Internet Engineering Task Force
Internet-Draft
Intended status: Proposed Standard
Expires: 27 March 2023

N. Elkins
Inside Products, Inc.
M. Ackermann
BCBS Michigan
A. Deshpande
NITK Surathkal/Google
T. Pecorella
A. Rashid
University of Florence

23 September 2022

IPv6 Performance and Diagnostic Metrics Version 2 (PDMv2) Destination Option

draft-ietf-ippm-encrypted-pdmv2-02.txt

Abstract

RFC8250 describes an optional Destination Option (DO) header embedded in each packet to provide sequence numbers and timing information as a basis for measurements. As this data is sent in clear-text, this may create an opportunity for malicious actors to get information for subsequent attacks. This document defines PDMv2 which has a lightweight handshake (registration procedure) and encryption to secure this data. Additional performance metrics which may be of use are also defined.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Current Performance and Diagnostic Metrics (PDM)

The current PDM is an IPv6 Destination Options header which provides information based on the metrics like Round-trip delay and Server delay. This information helps to measure the Quality of Service (QoS) and to assist in diagnostics. However, there are potential risks involved transmitting PDM data during a diagnostics session.

PDM metrics can help an attacker understand about the type of machine and its processing capabilities. Inferring from the PDM data, the attack can launch a timing attack. For example, if a cryptographic protocol is used, a timing attack may be launched against the keying material to obtain the secret.

Along with this, PDM does not provide integrity. It is possible for a Man-In-The-Middle (MITM) node to modify PDM headers leading to incorrect conclusions. For example, during the debugging process using PDM header, it can mislead the person showing there are no unusual server delays.

1.2. PDMv2 Introduction

PDMv2 adds confidentiality, integrity and authentication to PDM.

PDMv2 consists of three kinds of flows:

- Primary to Primary

- Primary to Secondary

- Secondary to Secondary

These terms are defined in Section 3. Sample topologies may be found in Appendix 1.

This document describes the Secondary to Secondary protocol and security requirements. The Primary to Primary and Primary to Secondary protocol will be described in a subsequent document.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] .

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

3. Terminology

* Primary Client (PC): An authoritative node that creates cryptographic keys for multiple Secondary clients.

* Primary Server (PS): An authoritative node that creates cryptographic keys for multiple Secondary servers.

* Secondary Client (SC): An endpoint node which initiates a session with a listening port and sends PDM data. Connects to the Primary Client to get cryptographic key material.

* Secondary Server (SS): An endpoint node which has a listening port and sends PDM data. Connects to the Primary Server to get cryptographic key material.

Note: a client may act as a server (have listening ports).

* Symmetric Key (K): A uniformly random bitstring as an input to the encryption algorithm, known only to Secondary Clients and Secondary Servers, to establish a secure communication.

* Public and Private Keys: A pair of keys that is used in asymmetric cryptography. If one is used for encryption, the other is used for decryption. Private Keys are kept hidden by the source of the key pair generator, but Public Key is known to everyone. pkX (Public Key) and skX (Private Key). Where X can be, any client or any server.

* Pre-shared Key (PSK): A symmetric key. Uniformly random bitstring, shared between any client or any server or a key shared between an entity that forms client-server relationship. This could happen through an out-of band mechanism: e.g., a physical meeting or use of another protocol.

* Session Key: A temporary key which acts as a symmetric key for the whole session.

4. Protocol Flow

The protocol will proceed in 3 steps.

Step 1: Negotiation between Primary Server and Primary Client.

Step 2: Registration between Primary Server / Client and Secondary Server / Client

Step 3: PDM data flow between Secondary Client and Secondary Server

After-the-fact (or real-time) data analysis of PDM flow may occur by network diagnosticians or network devices. The definition of how this is done is out of scope for this document.

4.1. Registration Phase

4.1.1. Rationale of Primary and Secondary Roles

Enterprises have many servers and many clients. These clients and servers may be in multiple locations. It may be less overhead to have a secure location (ex. Shared database) for servers and clients to share keys. Otherwise, each client needs to keep track of the keys for each server.

Please view Appendix 1 for some sample topologies and further explanation.

4.1.2. Diagram of Registration Flow

```
          +-----------+                      +-----------+
          |Primary    |<====================>|Primary    |
          |Client (PC)|                      |Server (PS)|
          +-----+-----+                      +-----+-----+
               ||                                  ||
               ||                                  ||
   +-------------------------+         +-------------------------+
   | Secondary Clients(SC's) |         | Secondary Servers (SS's)|
   |                         |         |                         |
   | +----+ +----+   +----+  |         | +----+ +----+   +----+  |
   | |SC1 | |SC2 |.. |SC N|  |<=======>| |SS 1| |SS 2|.. |SS N|  |
   | +----+ +----+   +----+  |         | +----+ +----+   +----+  |
   |                         |         |                         |
   +-------------------------+         +-------------------------+
```

4.2. Primary Client - Primary Server Negotiation Phase

The two entities exchange a set of data to ensure the respective identities.

They use HPKE KEM to negotiate a "SharedSecret".

4.3. Primary Server / Client - Secondary Server / Client Registration Phase

The "SharedSecret" is shared securely:

- By the Primary Client to all the Secondary Clients under its control. The protocol to define this will be defined in a subsequent document.

- By the Primary Server to all the Secondary Servers under its control. The protocol to define this will be defined in a subsequent document.

4.4. Secondary Client - Secondary Server communication

Each Client and Server derive a "SessionTemporaryKey" by using HPKE KDF, using the following inputs:

- The "SharedSecret".

- The 5-tuple (SrcIP, SrcPort, DstIP, DstPort, Protocol) of the communication.

- A Key Rotation Index (Kri).

The Kri SHOULD be initialized to zero.

The server and client initialize (separately) a pseudo-random non-repeating sequence between 1 and 2^15-1. How to generate this sequence is beyond the scope of this document, and does not affect the rest of the specification. When the sequence is used fully, or earlier if appropriate, the sender signals the other party that a key change is necessary. This is achieved by flipping the "F bit" and resetting the PRSEQ. The receiver increments the Kri of the sender, and derives another SessionTemporaryKey to be used for decryption.

It shall be stressed that the two SessionTemporaryKeys used in the communication are never the same, as the 5-tuple is reversed for the Server and Client. Moreover, the time evolution of the respective Kri can be different. As a consequence, each entity must maintain a table with (at least) the following informations:

- Flow 5-tuple, Own Kri, Other Kri

An implementation might optimize this further by caching the OwnSessionTemporaryKey (used in Encryption) and OtherSessionTemporaryKey (used in Decryption).

5. Security Goals

As discussed in the introduction, PDM data can represent a serious data leakage in presence of a malicious actor.

In particular, the sequence numbers included in the PDM header allows correlating the traffic flows, and the timing data can highlight the operational limits of a server to a malicious actor. Moreover, forging PDM headers can lead to unnecessary, unwanted, or dangerous operational choices, e.g., to restore an apparently degraded Quality of Service (QoS).

Due to this, it is important that the confidentiality and integrity of the PDM headers is maintained. PDM headers can be encrypted and authenticated using the methods discussed in Section 5.4, thus ensuring confidentiality and integrity. However, if PDM is used in a scenario where the integrity and confidentiality is already ensured by other means, they can be transmitted without encryption or authentication. This includes, but is not limited to, the following cases:

a) PDM is used over an already encrypted medium (For example VPN tunnels).

b) PDM is used in a link-local scenario.

c) PDM is used in a corporate network where there are security measures strong enough to consider the presence of a malicious actor a negligible risk.

5.1. Security Goals for Confidentiality

PDM data must be kept confidential between the intended parties, which includes (but is not limited to) the two entities exchanging PDM data, and any legitimate party with the proper rights to access such data.

5.2. Security Goals for Integrity

PDM data must not be forged or modified by a malicious entity. In other terms, a malicious entity must not be able to generate a valid PDM header impersonating an endpoint, and must not be able to modify a valid PDM header.

5.3. Security Goals for Authentication

An unauthorized party must not be able to send PDM data and must not be able to authorize another entity to do so. The protocol to define this will be defined in a subsequent document. Alternatively, if authentication is done via any of the following, this requirement may be seen to be met.

a) PDM is used over an already authenticated medium (For example, TLS session).

b) PDM is used in a link-local scenario.

c) PDM is used in a corporate network where security measures are strong enough to consider the presence of a malicious actor a negligible risk.

5.4. Cryptographic Algorithm

Symmetric key cryptography has performance benefits over asymmetric cryptography; asymmetric cryptography is better for key management. Encryption schemes that unite both have been specified in [RFC1421], and have been participating practically since the early days of public-key cryptography. The basic mechanism is to encrypt the symmetric key with the public key by joining both yields. Hybrid public-key encryption schemes (HPKE) [RFC9180] used a different approach that generates the symmetric key and its encapsulation with the public key of the receiver.

Our choice is to use the HPKE framework that incorporates key encapsulation mechanism (KEM), key derivation function (KDF) and authenticated encryption with associated data (AEAD). These multiple schemes are more robust and significantly efficient than the traditional schemes and thus lead to our choice of this framework.

6. PDMv2 Destination Options

6.1. Destinations Option Header

The IPv6 Destination Options extension header [RFC8200] is used to carry optional information that needs to be examined only by a packet's destination node(s). The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header and is defined in RFC 8200 [RFC8200]. The IPv6 PDMv2 destination option is implemented as an IPv6 Option carried in the Destination Options header.

6.2. Metrics information in PDMv2

The IPv6 PDMv2 destination option contains the following base fields:

SCALEDTLR: Scale for Delta Time Last Received
SCALEDTLS: Scale for Delta Time Last Sent
GLOBALPTR: Global Pointer
PSNTP: Packet Sequence Number This Packet
PSNLR: Packet Sequence Number Last Received
DELTATLR: Delta Time Last Received
DELTATLS: Delta Time Last Sent

PDMv2 adds a new metric to the existing PDM [RFC8250] called the Global Pointer. The existing PDM fields are identified with respect to the identifying information called a "5-tuple".

The 5-tuple consists of:

SADDR: IP address of the sender
SPORT: Port for the sender
DADDR: IP address of the destination
DPORT: Port for the destination
PROTC: Upper-layer protocol (TCP, UDP, ICMP, etc.)

Unlike PDM fields, Global Pointer (GLOBALPTR) field in PDMv2 is defined for the SADDR type. Following are the SADDR address types considered:

a) Link-Local

b) Global Unicast

The Global Pointer is treated as a common entity over all the 5-tuples with the same SADDR type. It is initialised to the value 1 and increments for every packet sent. Global Pointer provides a measure of the amount of IPv6 traffic sent by the PDMv2 node.

When the SADDR type is Link-Local, the PDMv2 node sends Global Pointer defined for Link-Local addresses, and when the SADDR type is Global Unicast, it sends the one defined for Global Unicast addresses.

6.3. PDMv2 Layout

PDMv2 has two different header formats corresponding to whether the metric contents are encrypted or unencrypted. The difference between the two types of headers is determined from the Options Length value.

Following is the representation of the unencrypted PDMv2 header:

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Option Type  | Option Length | Vrsn  |     Reserved Bits     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |      Random Number          |f|   ScaleDTLR   |   ScaleDTLS   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                         Global Pointer                        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |      PSN This Packet          |     PSN Last Received         |
     |-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Delta Time Last Received    |     Delta Time Last Sent      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Following is the representation of the encrypted PDMv2 header:

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Option Type  | Option Length | Vrsn  |     Reserved Bits     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |      Random Number          |f|                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               :
     |                     Encrypted PDM Data                        :
     :                         (30 bytes)                            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

0x0F

8-bit unsigned integer. The Option Type is adopted from RFC 8250 [RFC8250].

Option Length

0x12: Unencrypted PDM

0x22: Encrypted PDM

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. The options length is used for differentiating PDM [RFC8250], unencrypted PDMv2 and encrypted PDMv2.

Version Number

0x2

4-bit unsigned number.

Reserved Bits

12-bits.

Reserved bits for future use. They are initialised to 0 for PDMv2.

Random Number

15-bit unsigned number.

This is a random number with as much entropy as desired by the implementation. The level of entropy should be clearly specified to the user.

Flag Bit

1-bit field.

The flag bit indicates that the sender has used a new _SessionTemporaryKey_ and the receiver should increment the Kri of the sender and derive the same new _SessionTemporaryKey_.

Scale Delta Time Last Received (SCALEDTLR)

8-bit unsigned number.

This is the scaling value for the Delta Time Last Sent (DELTATLS) field.

Scale Delta Time Last Sent (SCALEDTLS)

8-bit unsigned number.

This is the scaling value for the Delta Time Last Sent (DELTATLS) field.

Global Pointer

32-bit unsigned number.

Global Pointer is initialized to 1 for the different source address types and incremented monotonically for each packet with the corresponding source address type.

This field stores the Global Pointer type corresponding to the SADDR type of the packet.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned number.

This field is initialized at a random number and is incremented monotonically for each packet of the 5-tuple.

Packet Sequence Number Last Received (PSNLR)

16-bit unsigned number.

This field is the PSNTP of the last received packet on the 5-tuple.

Delta Time Last Received (DELTATLR)

16-bit unsigned integer.

The value is set according to the scale in SCALEDTLR.

Delta Time Last Received =
(send time packet n - receive time packet (n - 1))

Delta Time Last Sent (DELTATLS)

16-bit unsigned integer.

The value is set according to the scale in SCALEDTLS.

Delta Time Last Sent =
(receive time packet n - send time packet (n - 1))

7. Security Considerations

PDMv2 DOH can be used by an attacker to gather information about a victim (passive attack) or to force the victim to modify its operational parameters to comply with forged data (active attacks).

In order to mitigate these, it is important that the PDMv2 DOH is subject to:

1) Confidentiality and

2) Integrity

with respect to an attacker.

In the following we will refer to two different "groups", that can or cannot belong to the same operational and management domain:

1) Servers - implementing services.

2) Clients-devices willing to interact with the services offered by Servers.

We will assume, for the sake of generalization, that the Servers are managed by an Organization (OrgA) implementing management procedures over them, and the Clients by a different Organization (OrgB).

An attacker could be in the following positions:

1) External to OrgA or OrgB.

2) Inside OrgA (i.e., a Server), either because it is a legitimate-but-curious device, or as a consequence of an attack to a device.

3) Inside OrgB (i.e., a Client), either because it is a legitimate-but-curious device, or as a consequence of an attack to a device

Furthermore, since PDMv2 DOH encryption could consume resources (albeit limited), it is possible to foresee a call of DoS by resource exhaustion. Hence, it is relevant to consider a form of access control to verify that the Server and Client belong to OrgA and OrgB respectively. This could be a _delegated trust_.

In other terms, a Client could just want to verify that the Server belongs to OrgA, without actually verifying the identity of the Server.

The Authentication and Authorization of Clients and Servers is thus delegated to the respective Organizations. In other terms, we do not expect, or want, that a Client and a Server should be forced to verify the respective identities (Authentication) or the permissions to use PDMv2 (Authorization).

The simple knowledge of the secrets required by the flow is considered sufficient to enable PDMv2. On the opposite, an unsuccessful decryption MUST result in dropping the PDMv2 DOH without further processing or, if configured to do so, might lead to throttling, filtering, and/or logging the activity of the other entity (Client or Server).

The present document specifies a methodology to enable this delegated trust, along with the Confidentiality and Integrity requirements, in the PDMv2 DOH.

We assume that PS and PC have verified the respective identities and the authorization to enable PDMv2 DOH on a set of devices under their responsibility: Secondary Servers (SS) and Secondary Clients (SC).

PS-PC

* Perform a HPKE KEM and obtain a PairMasterSecret (PMS).

* The PMS is stored securely in both PS and PC, and is NOT to be leaked.

* The PMS is valid only for the PC-PS pair.

In other terms, if a PS would want to establish a pair with two PCs, it will have two different PMSs.

* PMS might be re-negotiated after a given amount of time [renegotiation TBD]

* PS and PC exchange respectively the list of the SS and SC enabled to use PDMv2. The list can be:

  - A range of IP addresses, e.g.: 2001:db8:food:beef:cafe::0/80

  - A list of IP addresses, e.g., [2001:db8:food::1/128, 2001:db8:food::1/128]

Note:

1) How to represent the list in a compact way is out of scope of the present document,

2) The list could be dynamically updated.

3) Inside OrgB (i.e., a Client), either because it is a legitimate-but-curious device, or as a consequence of an attack to a device

- PS sends to the PC the Security Mode of Operation (SecMoP) to be used, see below.

PS-SS and PC-SC

- Each Secondary Sever (or Client) MUST authenticate itself with the Primary Server (or Client). This is out of scope of the present specification.

- Each SS receives a PairServerSecret (PSS), derived using HPKE KDF, and valid for the specific SS and the list of SCs defined above.

- Each SC receives a PairClientSecret (PCS), derived using HPKE KDF, and valid for the specific SC and the list of SSs defined above.

Since there are multiple use-cases, we define 4 modes of operations:

- *No Protection*: The Secrets are discarded (or not even created), and the flows do not use PDMv2. The scheme above is used only to disseminate the list of Secondary Clients and Secondary Servers. By sharing lists, this mode act as ACL (Access Control List) or authorization of the secondaries.

- *TrustedServers*: The Secondary Servers are trusted, and they do know a secret derived by the PMS.

- *AsymmetricPoll*: One Secondary (Server or Client) must acquire a secret from the respective Primary.

- *Identity Based Cryptography (IBC)*: IBC (RFC5091) is used to generate a shared secret between the SS and the SC.

The *TrustedServers* MoP has the benefit of requiring no additional steps to send and receive PDMv2 DOH, because each flow is protected by a SessionKey that can be derived autonomously by both the SC and the SS, without any interaction with the PS and PC, or any negotiation between the SS and the SC.

The possible vulnerabilities of the *TrustedServers* MoP are the following:

- Any SS can inspect the flows directed to a different SS in the same group.

- An attack to a SS might result in compromising the security of all the flows between all the clients and the Secondary Servers belonging to the same group.

A possible mitigation is to split the Secondary Servers in different sub-groups. This is a scenario similar to the one of a PC negotiating PDMv2 access with different PSs.

The *AsymmetricPoll* MoP has the benefit of isolating each SS and each SC. Only the SS and SC involved in a communication can decrypt their flows.

The *IBC* MoP has the same security properties of the *AsymmetricPoll* MoP, and the advantage of not requiring any interaction between the Primary and the Secondary. The disadvantage is the requirement of performing a "pairing" session negotiation between the Secondaries.

It must be considered that, while secure, this MoP could be used to perform a resource exhaustion attack on the PairDeviceKey establishment. Hence, a device MUST NOT reply to an IP address that is not in the Secondary[client, server] list, and MUST NOT reply with negative acknowledgments (e.g., in case of an incorrect decoding).

8. Privacy Considerations

PDMv2 greatly improves the privacy aspects of PDM by providing encryption.

9. IANA Considerations

Option Type to be assigned by IANA [RFC2780].

10. Contributors

The authors wish to thank NITK Surathkal for their support and assistance in coding and review. In particular Dr. Mohit Tahiliani and Abhishek Kumar (now with Google). Thanks also to Priyanka Sinha for her comments. Thanks to the India Internet Engineering Society (iiesoc.in), in particular Dhruv Dhody, for providing the funding for servers needed for protocol development.

11. References

11.1. References

11.2. 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>.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, DOI 10.17487/RFC2780, March 2000, <https://www.rfc-editor.org/info/rfc2780>.

[RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", RFC 8250, DOI 10.17487/RFC8250, September 2017, <https://www.rfc-editor.org/info/rfc8250>.

[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>.

11.3. Informative References

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/info/rfc9180>.

[RFC1421] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, DOI 10.17487/RFC1421, February 1993, <https://www.rfc-editor.org/info/rfc1421>.

Appendix A. Rationale for Primary Server / Primary Client

A.1. One Client / One Server

Let's start with one client and one server.

```
     +------------+  Derived Shared Secret  +------------+
     |   Client   |    ------------------>  |   Server   |
     +------+-----+                         +------+-----+
            |                                      |
            V                                      V
      Client Secret                         Server Secret
```

The Client and Server create public / private keys and derive a shared secret. Let's not consider Authentication or Certificates at this point.

What is stored at the Client and Server to be able to encrypt and decrypt packets? The shared secret or private key.

Since we only have one Server and one Client, then we don't need to have any kind of identifier for which private key to use for which Server or Client because there is only one of each.

Of course, this is a ludicrous scenario since no real organization of interest has only one server and one client.

A.2. Multiple Clients / One Server

So, let's try with multiple clients and one Primary server

```
     +------------+
     |  Client 1  |  --------+
     +------------+          |
     +------------+          +--->
     |  Client 2  |    ---------->    +------------+
     +------+-----+         :         |   Server   |
            :               :         +------+-----+
            :               +---->
     +------------+         |
     |  Client n  |  -------+
     +------+-----+
```

The Clients and Server create public / private keys and derive a shared secret. Each Client has a unique private key.

What is stored at the Client and Server to be able to encrypt and decrypt packets?

Clients each store a private key. Server stores: Client Identifier and Private Key.

Since we only have one Server and multiple Clients, then the Clients don't need to have any kind of identifier for which private key to use for which Server but the Server needs to know which private key to use for which Client. So, the Server has to store an identifier as well as the Key.

But, this also is a ludicrous scenario since no real organization of interest has only one server.

A.3. Multiple Clients / Multiple Servers

When we have multiple clients and multiple servers, then each not only does the Server need to know which key to use for which Client, but the Client needs to know which private key to use for which Server.

A.4. Primary Client / Primary Server

Based on this rationale, we have chosen a Primary Server / Primary Client topology.

Appendix B. Sample Implementation of Registration

B.1. Overall summary

In the Registration phase, the objective is to generate a shared secret that will be used in encryption and decryption during the Data Transfer phase. We have adopted a Primary-Secondary architecture to represent the clients and servers (see Section 4.1.1). The primary server and primary client perform Key Encapsulation Mechanism (KEM) [RFC9180] to generate a primary shared secret. The primary server shares this secret with secondary servers, whereas the primary client performs Key Derivation Function (KDF) [RFC9180] to share client-specific secrets to corresponding secondary clients. During the Data Transfer phase, the secondary servers generate the client-specific secrets on the arrival of the first packet from the secondary client.

B.2. High level flow

The following steps describe the protocol flow:

1. Primary client initiates a request to the primary server. The request contains a list of available ciphersuites for KEM, KDF, and AEAD.

2. Primary server responds to the primary client with one of the available ciphersuites and shares its public key.

3. Primary client generates a secret and its encapsulation. The primary client sends the encapsulation and a salt to the primary server. The salt is required during KDF in the Data Transfer phase.

4. Primary Server generates the secret with the help of the encapsulation and responds with a status message.

5. Primary server shares this key with secondary servers over TLS.

6. Primary client generates the client-specific secrets with the help of KDF by using the info parameter as the Client IP address. The primary client shares these keys with the corresponding secondary clients over TLS.

B.3. Commands used

Two commands are used between the primary client and the primary server to denote the setup and KEM phases. Along with this, we have a "req / resp" to indicate whether it's a request or response.

Between primary and secondary entities, we have one command to denote the sharing of the secret keys.

Appendix C. Change Log

Note to RFC Editor: if this document does not obsolete an existing RFC, please remove this appendix before publication as an RFC.

Appendix D. Open Issues

Note to RFC Editor: please remove this appendix before publication as an RFC.

Authors' Addresses

Nalini Elkins
Inside Products, Inc.
36A Upper Circle
Carmel Valley, CA, 93924
United States of America
Phone: +1 831 234 4232
Email: nalini.elkins@insidethestack.com

Michael Ackermann
BCBS Michigan
P.O. Box 2888
Detroit, Michigan, 48231
United States of America
Phone: +1 248 703 3600
Email: mackermann@bcbsm.com
URI: http://www.bcbsm.com

Ameya Deshpande
NITK Surathkal/Google
Pashan-Baner Link Road, Pashan
Pune, Maharashtra, 411021
India
Phone: +91 96893 26060
Email: ameyanrd@gmail.com
URI: https://www.nitk.ac.in/

Tommaso Pecorella
University of Florence
Dept. of Information Engineering, Via di Santa Marta, 3, 50139
Firenze
Italy
Phone: +39 055 2758540
Email: tommaso.pecorella@unifi.it
URI: https://www.unifi.it/

Adnan Rashid
University of Florence
Dept. of Information Engineering, Via di Santa Marta, 3, 50139
Firenze
Italy
Phone: +39 347 9821 467
Email: adnan.rashid@unifi.it
URI: https://www.unifi.it/