2.14 of \[RFC7296\]](#)).

$$\text{SK_d} = \text{prf+}(\text{SKEYSEED}, \text{Ni} \parallel \text{Nr} \parallel \text{SPIi} \parallel \text{SPIr})$$
$$\text{KEYMAT} = \text{prf+}(\text{SK_d}, \text{Ni} \parallel \text{Nr})$$

However, with the simplification where only SK_d is generated, it can be observed that the derivation of SK_d could be skipped entirely, and an optimized derivation of KEYMAT could be as follows:

$$\text{KEYMAT} = \text{prf+}(\text{SKEYSEED}, \text{Ni} \parallel \text{Nr} \parallel \text{SPIi} \parallel \text{SPIr})$$

Note: A single specification for generating KEYMAT will be determined in a future version of this document.

5.3. Peer Authorization Database (PAD)

The PAD identifies authorized peers. PAD entries are either statically configured, or may be dynamically updated by the controller.

The PAD omits authentication data for each peer, because it has delegated authentication and authorization to the controller.

The controller protocol MUST be able to describe an identity that a receiver can match against its local PAD database, to ensure that the peer is an authorized peer.

6. Policy distributed by the Controller

An IPsec device distributes to a controller a DH public value and the associated information and policy needed to create IPsec SAs in a Device Information Message (DIM). The controller then distributes the DIM to all authorized peers of that device. The following data elements MUST be embedded in a DIM message:

- o DH public number (used for key computation)
- o Nonce (used for key computation and SPI generation)
- o Peer identity (used to identify a peer in the PAD)
- o An Indication whether this is the initial distributed policy.

In cases where a single fixed IPsec policy has been pre-distributed, it is not necessary for the peer to send or receive that policy in a DIM. However, in cases where an IPsec device needs to indicate the policy it is willing to use, the following data elements SHOULD be included in a DIM:

- o A rekey counter, which increases for each unique DIM sent
- o An IPsec policy or policies

[Appendix A](#) suggests different ways that this policy may be included in a controller protocol. This document does not define a normative protocol format, because the DIM very likely needs to be integrated into an existing controller protocol rather than be an independent key management protocol. However, the controller protocol MUST provide a strong authentication between the device and the controller, and integrity of the messages MUST be provided. Confidentiality of the messages SHOULD also be provided. It is important that the controller protocol be protected with algorithms that are at least as strong as the algorithms used to protect the IPsec packets.

[6.1.](#) IPsec policy negotiation

In many controller based networks, there is a single IPsec policy used by all devices and there is no need to redistribute or select policy details. However, in some circumstances, there may be a need to have multiple policy options. This could happen when a controller changes the policy and wants to smoothly migrate all devices to the new policy. Or it could happen if a network supports devices with different capabilities. In these cases, devices need to be able to choose the correct policy to use for each other device, and must do this without sending additional messages and without sending individual messages to each peer. When a device supports multiple policies, it MUST include those policies within the DIM. This is done by sending multiple distinct policies, in order of preference, where the first policy is the most preferred. The policy to use is selected by taking the receiver's list of policies (i.e., the list advertised by the device that generates N_r), starting with the first policy, compare against the initiator's (device that generates N_i) list, and choosing the first one found in common. The method conforms to the IKEv2 Cryptographic Algorithm Negotiation described in [Section 2.7 of \[RFC7296\]](#). (However, see additional discussion when IKEv2 payloads are used in [Appendix A.3.1](#)).

If there is no match, this indicates a controller configuration error. These devices MUST NOT establish new SAs until a DIM is received that does produce a match.

When a device supports more than one DH group, then a unique DH public number MUST be specified for each in order of preference. The selection of which DH group to use follows the same logic as Policy selection, using the receiver's list order until a match is found in the initiator's list.

7. Security Considerations

This document proposes that a device re-use an ephemeral Diffie-Hellman exponential with multiple peers. There are some known potential vulnerabilities to this approach, which can be mitigated by the device first validating a peer's public value to be a safe public value before combining its own private value with it. The tests which MUST be performed are described in [\[RFC6989\]](#). See [\[REUSE\]](#) for additional security considerations when reusing ephemeral Diffie-Hellman keys.

A controller acts as a "trusted third party", which asserts that a particular Diffie-Hellman public value is associated with a particular entity. A device receiving the public key is not required to validate the assertion.

A subverted controller can act as a "man-in-the-middle" between a pair of devices. The easiest attack would be for the attacker to adjust the routing for the desired traffic through a compromised gateway and directly observe the cleartext. It is also possible that a subverted controller could provide a device with a Diffie-Hellman public value that actually belongs to a compromised gateway rather than the intended gateway, but doing so does not seem to be necessary. Nonetheless, the attack of a subverted controller can be mitigated by having a device sign its Diffie-Hellman public value (e.g, as a CMS Signed data object), where the receiver validates the digital signature on the object. However, this adds significant processing cost to a rekey and does not fit the controller-based network architecture model.

A subverted IPsec device whose DH pair has been compromised would be vulnerable to all of its IPsec traffic using that DH pair being compromised. Assuming the use of strong DH algorithms (including quantum resistant algorithms as they become available), the compromise would most likely be due to the device itself being compromised. Such a compromised device is also vulnerable to a direct plaintext compromise.

PFS is achieved between rekey periods, as DH pairs are required to be generated independently. However, because a device uses the same long-term key to generate session key with multiple peers, there is no PFS between sessions within the same rekey period. To reduce key

exposure outside of a rekey period, when a connection is closed each endpoint MUST forget not only the keys used by the connection but also any information that could be used to recompute those keys. However, the DH private key value and the nonce distributed with it may be forgotten only once the last IPsec SA that uses the private key value is removed from the SAD and there is no chance that a new IPsec SA could be setup that requires the private key value.

If quantum resistance is considered to be an issue, the controller can distribute a PSK, which could be used to create the SK_d in the manner shown in [[I-D.ietf-ipsecme-gr-ikev2](#)].

8. IANA Considerations

This memo specifies no IANA actions.

9. Acknowledgements

Graham Bartlett provided many useful comments and suggestions.

10. References

10.1. Normative References

[IKEV2-IANA]

IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", February 2016, <<http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-8>>.

[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>>.

[RFC6989] Sheffer, Y. and S. Fluhrer, "Additional Diffie-Hellman Tests for the Internet Key Exchange Protocol Version 2 (IKEv2)", [RFC 6989](#), DOI 10.17487/RFC6989, July 2013, <<https://www.rfc-editor.org/info/rfc6989>>.

[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>>.

[10.2. Informative References](#)

- [I-D.ietf-ipsecme-qr-ikev2] Fluhrer, S., McGrew, D., Kampanakis, P., and V. Smyslov, "Postquantum Preshared Keys for IKEv2", [draft-ietf-ipsecme-qr-ikev2-04](#) (work in progress), July 2018.
- [REUSE] Menezes, A. and B. Ustaoglu, "On Reusing Ephemeral Keys In Diffie-Hellman Key Agreement Protocols", December 2008, <<http://www.cacr.math.uwaterloo.ca/techreports/2008/cacr2008-24.pdf>>.
- [RFC7018] Manral, V. and S. Hanna, "Auto-Discovery VPN Problem Statement and Requirements", [RFC 7018](#), DOI 10.17487/RFC7018, September 2013, <<https://www.rfc-editor.org/info/rfc7018>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", [RFC 7426](#), DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", [RFC 8192](#), DOI 10.17487/RFC8192, July 2017, <<https://www.rfc-editor.org/info/rfc8192>>.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", [RFC 8329](#), DOI 10.17487/RFC8329, February 2018, <<https://www.rfc-editor.org/info/rfc8329>>.

[Appendix A. Example Controller protocols](#)

This section contains suggestions of how a Controller protocol might distribute policy for the Controller-based IKE.

[A.1. Example: I2NSF](#)

IPsec devices described in this document could be implemented as an Network Security Function (NSF) in the I2NSF Framework [[RFC8329](#)]. An I2NSF Controller or NSF Manager could distribute a DIM as a new type

of I2NSF Policy Rule. A YANG configuration data model would need to be defined for this.

[A.2.](#) Example: Network Controller

Site-to-site networks (e.g., an L2VPN or L3VPN) often use a network controller to share networking state between routers. When those routers use IPsec to protect the communications between themselves, this same network controller could distribute DH public number and nonces as well. For example, when a BGP Route Reflector (RR) is used in a network, a new address family (AFI) could distribute the state necessary for a controller-based IPsec key exchange. The BGP session between BGP routers and the Route Reflector (RR) would need to at least be integrity protected from a man in the middle and SHOULD be protected for confidentiality to prevent identity leakage.

The controller protocol MUST provide for adequate synchronization of the state. For example, when IPsec devices are synchronized with a key management protocol it is often necessary for the protocol to indicate when a device has rebooted and thinks that it is contacting peers for the first time. This alerts peers to destroy earlier keying state that they may still believe is current.

One possible method for distributing a DIM within a controller protocol is to use a set of IKEv2 payloads. For example, when a single set of IPsec policy has been distributed to all IPsec devices by a configuration server then the following minimum required data elements can be distributed using the following IKEv2 payloads.

ID, [N(INITIAL_CONTACT),] KE, Ni

When initiating a rekey, the IPsec device need only distribute its new DH public number due to Rule 1. Existing peers receiving the new DH public number need not be re-told about the previous DH public number. Any new peer that receives and acts upon a "stale" controller message (containing the old DH public number) will still be able to setup IPsec SAs using the old DH public number, and can use them until the new DH public number is received.

[A.3.](#) Additional controller protocol considerations

If the controller protocol is more complicated, there are some additional considerations.

A.3.1. Peer-to-peer distribution of IPsec policy

In some cases an IPsec device may have more than one IPsec policy under which it is willing to communicate. This would result in an IPsec device using the decision process described in [Section 6.1](#) to determine which policy to use between itself and that peer. An IKEv2 SA payload could be used to distribute the policies, and the decision process could be used to determine which single set of policy is to be used. Note that the same decision process is followed by both peers. It is important that when an SA payload is used, that each proposal within the SA payload MUST contain at most a single transform for each Transform type (e.g., ENCR and (optionally) ESN for combined mode algorithms, ENCR, INTEG, and (optionally) ESN otherwise). This is due to the absence of a direct peer-to-peer reply message, which would alert the sender of which proposal was chosen.

1. Determine the Responder (as defined in [Section 5.2.1.2](#)).
2. Follow the negotiation rules defined in [Section 2.7](#) of IKEv2 [[RFC7296](#)] (with the restrictions that more than one transform of each type MUST NOT be present, and no error notifications are returned to the peer). Each peer will independently compare each Proposal in the Responder's SA payload to each Proposal in the Initiator's SA payload and choose the first match.
3. If there is no match, then this is considered a controller error, and the IPsec devices SHOULD report the error to the controller.

Payloads distributed in the controller protocol could be as follows:

ID, [N(INITIAL_CONTACT),] SA, KE, Ni

where the SA payload contains one or more Proposals, each of this includes a transform indicating the Diffie-Hellman group it is willing to use (D-H Transform), and IPsec transforms that it is willing to use (e.g., ENCR, INTEG, and ESN Transforms). The KE payload includes a DH public number matching the D-H Transform.

Because there is no direct peer-to-peer IKE messages, there is a need for peers to reliably know which Proposal in the SA payload was chosen. That is, if they do not reliably follow the same decision process they may end up installing IPsec SAs with incompatible policy. A straightforward method to verify that a peer has chosen the same policy is to include the SA Proposal number (SPN) from the SA payload in the SPI calculation.

$$\{SPI_i \mid SPI_r\} = prf+(Ni \mid Nr, "SPI\ generation" \mid SPNi \mid SPNr)$$

If a device is willing to use more than one DH group, then a single SA payload should be distributed, but the included Proposals may contain different D-H Transforms. A KE payload must be included for each D-H Group that is offered in the SA payload.

ID, [N(INITIAL_CONTACT),] [SA,] KE, [KE,] Ni

[A.3.2.](#) Ordering of messages distributed to a controller

A controller protocol may require a method of determining ordering of messages that are distributed, i.e. a Rekey Counter (RC). It is RECOMMENDED that the ordering be defined by a monotonically increasing counter value distributed with the IPsec policy. It is further RECOMMENDED that to ensure ordering after a device reboot that the counter include a "boot count", which increments after each reboot. For example, the counter could be a 64-bit counter where the high order 32 bits are a "boot count", followed by the counter that begins at 1 following the increment of the "boot count".

[Appendix B.](#) Differences between IKEv2 and Controller based methods

Placeholder appendix.

Authors' Addresses

David Carrel
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-525-7852
Email: carrel@cisco.com

Brian Weis
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-526-4796
Email: bew@cisco.com

