

Internet Engineering Task Force  
Internet-Draft  
Intended status: Proposed Standard  
Expires: 29 August 2022

N. Elkins  
Inside Products, Inc.  
M. Ackermann  
BCBS Michigan  
A. Deshpande  
NITK Surathkal  
T. Pecorella  
A. Rashid  
University of Florence  
25 February 2022

**IPv6 Performance and Diagnostic Metrics Version 2 (PDMv2) Destination  
Option  
draft-elkins-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

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 29 August 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

<a href="#">1.</a>	<a href="#">Introduction</a>	<a href="#">3</a>
<a href="#">1.1.</a>	<a href="#">Current Performance and Diagnostic Metrics (PDM)</a>	<a href="#">3</a>
<a href="#">1.2.</a>	<a href="#">PDMv2 Introduction</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Conventions used in this document</a>	<a href="#">3</a>
<a href="#">3.</a>	<a href="#">Terminology</a>	<a href="#">4</a>
<a href="#">4.</a>	<a href="#">Protocol Flow</a>	<a href="#">4</a>
<a href="#">4.1.</a>	<a href="#">Registration Phase</a>	<a href="#">5</a>
4.1.1.	<a href="#">Rationale of Primary (Writer) and Secondary (Reader) Roles</a>	<a href="#">5</a>
<a href="#">4.1.2.</a>	<a href="#">Diagram of Registration Flow</a>	<a href="#">5</a>
4.2.	<a href="#">Primary (Writer) Client - Primary (Writer) Server Negotiation Phase</a>	<a href="#">5</a>
4.3.	<a href="#">Primary (Writer) Server / Client - Secondary (Reader) Server / Client Registration Phase</a>	<a href="#">6</a>
4.4.	<a href="#">Secondary (Reader) Client - Secondary (Reader) Server communication</a>	<a href="#">6</a>
<a href="#">5.</a>	<a href="#">Security Goals</a>	<a href="#">7</a>
<a href="#">5.1.</a>	<a href="#">Security Goals for Confidentiality</a>	<a href="#">7</a>
<a href="#">5.2.</a>	<a href="#">Security Goals for Integrity</a>	<a href="#">7</a>
<a href="#">5.3.</a>	<a href="#">Security Goals for Authentication</a>	<a href="#">7</a>
<a href="#">5.4.</a>	<a href="#">Cryptographic Algorithm</a>	<a href="#">8</a>
<a href="#">6.</a>	<a href="#">PDMv2 Destination Options</a>	<a href="#">8</a>
<a href="#">6.1.</a>	<a href="#">Destinations Option Header</a>	<a href="#">8</a>
<a href="#">6.2.</a>	<a href="#">Metrics information in PDMv2</a>	<a href="#">8</a>
<a href="#">6.3.</a>	<a href="#">PDMv2 Layout</a>	<a href="#">9</a>
<a href="#">7.</a>	<a href="#">Security Considerations</a>	<a href="#">12</a>
<a href="#">8.</a>	<a href="#">Privacy Considerations</a>	<a href="#">12</a>
<a href="#">9.</a>	<a href="#">IANA Considerations</a>	<a href="#">12</a>
<a href="#">10.</a>	<a href="#">Contributors</a>	<a href="#">12</a>
<a href="#">11.</a>	<a href="#">References</a>	<a href="#">12</a>
<a href="#">11.1.</a>	<a href="#">References</a>	<a href="#">12</a>
<a href="#">11.2.</a>	<a href="#">Normative References</a>	<a href="#">12</a>
<a href="#">11.3.</a>	<a href="#">Informative References</a>	<a href="#">13</a>
<a href="#">Appendix A.</a>	<a href="#">Rationale for Primary (Writer) Server / Primary (Writer) Client</a>	<a href="#">13</a>
<a href="#">A.1.</a>	<a href="#">One Client / One Server</a>	<a href="#">13</a>
<a href="#">A.2.</a>	<a href="#">Multiple Clients / One Server</a>	<a href="#">14</a>



<a href="#">A.3.</a>	Multiple Clients / Multiple Servers . . . . .	<a href="#">15</a>
<a href="#">A.4.</a>	Primary (Writer) Client / Primary (Writer) Server . . . . .	<a href="#">15</a>
<a href="#">Appendix B.</a>	Sample Implementation of Registration . . . . .	<a href="#">15</a>
<a href="#">B.1.</a>	Overall summary . . . . .	<a href="#">15</a>
<a href="#">B.2.</a>	High level flow . . . . .	<a href="#">15</a>
<a href="#">B.3.</a>	Commands used . . . . .	<a href="#">16</a>
<a href="#">Appendix C.</a>	Change Log . . . . .	<a href="#">16</a>
<a href="#">Appendix D.</a>	Open Issues . . . . .	<a href="#">16</a>
	Authors' Addresses . . . . .	<a href="#">16</a>

## **[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 introduces confidential, integrity and authentication.

TBD

## **[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 (Writer) Client (WC): An authoritative node that creates cryptographic keys for multiple reader clients.
- \* Primary (Writer) Server (WS): An authoritative node that creates cryptographic keys for multiple reader servers.
- \* Secondary (Reader) Client (RC): An endpoint node which initiates a session with a listening port and sends PDM data. Connects to the Primary (Writer) Client to get cryptographic key material.
- \* Secondary (Reader) Server (RS): An endpoint node which has a listening port and sends PDM data. Connects to the Primary (Writer) 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 (Reader) Clients and Secondary (Reader) 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 (Writer) Server and Primary (Writer) Client.

Step 2: Registration between Primary (Writer) Server / Client and Secondary (Reader) Server / Client



Step 3: PDM data flow between Secondary (Reader) Client and Secondary (Reader) 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.

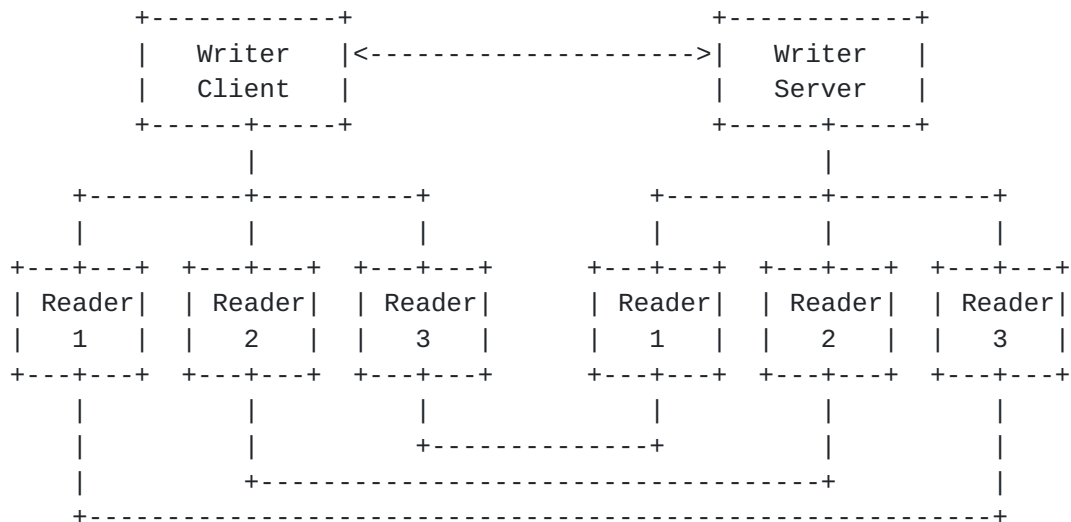
#### 4.1. Registration Phase

##### 4.1.1. Rationale of Primary (Writer) and Secondary (Reader) 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



#### 4.2. Primary (Writer) Client - Primary (Writer) 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 (Writer) Server / Client - Secondary (Reader) Server / Client Registration Phase**

The "SharedSecret" is shared securely:

- \* By the Primary (Writer) Client to all the Secondary (Reader) Clients under its control. How this is achieved is beyond the scope of the present specification.
- \* By the Primary (Writer) Server to all the Secondary (Reader) Servers under its control. How this is achieved is beyond the scope of the present specification.

#### **4.4. Secondary (Reader) Client - Secondary (Reader) 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 [X], 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**

TBD



## **[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:

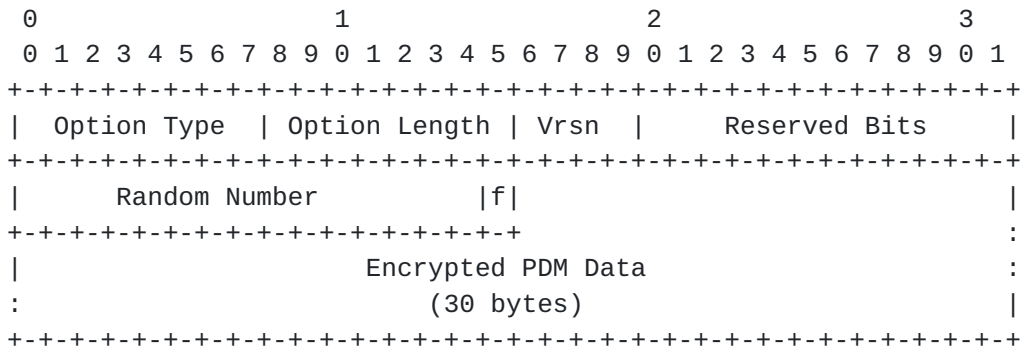








Following is the representation of the encrypted PDMv2 header:



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.

TBD

Flag Bit

1-bit field.

TBD

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 Recieved (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**

TBD

## **8. Privacy Considerations**

TBD

## **9. IANA Considerations**

This memo includes no request to IANA.

## **10. Contributors**

TBD

## **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>>.



[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**

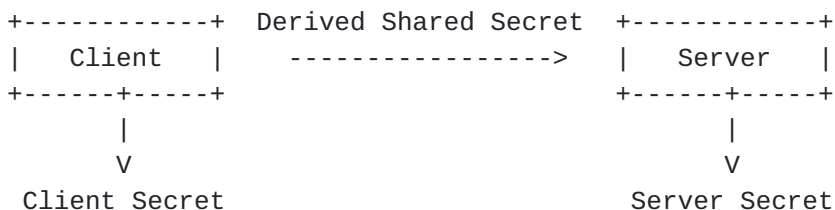
[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", [RFC 9180](#), DOI 10.17487/RFC9180, February 2022, <<https://www.rfc-editor.org/info/rfc9180>>.

[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 (Writer) Server / Primary (Writer) Client**

**[A.1](#). One Client / One Server**

Let's start with one client and one server.



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.



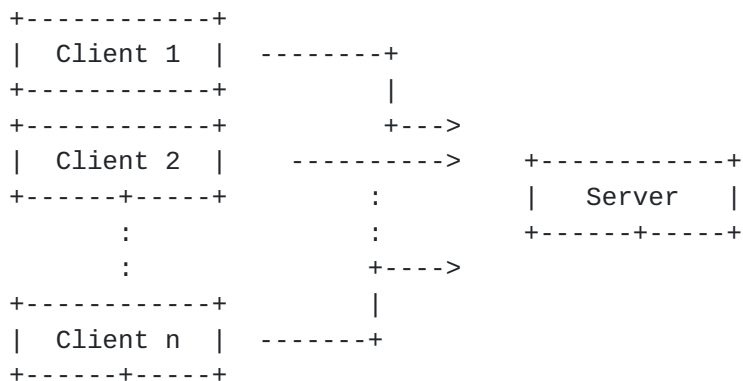


Since we only have one Server and one Client, then we don't need to have any kind of identifier for which private key to use for which Server or Client because there is only one of each.

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 (Writer) server



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 (Writer) Client / Primary (Writer) Server**

Based on this rationale, we have chosen a Primary (Writer) Server / Primary (Writer) 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](mailto: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](mailto:mackermann@bcbsm.com)  
URI: <http://www.bcbsm.com>



Ameya Deshpande  
NITK Surathkal  
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/>



