

Network Working Group
INTERNET-DRAFT
Category: Standards Track
<draft-dekok-saag-dhcp-keys-00.txt>
24 October 2016

DeKok, Alan
Network RADIUS SARL

DHCP Keys via 802.1X

Abstract

While RFC 3118 made provisions for securing DHCP, it made no provisions for creating or distributing authentication keys. This specification describes how in some cases, DHCP keys can be automatically derived from 802.1X authentication. The pros and cons of this approach are also discussed

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on May 24, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info/>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|---|
| 1. Introduction | 4 |
| 1.1. Problem Statement | 4 |
| 1.2. Proposed Solution | 4 |
| 1.3. Requirements Language | 4 |
| 2. Generating Keying Material | 5 |
| 2.1. Implementation Considerations | 6 |
| 2.2. What does the Signature mean? | 6 |
| 2.3. Open Questions | 6 |
| 3. Security Considerations | 7 |
| 4. IANA Considerations | 7 |
| 5. References | 8 |
| 5.1. Normative References | 8 |
| 5.2. Informative References | 8 |

1. Introduction

There has been increased interest in, and awareness of, securing basic networking protocols such as DHCP [RFC2131]. While provisions were made in [RFC3118] for securing the protocol via secret keys, there is little discussion on how the secret keys are created or managed. This specification addresses that issue, for the limited case of DHCP which occurs after 802.1X authentication.

1.1. Problem Statement

This document addresses the situation where a client machine connects to the network via 802.1X / EAP, and where keying material is derived as part of the EAP conversation. Once the client machine authenticated to the network, it requests an IP address via DHCP.

However, there is essentially no communication or interaction between the AAA server which authenticates the client machine, and the DHCP server which allocates IP addresses. This lack of communication means that it is possible to attack the systems independent. That is, the two systems do not work together to increase security.

1.2. Proposed Solution

Have the AAA server derive a shared secret key for signing DHCP packets. And then use that key to sign DHCP packets.

1.3. Requirements Language

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

2. Generating Keying Material

The algorithm used to generate the keying material is similar to that used for EAP methods, such as EAP-TLS ([RFC5216] Section 2.3), and TTLS ([RFC7542] Section 8).

Upon successful conclusion of an EAP-TTLS negotiation, 128 octets of keying material are generated and exported for use in securing the data connection between client and access point. The first 4 octets of the keying material constitute the secret ID, the last 124 octets constitute the DHCP shared secret key.

The keying material is generated using the TLS PRF function [RFC5246], with inputs consisting of the TLS master secret, the ASCII-encoded constant string "dhcp keying material", the TLS client random, and the TLS server random. The constant string is not null-terminated.

```
Keying Material = PRF-128(SecurityParameters.master_secret, "dhcp
                          keying material", SecurityParameters.client_random +
                          SecurityParameters.server_random)
```

```
Secret ID = Keying Material [0..3]
```

```
DHCP Shared secret key = Keying Material [4..127]
```

We perhaps don't want to use the keying material directly in the Secret ID. Instead, maybe use the last 4 octets of the keying material? Which should give less information than the first 4 octets. Or, derive the secret ID from a different PRF? Or set it to a fixed ID, which indicates that the client is using this method for signing packets?

The lifetime of the key is the lifetime of the underlying authentication session. If the client machine re-authenticates, a new DHCP shared secret key is derived.

The lifetime of the key MUST be no longer than the lifetime of the underlying authentication session. That is, once the authentication session has expired, the client MUST discard the key along with the corresponding secret ID.

We should also do something useful with the RDM / Reply Detection fields.

2.1. Implementation Considerations

There has historically been little communication between DHCP servers and AAA servers. As such, there is no clear way for the AAA server to share the derived key with the DHCP server. We leave that problem for implementors to solve.

The DHCP server could receive a packet, and request the corresponding key from the AAA server. Or, it may hand the packet to the AAA server for verification and/or signing. Or, it may retrieve the key from a secure co-located database. Or, the AAA server may proactively inform the DHCP server that a key exists, along with the key's value.

All of these scenarios are possible, and it is difficult to recommend any one in particular. We can, however, make general security recommendations.

The keys are highly secret information. As such, any exchange of keys **MUST** be done in a secure manner. Keys **MUST NOT** be visible to any entity other than the DHCP server and AAA server. If keys are stored in a database, they **MUST** be encrypted with an encryption key known only to the DHCP server and AAA server.

2.2. What does the Signature mean?

While it is nice to sign a DHCP packet for security, it is not at all clear what is meant by the signature. The minimum we can say is that the signature means that the DHCP server has communicated with the AAA server, and obtained a copy of the key.

There is no way to know if the DHCP server is the correct one, or malicious, or under the control of an attacker. Though if that is true, there is no need for the DHCP server to obtain the key. It can just hand out any addresses it wants. And if you can compromise a DHCP server on the network, or run a rogue DHCP server, having signed packets is probably the least of your worries.

We suggest that the signature means (in the expected case), that the DHCP server is known to the AAA server, and is likely known to the network administrators, and is likely the correct DHCP server for the client to communicate with.

2.3. Open Questions

Q: Does this add security?

A: Perhaps. A larger discussion and analysis is necessary. It does

not appear to reduce security. i.e. forging the key is difficult to impossible. The most that can be done is to disable this mechanism, which reduces DHCP to it's current level of security.

Q: How does the AAA server indicate to the DHCP client that it has this capability?

A: There is no way for it to do so. The DHCP client can just sign packets speculatively.

Q: How does the DHCP server know to use the key?

A: RFC3118 has provisions for the client signalling that it has a key.

Q: What does a DHCP server do when it receives a message from the client indicating that the client has a key, but the DHCP server does not have the key?

A: RFC 3118 is silent on this topic. The likeliest response is for the DHCP server to ignore the signature.

Q: How many networks are technically capable of using 802.1X?

A: It's 2016, pretty much all of them.

Q: How many networks are administratively capable of using 802.1X?

A: Some. Not so much for home networks, WiFi hot spots, etc. Most enterprises, telcos (3G), and WiFi systems with Hotspot 2.0 should be capable.

Q: What is the cost of implementing this?

A: Unknown. Some modifications to AAA / DHCP servers. And communication between them via some method to be determined.

3. Security Considerations

This specification is concerned entirely with security. As such, additional security discussion is not necessary.

4. IANA Considerations

There are no IANA considerations for this document.

5. References

5.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC2131]

Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2865]

Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3118]

Droms R. (Ed) and Arbaugh W. (Ed), "Authentication for DHCP Messages", RFC 3118, June 2001.

5.2. Informative References

[RFC5216]

Simon, D., Aboba, B., and Hurst, R., "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.

[RFC5246]

Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5281]

Funk, P, and Blake-Wilson, S., "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008

[RFC7542]

DeKok, A., "The Network Access Identifier", RFC 7542, May 2015.

Acknowledgments

None at this time.

Authors' Addresses

Alan DeKok
Network RADIUS SARL
100 CentrepoinTE Drive
Suite 200

Ottawa, ON
K2G 6B1
Canada

Email: aland@networkradius.com

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: September 12, 2019

F. Gont
SI6 Networks / UTN-FRH
I. Arce
Quarkslab
March 11, 2019

Security and Privacy Implications of Numeric Identifiers Employed in
Network Protocols
draft-gont-predictable-numeric-ids-03

Abstract

This document performs an analysis of the security and privacy implications of different types of "numeric identifiers" used in IETF protocols, and tries to categorize them based on their interoperability requirements and the associated failure severity when such requirements are not met. It describes a number of algorithms that have been employed in real implementations to meet such requirements and analyzes their security and privacy properties. Additionally, it provides advice on possible algorithms that could be employed to satisfy the interoperability requirements of each identifier type, while minimizing the security and privacy implications, thus providing guidance to protocol designers and protocol implementers. Finally, it provides recommendations for future protocol specifications regarding the specification of the aforementioned numeric identifiers.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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.

This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Terminology | 4 |
| 3. Threat Model | 5 |
| 4. Issues with the Specification of Identifiers | 5 |
| 5. Timeline of Vulnerability Disclosures Related to Some Sample Identifiers | 6 |
| 5.1. IPv4/IPv6 Identification | 6 |
| 5.2. TCP Initial Sequence Numbers (ISNs) | 7 |
| 6. Protocol Failure Severity | 9 |
| 7. Categorizing Identifiers | 9 |
| 8. Common Algorithms for Identifier Generation | 11 |
| 8.1. Category #1: Uniqueness (soft failure) | 11 |
| 8.1.1. Simple Randomization Algorithm | 11 |
| 8.1.2. Another Simple Randomization Algorithm | 12 |
| 8.2. Category #2: Uniqueness (hard failure) | 13 |
| 8.3. Category #3: Uniqueness, constant within context (soft-failure) | 13 |
| 8.4. Category #4: Uniqueness, monotonically increasing within context (hard failure) | 14 |
| 8.4.1. Predictable Linear Identifiers Algorithm | 14 |
| 8.4.2. Per-context Counter Algorithm | 16 |
| 8.4.3. Simple Hash-Based Algorithm | 18 |
| 8.4.4. Double-Hash Algorithm | 20 |
| 8.4.5. Random-Increments Algorithm | 21 |
| 9. Common Vulnerabilities Associated with Identifiers | 23 |
| 9.1. Category #1: Uniqueness (soft failure) | 23 |
| 9.2. Category #2: Uniqueness (hard failure) | 23 |
| 9.3. Category #3: Uniqueness, constant within context (soft | |

| | |
|--|----|
| failure) | 23 |
| 9.4. Category #4: Uniqueness, monotonically increasing within context (hard failure) | 24 |
| 10. Security and Privacy Requirements for Identifiers | 25 |
| 11. IANA Considerations | 26 |
| 12. Security Considerations | 26 |
| 13. Acknowledgements | 26 |
| 14. References | 26 |
| 14.1. Normative References | 26 |
| 14.2. Informative References | 27 |
| Authors' Addresses | 31 |

1. Introduction

Network protocols employ a variety of numeric identifiers for different protocol entities, ranging from DNS Transaction IDs (TxIDs) to transport protocol numbers (e.g. TCP ports) or IPv6 Interface Identifiers (IIDs). These identifiers usually have specific properties that must be satisfied such that they do not result in negative interoperability implications (e.g. uniqueness during a specified period of time), and associated failure severities when such properties are not met, ranging from soft to hard failures.

For more than 30 years, a large number of implementations of the TCP/IP protocol suite have been subject to a variety of attacks, with effects ranging from Denial of Service (DoS) or data injection, to information leakage that could be exploited for pervasive monitoring [RFC7528]. The root of these issues has been, in many cases, the poor selection of identifiers in such protocols, usually as a result of an insufficient or misleading specification. While it is generally trivial to identify an algorithm that can satisfy the interoperability requirements for a given identifier, there exists practical evidence that doing so without negatively affecting the security and/or privacy properties of the aforementioned protocols is prone to error.

For example, implementations have been subject to security and/or privacy issues resulting from:

- o Predictable TCP sequence numbers
- o Predictable transport protocol numbers
- o Predictable IPv4 or IPv6 Fragment Identifiers
- o Predictable IPv6 IIDs
- o Predictable DNS TxIDs

Recent history indicates that when new protocols are standardized or new protocol implementations are produced, the security and privacy properties of the associated identifiers tend to be overlooked and inappropriate algorithms to generate identifier values are either suggested in the specification or selected by implementators. As a result, we believe that advice in this area is warranted.

This document contains a non-exhaustive survey of identifiers employed in various IETF protocols, and aims to categorize such identifiers based on their interoperability requirements, and the associated failure severity when such requirements are not met. Subsequently, it analyzes several algorithms that have been employed in real implementation to meet such requirements and analyzes their security and privacy properties, and provides advice on possible algorithms that could be employed to satisfy the interoperability requirements of each category, while minimizing the associated security and privacy implications. Finally, it provides recommendations for future protocol specifications regarding the specification of the aforementioned numeric identifiers.

2. Terminology

Identifier:

A data object in a protocol specification that can be used to definitely distinguish a protocol object (a datagram, network interface, transport protocol endpoint, session, etc) from all other objects of the same type, in a given context. Identifiers are usually defined as a series of bits and represented using integer values. We note that different identifiers may have additional requirements or properties depending on their specific use in a protocol. We use the term "identifier" as a generic term to refer to any data object in a protocol specification that satisfies the identification property stated above.

Failure Severity:

The consequences of a failure to comply with the interoperability requirements of a given identifier. Severity considers the worst potential consequence of a failure, determined by the system damage and/or time lost to repair the failure. In this document we define two types of failure severity: "soft" and "hard".

Hard Failure:

A hard failure is a non-recoverable condition in which a protocol does not operate in the prescribed manner or it operates with excessive degradation of service. For example, an established TCP connection that is aborted due to an error condition constitutes, from the point of view of the transport protocol, a hard failure,

since it enters a state from which normal operation cannot be recovered.

Soft Failure:

A soft failure is a recoverable condition in which a protocol does not operate in the prescribed manner but normal operation can be resumed automatically in a short period of time. For example, a simple packet-loss event that is subsequently recovered with a retransmission can be considered a soft failure.

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

3. Threat Model

Throughout this document, we assume an attacker does not have physical or logical device to the device(s) being attacked. We assume the attacker can simply send any traffic to the target devices, to e.g. sample identifiers employed by such devices.

4. Issues with the Specification of Identifiers

While assessing protocol specifications regarding the use of identifiers, we found that most of the issues discussed in this document arise as a result of one of the following:

- o Protocol specifications which under-specify the requirements for their identifiers
- o Protocol specifications that over-specify their identifiers
- o Protocol implementations that simply fail to comply with the specified requirements

A number of protocol implementations (too many of them) simply overlook the security and privacy implications of identifiers. Examples of them are the specification of TCP port numbers in [RFC0793], the specification of TCP sequence numbers in [RFC0793], or the specification of the DNS TxID in [RFC1035].

On the other hand, there are a number of protocol specifications that over-specify some of their associated protocol identifiers. For example, [RFC4291] essentially results in link-layer addresses being embedded in the IPv6 Interface Identifiers (IIDs) when the interoperability requirement of uniqueness could be achieved in other ways that do not result in negative security and privacy implications [RFC7721]. Similarly, [RFC2460] suggests the use of a global counter

for the generation of Fragment Identification values, when the interoperability properties of uniqueness per {Src IP, Dst IP} could be achieved with other algorithms that do not result in negative security and privacy implications.

Finally, there are protocol implementations that simply fail to comply with existing protocol specifications. For example, some popular operating systems (notably Microsoft Windows) still fail to implement randomization of transport protocol ephemeral ports, as specified in [RFC6056].

5. Timeline of Vulnerability Disclosures Related to Some Sample Identifiers

This section contains a non-exhaustive timeline of vulnerability disclosures related to some sample identifiers and other work that has led to advances in this area. The goal of this timeline is to illustrate:

- o That vulnerabilities related to how the values for some identifiers are generated and assigned have affected implementations for an extremely long period of time.
- o That such vulnerabilities, even when addressed for a given protocol version, were later reintroduced in new versions or new implementations of the same protocol.
- o That standardization efforts that discuss and provide advice in this area can have a positive effect on protocol specifications and protocol implementations.

5.1. IPv4/IPv6 Identification

December 1998:

[Sanfilippo1998a] finds that predictable IPv4 Identification values can be leveraged to count the number of packets sent by a target node. [Sanfilippo1998b] explains how to leverage the same vulnerability to implement a port-scanning technique known as dumb/idle scan. A tool that implements this attack is publicly released.

November 1999:

[Sanfilippo1999] discusses how to leverage predictable IPv4 Identification to uncover the rules of a number of firewalls.

November 1999:

[Bellovin2002] explains how the IPv4 Identification field can be exploited to count the number of systems behind a NAT.

December 2003:

[Zalewski2003] explains a technique to perform TCP data injection attack based on predictable IPv4 identification values which requires less effort than TCP injection attacks performed with bare TCP packets.

November 2005:

[Silbersack2005] discusses shortcoming in a number of techniques to mitigate predictable IPv4 Identification values.

October 2007:

[Klein2007] describes a weakness in the pseudo random number generator (PRNG) in use for the generation of the IP Identification by a number of operating systems.

June 2011:

[Gont2011] describes how to perform idle scan attacks in IPv6.

November 2011:

Linux mitigates predictable IPv6 Identification values
[RedHat2011] [SUSE2011] [Ubuntu2011].

December 2011:

[I-D.ietf-6man-predictable-fragment-id-08] describes the security implications of predictable IPv6 Identification values, and possible mitigations.

May 2012:

[Gont2012] notes that some major IPv6 implementations still employ predictable IPv6 Identification values.

June 2015:

[I-D.ietf-6man-predictable-fragment-id-08] notes that some popular host and router implementations still employ predictable IPv6 Identification values.

5.2. TCP Initial Sequence Numbers (ISNs)

September 1981:

[RFC0793], suggests the use of a global 32-bit ISN generator, whose lower bit is incremented roughly every 4 microseconds. However, such an ISN generator makes it trivial to predict the ISN that a TCP will use for new connections, thus allowing a variety of attacks against TCP.

February 1985:

[Morris1985] was the first to describe how to exploit predictable TCP ISNs for forging TCP connections that could then be leveraged for trust relationship exploitation.

April 1989:

[Bellovin1989] discussed the security implications of predictable ISNs (along with a range of other protocol-based vulnerabilities).

February 1995:

[Shimomura1995] reported a real-world exploitation of the attack described in 1985 (ten years before) in [Morris1985].

May 1996:

[RFC1948] was the first IETF effort, authored by Steven Bellovin, to address predictable TCP ISNs. The same concept specified in this document for TCP ISNs was later proposed for TCP ephemeral ports [RFC6056], TCP Timestamps, and eventually even IPv6 Interface Identifiers [RFC7217].

March 2001:

[Zalewski2001] provides a detailed analysis of statistical weaknesses in some ISN generators, and includes a survey of the algorithms in use by popular TCP implementations.

May 2001:

Vulnerability advisories [CERT2001] [USCERT2001] are released regarding statistical weaknesses in some ISN generators, affecting popular TCP/IP implementations.

March 2002:

[Zalewski2002] updates and complements [Zalewski2001]. It concludes that "while some vendors [...] reacted promptly and tested their solutions properly, many still either ignored the issue and never evaluated their implementations, or implemented a flawed solution that apparently was not tested using a known approach". [Zalewski2002].

February 2012:

[RFC6528], after 27 years of Morris' original work [Morris1985], formally updates [RFC0793] to mitigate predictable TCP ISNs.

August 2014:

[I-D.eddy-rfc793bis-04], the upcoming revision of the core TCP protocol specification, incorporates the algorithm specified in [RFC6528] as the recommended algorithm for TCP ISN generation.

6. Protocol Failure Severity

Section 2 defines the concept of "Failure Severity" and two types of failures that we employ throughout this document: soft and hard.

Our analysis of the severity of a failure is performed from the point of view of the protocol in question. However, the corresponding severity on the upper application or protocol may not be the same as that of the protocol in question. For example, a TCP connection that is aborted may or may not result in a hard failure of the upper application: if the upper application can establish a new TCP connection without any impact on the application, a hard failure at the TCP protocol may have no severity at the application level. On the other hand, if a hard failure of a TCP connection results in excessive degradation of service at the application layer, it will also result in a hard failure at the application.

7. Categorizing Identifiers

This section includes a non-exhaustive survey of identifiers, and proposes a number of categories that can accommodate these identifiers based on their interoperability requirements and their failure modes (soft or hard)

| Identifier | Interoperability Requirements | Failure Severity |
|---------------|--|------------------|
| IPv6 Frag ID | Uniqueness (for IP address pair) | Soft/Hard (1) |
| IPv6 IID | Uniqueness (and constant within IPv6 prefix) (2) | Soft (3) |
| TCP SEQ | Monotonically-increasing | Hard (4) |
| TCP eph. port | Uniqueness (for connection ID) | Hard |
| IPv6 Flow L. | Uniqueness | None (5) |
| DNS TxID | Uniqueness | None (6) |

Table 1: Survey of Identifiers

Notes:

- (1)
While a single collision of Fragment ID values would simply lead to a single packet drop (and hence a "soft" failure), repeated collisions at high data rates might trash the Fragment ID space, leading to a hard failure [RFC4963].
- (2)
While the interoperability requirements are simply that the Interface ID results in a unique IPv6 address, for operational reasons it is typically desirable that the resulting IPv6 address (and hence the corresponding Interface ID) be constant within each network [I-D.ietf-6man-default-iids] [RFC7217].
- (3)
While IPv6 Interface IDs must result in unique IPv6 addresses, IPv6 Duplicate Address Detection (DAD) [RFC4862] allows for the detection of duplicate Interface IDs/addresses, and hence such Interface ID collisions can be recovered.
- (4)
In theory there are no interoperability requirements for TCP sequence numbers, since the TIME-WAIT state and TCP's "quiet time" take care of old segments from previous incarnations of the connection. However, a widespread optimization allows for a new incarnation of a previous connection to be created if the Initial Sequence Number (ISN) of the incoming SYN is larger than the last sequence number seen in that direction for the previous incarnation of the connection. Thus, monotonically-increasing TCP sequence numbers allow for such optimization to work as expected [RFC6528].
- (5)
The IPv6 Flow Label is typically employed for load sharing [RFC7098], along with the Source and Destination IPv6 addresses. Reuse of a Flow Label value for the same set {Source Address, Destination Address} would typically cause both flows to be multiplexed into the same link. However, as long as this does not occur deterministically, it will not result in any negative implications.
- (6)
DNS TxIDs are employed, together with the Source Address, Destination Address, Source Port, and Destination Port, to match DNS requests and responses. However, since an implementation knows which DNS requests were sent for that set of {Source Address, Destination Address, Source Port, and Destination Port, DNS TxID}, a collision of TxID would result, if anything, in a small performance penalty (the response would be discarded when it

is found that it does not answer the query sent in the corresponding DNS query).

Based on the survey above, we can categorize identifiers as follows:

| Cat # | Category | Sample Proto IDs |
|-------|--|----------------------------------|
| 1 | Uniqueness (soft failure) | IPv6 Flow L., DNS TxIDs |
| 2 | Uniqueness (hard failure) | IPv6 Frag ID, TCP ephemeral port |
| 3 | Uniqueness, constant within context (soft failure) | IPv6 IIDs |
| 4 | Uniqueness, monotonically increasing within context (hard failure) | TCP ISN |

Table 2: Identifier Categories

We note that Category #4 could be considered a generalized case of category #3, in which a monotonically increasing element is added to a constant (within context) element, such that the resulting identifiers are monotonically increasing within a specified context. That is, the same algorithm could be employed for both #3 and #4, given appropriate parameters.

8. Common Algorithms for Identifier Generation

The following subsections describe common algorithms found for Protocol ID generation for each of the categories above.

8.1. Category #1: Uniqueness (soft failure)

8.1.1. Simple Randomization Algorithm

```
/* Ephemeral port selection function */
id_range = max_id - min_id + 1;
next_id = min_id + (random() % id_range);
count = next_id;

do {
    if(check_suitable_id(next_id))
        return next_id;

    if (next_id == max_id) {
        next_id = min_id;
    } else {
        next_id++;
    }

    count--;
} while (count > 0);

return ERROR;
```

Note:

random() is a function that returns a pseudo-random unsigned integer number of appropriate size. Note that the output needs to be unpredictable, and typical implementations of POSIX random() function do not necessarily meet this requirement. See [RFC4086] for randomness requirements for security.

The function check_suitable_id() can check, when possible, whether this identifier is e.g. already in use. When already used, this algorithm selects the next available protocol ID.

All the variables (in this and all the algorithms discussed in this document) are unsigned integers.

8.1.2. Another Simple Randomization Algorithm

The following pseudo-code illustrates another algorithm for selecting a random identifier in which, in the event the identifier is found to be not suitable (e.g., already in use), another identifier is selected randomly:

```
id_range = max_id - min_id + 1;
next_id = min_id + (random() % id_range);
count = id_range;

do {
    if(check_suitable_id(next_id))
        return next_id;

    next_id = min_id + (random() % id_range);
    count--;
} while (count > 0);

return ERROR;
```

This algorithm might be unable to select an identifier (i.e., return "ERROR") even if there are suitable identifiers available, when there are a large number of identifiers "in use".

8.2. Category #2: Uniqueness (hard failure)

One of the most trivial approaches for achieving uniqueness for an identifier (with a hard failure mode) is to implement a linear function. As a result, all of the algorithms described in Section 8.4 are of use for complying the requirements of this identifier category.

8.3. Category #3: Uniqueness, constant within context (soft-failure)

The goal of this algorithm is to produce identifiers that are constant for a given context, but that change when the aforementioned context changes.

Keeping one value for each possible "context" may in many cases be considered too onerous in terms of memory requirements. As a workaround, the following algorithm employs a calculated technique (as opposed to keeping state in memory) to maintain the constant identifier for each given context.

In the following algorithm, the function F() provides (statelessly) a constant identifier for each given context.

```
/* Protocol ID selection function */
id_range = max_id - min_id + 1;

counter = 0;

do {
    offset = F(CONTEXT, counter, secret_key);
    next_id = min_id + (offset % id_range);

    if(check_suitable_id(next_id))
        return next_id;

    counter++;
} while (counter <= MAX_RETRIES);

return ERROR;
```

The function `F()` provides a "per-CONTEXT" constant identifier for a given context. 'offset' may take any value within the storage type range since we are restricting the resulting identifier to be in the range `[min_id, max_id]` in a similar way as in the algorithm described in Section 8.1.1. Collisions can be recovered by incrementing the 'counter' variable and recomputing `F()`.

The function `F()` should be a cryptographic hash function like SHA-256 [FIPS-SHS]. Note: MD5 [RFC1321] is considered unacceptable for `F()` [RFC6151]. CONTEXT is the concatenation of all the elements that define a given context. For example, if this algorithm is expected to produce identifiers that are unique per network interface card (NIC) and SLAAC autoconfiguration prefix, the CONTEXT should be the concatenation of e.g. the interface index and the SLAAC autoconfiguration prefix (please see [RFC7217] for an implementation of this algorithm for the generation of IPv6 IIDs).

The secret should be chosen to be as random as possible (see [RFC4086] for recommendations on choosing secrets).

8.4. Category #4: Uniqueness, monotonically increasing within context (hard failure)

8.4.1. Predictable Linear Identifiers Algorithm

One of the most trivial ways to achieve uniqueness with a low identifier reuse frequency is to produce a linear sequence. This obviously assumes that each identifier will be used for a similar period of time.

For example, the following algorithm has been employed in a number of operating systems for selecting IP fragment IDs, TCP ephemeral ports, etc.

```
/* Initialization at system boot time. Could be random */
next_id = min_id;
id_inc= 1;

/* Identifier selection function */
count = max_id - min_id + 1;

do {
    if (next_id == max_id) {
        next_id = min_id;
    }
    else {
        next_id = next_id + id_inc;
    }

    if (check_suitable_id(next_id))
        return next_id;

    count--;
} while (count > 0);

return ERROR;
```

Note:

check_suitable_id() is a function that checks whether the resulting identifier is acceptable (e.g., whether its in use, etc.).

For obvious reasons, this algorithm results in predictable sequences. If a global counter is used (such as "next_id" in the example above), a node that learns one protocol identifier can also learn or guess values employed by past and future protocol instances. On the other hand, when the value of increments is known (such as "1" in this case), an attacker can sample two values, and learn the number of identifiers that were generated in-between.

Where identifier reuse would lead to a hard failure, one typical approach to generate unique identifiers (while minimizing the security and privacy implications of predictable identifiers) is to obfuscate the resulting protocol IDs by either:

- o Replace the global counter with multiple counters (initialized to a random value)

- o Randomizing the "increments"

Avoiding global counters essentially means that learning one identifier for a given context (e.g., one TCP ephemeral port for a given {src IP, Dst IP, Dst Port}) is of no use for learning or guessing identifiers for a different context (e.g., TCP ephemeral ports that involve other peers). However, this may imply keeping one additional variable/counter per context, which may be prohibitive in some environments. The choice of `id_inc` has implications on both the security and privacy properties of the resulting identifiers, but also on the corresponding interoperability properties. On one hand, minimizing the increments (as in "`id_inc = 1`" in our case) generally minimizes the identifier reuse frequency, albeit at increased predictability. On the other hand, if the increments are randomized predictability of the resulting identifiers is reduced, and the information leakage produced by global constant increments is mitigated.

8.4.2. Per-context Counter Algorithm

One possible way to achieve similar (or even lower) identifier reuse frequency while still avoiding predictable sequences would be to employ a per-context counter, as opposed to a global counter. Such an algorithm could be described as follows:

```
/* Initialization at system boot time. Could be random */
id_inc= 1;

/* Identifier selection function */
count = max_id - min_id + 1;

if(lookup_counter(CONTEXT) == ERROR){
    create_counter(CONTEXT);
}

next_id= lookup_counter(CONTEXT);

do {
    if (next_id == max_id) {
        next_id = min_id;
    }
    else {
        next_id = next_id + id_inc;
    }

    if (check_suitable_id(next_id)){
        store_counter(CONTEXT, next_id);
        return next_id;
    }

    count--;
} while (count > 0);

store_counter(CONTEXT, next_id);
return ERROR;
```

NOTE:

lookup_counter() returns the current counter for a given context, or an error condition if such a counter does not exist.

create_counter() creates a counter for a given context, and initializes such counter to a random value.

store_counter() saves (updates) the current counter for a given context.

check_suitable_id() is a function that checks whether the resulting identifier is acceptable (e.g., whether its in use, etc.).

Essentially, whenever a new identifier is to be selected, the algorithm checks whether there is a counter for the

corresponding context. If there is, such counter is incremented to obtain the new identifier, and the new identifier updates the corresponding counter. If there is no counter for such context, a new counter is created and initialized to a random value, and used as the new identifier.

This algorithm produces a per-context counter, which results in one linear function for each context. Since the origin of each "line" is a random value, the resulting values are unknown to an off-path attacker.

This algorithm has the following drawbacks:

- o If, as a result of resource management, the counter for a given context must be removed, the last identifier value used for that context will be lost. Thus, if subsequently an identifier needs to be generated for such context, that counter will need to be recreated and reinitialized to random value, thus possibly leading to reuse/collision of identifiers.
- o If the identifiers are predictable by the destination system (e.g., the destination host represents the context), a vulnerable host might possibly leak to third parties the identifiers used by other hosts to send traffic to it (i.e., a vulnerable Host B could leak to Host C the identifier values that Host A is using to send packets to Host B). Appendix A of [RFC7739] describes one possible scenario for such leakage in detail.

8.4.3. Simple Hash-Based Algorithm

The goal of this algorithm is to produce monotonically-increasing sequences, with a randomized initial value, for each given context. For example, if the identifiers being generated must be unique for each {src IP, dst IP} set, then each possible combination of {src IP, dst IP} should have a corresponding "next_id" value.

Keeping one value for each possible "context" may in many cases be considered too onerous in terms of memory requirements. As a workaround, the following algorithm employs a calculated technique (as opposed to keeping state in memory) to maintain the random offset for each possible context.

In the following algorithm, the function F() provides (statelessly) a random offset for each given context.

```
/* Initialization at system boot time. Could be random. */
counter = 0;

/* Protocol ID selection function */
id_range = max_id - min_id + 1;
offset = F(CONTEXT, secret_key);
count = id_range;

do {
    next_id = min_id +
              (counter + offset) % id_range;

    counter++;

    if(check_suitable_id(next_id))
        return next_id;

    count--;
} while (count > 0);

return ERROR;
```

The function `F()` provides a "per-CONTEXT" fixed offset within the identifier space. Both the 'offset' and 'counter' variables may take any value within the storage type range since we are restricting the resulting identifier to be in the range `[min_id, max_id]` in a similar way as in the algorithm described in Section 8.1.1. This allows us to simply increment the 'counter' variable and rely on the unsigned integer to wrap around.

The function `F()` should be a cryptographic hash function like SHA-256 [FIPS-SHS]. Note: MD5 [RFC1321] is considered unacceptable for `F()` [RFC6151]. CONTEXT is the concatenation of all the elements that define a given context. For example, if this algorithm is expected to produce identifiers that are monotonically-increasing for each set (Source IP Address, Destination IP Address), the CONTEXT should be the concatenation of these two values.

The secret should be chosen to be as random as possible (see [RFC4086] for recommendations on choosing secrets).

It should be noted that, since this algorithm uses a global counter ("counter") for selecting identifiers, if an attacker could, e.g., force a client to periodically establish a new TCP connection to an attacker-controlled machine (or through an attacker-observable routing path), the attacker could substract consecutive source port

values to obtain the number of outgoing TCP connections established globally by the target host within that time period (up to wrap-around issues and five-tuple collisions, of course).

8.4.4. Double-Hash Algorithm

A trade-off between maintaining a single global 'counter' variable and maintaining $2*N$ 'counter' variables (where N is the width of the result of $F()$) could be achieved as follows. The system would keep an array of `TABLE_LENGTH` integers, which would provide a separation of the increment of the 'counter' variable. This improvement could be incorporated into the algorithm from Section 8.4.3 as follows:

```
/* Initialization at system boot time */
for(i = 0; i < TABLE_LENGTH; i++)
    table[i] = random();

id_inc = 1;

/* Protocol ID selection function */
id_range = max_id - min_id + 1;
offset = F(CONTEXT, secret_key1);
index = G(CONTEXT, secret_key2);
count = id_range;

do {
    next_id = min_id + (offset + table[index]) % id_range;
    table[index] = table[index] + id_inc;

    if(check_suitable_id(next_id))
        return next_id;

    count--;
} while (count > 0);

return ERROR;
```

'table[]' could be initialized with random values, as indicated by the initialization code in pseudo-code above. The function $G()$ should be a cryptographic hash function. It should use the same `CONTEXT` as $F()$, and a secret key value to compute a value between 0 and $(TABLE_LENGTH-1)$. Alternatively, $G()$ could take an "offset" as input, and perform the exclusive-or (XOR) operation between all the bytes in 'offset'.

The array 'table[]' assures that successive identifiers for a given context will be monotonically-increasing. However, the increments space is separated into TABLE_LENGTH different spaces, and thus identifier reuse frequency will be (probabilistically) lower than that of the algorithm in Section 8.4.3. That is, the generation of identifier for one given context will not necessarily result in increments in the identifiers for other contexts.

It is interesting to note that the size of 'table[]' does not limit the number of different identifier sequences, but rather separates the *increments* into TABLE_LENGTH different spaces. The identifier sequence will result from adding the corresponding entry of 'table[]' to the variable 'offset', which selects the actual identifier sequence (as in the algorithm from Section 8.4.3).

An attacker can perform traffic analysis for any "increment space" into which the attacker has "visibility" -- namely, the attacker can force a node to generate identifiers where G(offset) identifies the target "increment space". However, the attacker's ability to perform traffic analysis is very reduced when compared to the predictable linear identifiers (described in Section 8.4.1) and the hash-based identifiers (described in Section 8.4.3). Additionally, an implementation can further limit the attacker's ability to perform traffic analysis by further separating the increment space (that is, using a larger value for TABLE_LENGTH) and/or by randomizing the increments.

8.4.5. Random-Increments Algorithm

This algorithm offers a middle ground between the algorithms that select ephemeral ports randomly (such as those described in Section 8.1.1 and Section 8.1.2), and those that offer obfuscation but no randomization (such as those described in Section 8.4.3 and Section 8.4.4).

```
/* Initialization code at system boot time. */
next_id = random();          /* Initialization value */
id_inc = 500;                /* Determines the trade-off */

/* Identifier selection function */
id_range = max_id - min_id + 1;

count = id_range;

do {
    /* Random increment */
    next_id = next_id + (random() % id_inc) + 1;

    /* Keep the identifier within acceptable range */
    next_id = min_id + (next_id % id_range);

    if(check_suitable_id(next_id))
        return next_id;

    count--;
} while (count > 0);

return ERROR;
```

This algorithm aims at producing a monotonically increasing sequence of identifiers, while avoiding the use of fixed increments, which would lead to trivially predictable sequences. The value "id_inc" allows for direct control of the trade-off between the level of obfuscation and the ID reuse frequency. The smaller the value of "id_inc", the more similar this algorithm is to a predictable, global monotonically-increasing ID generation algorithm. The larger the value of "id_inc", the more similar this algorithm is to the algorithm described in Section 8.1.1 of this document.

When the identifiers wrap, there is the risk of collisions of identifiers (i.e., identifier reuse). Therefore, "id_inc" should be selected according to the following criteria:

- o It should maximize the wrapping time of the identifier space.
- o It should minimize identifier reuse frequency.
- o It should maximize obfuscation.

Clearly, these are competing goals, and the decision of which value of "id_inc" to use is a trade-off. Therefore, the value of "id_inc"

should be configurable so that system administrators can make the trade-off for themselves.

9. Common Vulnerabilities Associated with Identifiers

This section analyzes common vulnerabilities associated with the generation of identifiers for each of the categories identified in Section 7.

9.1. Category #1: Uniqueness (soft failure)

Possible vulnerabilities associated with identifiers of this category are:

- o Use of trivial algorithms (e.g. global counters) that generate predictable identifiers
- o Use of flawed PRNGs.

Since the only interoperability requirement for these identifiers is uniqueness, the obvious approach to generate them is to employ a PRNG. An implementer should consult [RFC4086] regarding randomness requirements for security, and consult relevant documentation when employing a PRNG provided by the underlying system.

Use algorithms other than PRNGs for generating identifiers of this category is discouraged.

9.2. Category #2: Uniqueness (hard failure)

As noted in Section 8.2 this category typically employs the same algorithms as Category #4, since a monotonically-increasing sequence tends to minimize the identifier reuse frequency. Therefore, the vulnerability analysis of Section 9.4 applies to this case.

9.3. Category #3: Uniqueness, constant within context (soft failure)

There are two main vulnerabilities that may be associated with identifiers of this category:

1. Use algorithms or sources that result in predictable identifiers
2. Employing the same identifier across contexts in which constantcy is not required

At times, an implementation or specification may be tempted to employ a source for the identifier which is known to provide unique values. However, while unique, the associated identifiers may have other

properties such as being predictable or leaking information about the node in question. For example, as noted in [RFC7721], embedding link-layer addresses for generating IPv6 IIDs not only results in predictable values, but also leaks information about the manufacturer of the network interface card.

On the other hand, using an identifier across contexts where constancy is not required can be leveraged for correlation of activities. One of the most trivial examples of this is the use of IPv6 IIDs that are constant across networks (such as IIDs that embed the underlying link-layer address).

9.4. Category #4: Uniqueness, monotonically increasing within context (hard failure)

A simple way to generalize algorithms employed for generating identifiers of Category #4 would be as follows:

```
/* Identifier selection function */
count = max_id - min_id + 1;

do {
    linear(CONTEXT) = linear(CONTEXT) + increment();
    next_id = offset(CONTEXT) + linear(CONTEXT);

    if (check_suitable_id(next_id))
        return next_id;

    count--;
} while (count > 0);

return ERROR;
```

Essentially, an identifier (next_id) is generated by adding a linear function (linear()) to an offset value, which is unknown to the attacker, and constant for given context.

The following aspects of the algorithm should be considered:

- o For the most part, it is the offset() function that results in identifiers that are unpredictable by an off-path attacker. While the resulting sequence will be monotonically-increasing, the use of an offset value that is unknown to the attacker makes the resulting values unknown to the attacker.
- o The most straightforward "stateless" implementation of offset would be that in which offset() is the result of a

cryptographically-secure hash-function that takes the values that identify the context and a "secret" (not shown in the figure above) as arguments.

- o Another possible (but stateful) approach would be to simply generate a random offset and store it in memory, and then look-up the corresponding context when a new identifier is to be selected. The algorithm in Section 8.4.2 is essentially an implementation of this type.
- o The linear function is incremented according to `increment()`. In the most trivial case `increment()` could always return the constant "1". But it could also possibly return small integers such the increments are randomized.

Considering the generic algorithm illustrated above we can identify the following possible vulnerabilities:

- o If the offset value spans more than the necessary context, identifiers could be unnecessarily predictable by other parties, since the offset value would be unnecessarily leaked to them. For example, an implementation that means to produce a per-destination counter but replaces `offset()` with a constant number (i.e., employs a global counter), will unnecessarily result in predictable identifiers.
- o The function `linear()` could be seen as representing the number of identifiers that have so far been generated for a given context. If `linear()` spans more than the necessary context, the "increments" could be leaked to other parties, thus disclosing information about the number of identifiers that have so far been generated. For example, an implementation in which `linear()` is implemented as a single global counter will unnecessarily leak information the number of identifiers that have been produced.
- o `increment()` determines how the `linear()` is incremented for each identifier that is selected. In the most trivial case, `increment()` will return the integer "1". However, an implementation may have `increment()` return a "small" integer value such that even if the current value employed by the generator is guessed (see Appendix A of [RFC7739]), the exact next identifier to be selected will be slightly harder to identify.

10. Security and Privacy Requirements for Identifiers

Protocol specifications that specify identifiers should:

1. Clearly specify the interoperability requirements for selecting the aforementioned identifiers.
2. Provide a security and privacy analysis of the aforementioned identifiers.
3. Recommend an algorithm for generating the aforementioned identifiers that mitigates security and privacy issues, such as those discussed in Section 9.

11. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

12. Security Considerations

The entire document is about the security and privacy implications of identifiers.

13. Acknowledgements

The authors would like to thank (in alphabetical order) Steven Bellovin, Joseph Lorenzo Hall, Gre Norcie, and Martin Thomson, for providing valuable comments on earlier versions of this document.

The authors would like to thank Diego Armando Maradona for his magic and inspiration.

14. References

14.1. Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [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>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.

- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <<https://www.rfc-editor.org/info/rfc5722>>.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.
- [RFC6528] Gont, F. and S. Bellovin, "Defending against Sequence Number Attacks", RFC 6528, DOI 10.17487/RFC6528, February 2012, <<https://www.rfc-editor.org/info/rfc6528>>.
- [RFC7098] Carpenter, B., Jiang, S., and W. Tarreau, "Using the IPv6 Flow Label for Load Balancing in Server Farms", RFC 7098, DOI 10.17487/RFC7098, January 2014, <<https://www.rfc-editor.org/info/rfc7098>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.

14.2. Informative References

- [Bellovin1989] Bellovin, S., "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review, vol. 19, no. 2, pp. 32-48, 1989, <<https://www.cs.columbia.edu/~smb/papers/ipext.pdf>>.

[Bellovin2002]

Bellovin, S., "A Technique for Counting NATted Hosts",
IMW'02 Nov. 6-8, 2002, Marseille, France, 2002.

[CERT2001]

CERT, "CERT Advisory CA-2001-09: Statistical Weaknesses in
TCP/IP Initial Sequence Numbers", 2001,
<<http://www.cert.org/advisories/CA-2001-09.html>>.

[CPNI-TCP]

Gont, F., "Security Assessment of the Transmission Control
Protocol (TCP)", United Kingdom's Centre for the
Protection of National Infrastructure (CPNI) Technical
Report, 2009, <[http://www.gont.com.ar/papers/
tn-03-09-security-assessment-TCP.pdf](http://www.gont.com.ar/papers/tn-03-09-security-assessment-TCP.pdf)>.

[FIPS-SHS]

FIPS, "Secure Hash Standard (SHS)", Federal Information
Processing Standards Publication 180-4, March 2012,
<[http://csrc.nist.gov/publications/fips/fips180-4/
fips-180-4.pdf](http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf)>.

[Fyodor2004]

Fyodor, "Idle scanning and related IP ID games", 2004,
<<http://www.insecure.org/nmap/idlescan.html>>.

[Gont2011]

Gont, F., "Hacking IPv6 Networks (training course)", Hack
In Paris 2011 Conference Paris, France, June 2011.

[Gont2012]

Gont, F., "Recent Advances in IPv6 Security", BSDCan 2012
Conference Ottawa, Canada. May 11-12, 2012, May 2012.

[I-D.eddy-rfc793bis-04]

Eddy, W., "Transmission Control Protocol Specification",
draft-eddy-rfc793bis-04 (work in progress), August 2014.

[I-D.gont-6man-flowlabel-security]

Gont, F., "Security Assessment of the IPv6 Flow Label",
draft-gont-6man-flowlabel-security-03 (work in progress),
March 2012.

[I-D.ietf-6man-default-iids]

Gont, F., Cooper, A., Thaler, D., and S. LIU,
"Recommendation on Stable IPv6 Interface Identifiers",
draft-ietf-6man-default-iids-16 (work in progress),
September 2016.

- [I-D.ietf-6man-predictable-fragment-id-08]
Gont, F., "Security Implications of Predictable Fragment Identification Values", draft-ietf-6man-predictable-fragment-id-08 (work in progress), June 2015.
- [Joncheray1995]
Joncheray, L., "A Simple Active Attack Against TCP", Proc. Fifth Usenix UNIX Security Symposium, 1995.
- [Klein2007]
Klein, A., "OpenBSD DNS Cache Poisoning and Multiple O/S Predictable IP ID Vulnerability", 2007,
<http://www.trusteer.com/files/OpenBSD_DNS_Cache_Poisoning_and_Multiple_OS_Predictable_IP_ID_Vulnerability.pdf>.
- [Morris1985]
Morris, R., "A Weakness in the 4.2BSD UNIX TCP/IP Software", CSTR 117, AT&T Bell Laboratories, Murray Hill, NJ, 1985,
<<https://pdos.csail.mit.edu/~rtm/papers/117.pdf>>.
- [RedHat2011]
RedHat, "RedHat Security Advisory RHSA-2011:1465-1: Important: kernel security and bug fix update", 2011,
<<https://rhn.redhat.com/errata/RHSA-2011-1465.html>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992,
<<https://www.rfc-editor.org/info/rfc1321>>.
- [RFC1948] Bellare, S., "Defending Against Sequence Number Attacks", RFC 1948, DOI 10.17487/RFC1948, May 1996,
<<https://www.rfc-editor.org/info/rfc1948>>.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007,
<<https://www.rfc-editor.org/info/rfc4963>>.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010,
<<https://www.rfc-editor.org/info/rfc5927>>.

- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <<https://www.rfc-editor.org/info/rfc6056>>.
- [RFC7528] Higgs, P. and J. Piesing, "A Uniform Resource Name (URN) Namespace for the Hybrid Broadcast Broadband TV (HbbTV) Association", RFC 7528, DOI 10.17487/RFC7528, April 2015, <<https://www.rfc-editor.org/info/rfc7528>>.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <<https://www.rfc-editor.org/info/rfc7707>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<https://www.rfc-editor.org/info/rfc7739>>.
- [Sanfilippo1998a]
Sanfilippo, S., "about the ip header id", Post to Bugtraq mailing-list, Mon Dec 14 1998, <<http://seclists.org/bugtraq/1998/Dec/48>>.
- [Sanfilippo1998b]
Sanfilippo, S., "Idle scan", Post to Bugtraq mailing-list, 1998, <<http://www.kyuzz.org/antirez/papers/dumbscan.html>>.
- [Sanfilippo1999]
Sanfilippo, S., "more ip id", Post to Bugtraq mailing-list, 1999, <<http://www.kyuzz.org/antirez/papers/moreipid.html>>.
- [Shimomura1995]
Shimomura, T., "Technical details of the attack described by Markoff in NYT", Message posted in USENET's comp.security.misc newsgroup Message-ID: <3g5gkl\$5jl@ariel.sdsc.edu>, 1995, <<http://www.gont.com.ar/docs/post-shimomura-usenet.txt>>.

[Silbersack2005]

Silbersack, M., "Improving TCP/IP security through randomization without sacrificing interoperability", EuroBSDCon 2005 Conference, 2005, <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.4542&rep=rep1&type=pdf>>.

[SUSE2011]

SUSE, "SUSE Security Announcement: Linux kernel security update (SUSE-SA:2011:046)", 2011, <<http://lists.opensuse.org/opensuse-security-announce/2011-12/msg00011.html>>.

[Ubuntu2011]

Ubuntu, "Ubuntu: USN-1253-1: Linux kernel vulnerabilities", 2011, <<http://www.ubuntu.com/usn/usn-1253-1/>>.

[USCERT2001]

US-CERT, "US-CERT Vulnerability Note VU#498440: Multiple TCP/IP implementations may use statistically predictable initial sequence numbers", 2001, <<http://www.kb.cert.org/vuls/id/498440>>.

[Zalewski2001]

Zalewski, M., "Strange Attractors and TCP/IP Sequence Number Analysis", 2001, <<http://lcamtuf.coredump.cx/oldtcp/tcpseq.html>>.

[Zalewski2002]

Zalewski, M., "Strange Attractors and TCP/IP Sequence Number Analysis - One Year Later", 2001, <<http://lcamtuf.coredump.cx/newtcp/>>.

[Zalewski2003]

Zalewski, M., "A new TCP/IP blind data injection technique?", 2003, <<http://lcamtuf.coredump.cx/ipfrag.txt>>.

Authors' Addresses

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <http://www.si6networks.com>

Ivan Arce
Quarkslab

Email: iarce@quarkslab.com
URI: <https://www.quarkslab.com>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 16, 2018

K. Moriarty, Ed.
Dell EMC
A. Morton, Ed.
AT&T Labs
March 15, 2018

Effects of Pervasive Encryption on Operators
draft-mm-wg-effect-encrypt-25

Abstract

Pervasive Monitoring (PM) attacks on the privacy of Internet users are of serious concern to both the user and the operator communities. RFC7258 discussed the critical need to protect users' privacy when developing IETF specifications and also recognized making networks unmanageable to mitigate PM is not an acceptable outcome; an appropriate balance is needed. This document discusses current security and network operations and management practices that may be impacted by the shift to increased use of encryption to help guide protocol development in support of manageable and secure networks.

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 September 16, 2018.

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 | 3 |
| 1.1. Additional Background on Encryption Changes | 4 |
| 1.2. Examples of Attempts to Preserve Functions | 6 |
| 2. Network Service Provider Monitoring | 7 |
| 2.1. Passive Monitoring | 8 |
| 2.1.1. Traffic Surveys | 8 |
| 2.1.2. Troubleshooting | 8 |
| 2.1.3. Traffic Analysis Fingerprinting | 11 |
| 2.2. Traffic Optimization and Management | 12 |
| 2.2.1. Load Balancers | 12 |
| 2.2.2. Differential Treatment based on Deep Packet Inspection (DPI) | 14 |
| 2.2.3. Network Congestion Management | 15 |
| 2.2.4. Performance-enhancing Proxies | 15 |
| 2.2.5. Caching and Content Replication Near the Network Edge | 16 |
| 2.2.6. Content Compression | 17 |
| 2.2.7. Service Function Chaining | 18 |
| 2.3. Content Filtering, Network Access, and Accounting | 18 |
| 2.3.1. Content Filtering | 19 |
| 2.3.2. Network Access and Data Usage | 20 |
| 2.3.3. Application Layer Gateways | 21 |
| 2.3.4. HTTP Header Insertion | 22 |
| 3. Encryption in Hosting and Application SP Environments | 22 |
| 3.1. Management Access Security | 22 |
| 3.1.1. Customer Access Monitoring | 23 |
| 3.1.2. SP Content Monitoring of Applications | 24 |
| 3.2. Hosted Applications | 26 |
| 3.2.1. Monitoring Managed Applications | 26 |
| 3.2.2. Mail Service Providers | 27 |
| 3.3. Data Storage | 27 |
| 3.3.1. Object-level Encryption | 27 |
| 3.3.2. Disk Encryption, Data at Rest | 28 |
| 3.3.3. Cross Data Center Replication Services | 29 |
| 4. Encryption for Enterprises | 29 |
| 4.1. Monitoring Practices of the Enterprise | 30 |
| 4.1.1. Security Monitoring in the Enterprise | 30 |
| 4.1.2. Application Performance Monitoring in the Enterprise | 31 |
| 4.1.3. Enterprise Network Diagnostics and Troubleshooting | 32 |
| 4.2. Techniques for Monitoring Internet Session Traffic | 34 |
| 5. Security Monitoring for Specific Attack Types | 36 |

| | | |
|------|---|----|
| 5.1. | Mail Abuse and spam | 36 |
| 5.2. | Denial of Service | 37 |
| 5.3. | Phishing | 37 |
| 5.4. | Botnets | 38 |
| 5.5. | Malware | 38 |
| 5.6. | Spoofed Source IP Address Protection | 39 |
| 5.7. | Further work | 39 |
| 6. | Application-based Flow Information Visible to a Network | 39 |
| 6.1. | IP Flow Information Export | 39 |
| 6.2. | TLS Server Name Indication | 40 |
| 6.3. | Application Layer Protocol Negotiation (ALPN) | 41 |
| 6.4. | Content Length, BitRate and Pacing | 41 |
| 7. | Effect of Encryption on Mobile Network Evolution | 41 |
| 8. | Response to Increased Encryption and Looking Forward | 42 |
| 9. | Security Considerations | 43 |
| 10. | IANA Considerations | 43 |
| 11. | Acknowledgements | 43 |
| 12. | Informative References | 43 |
| | Authors' Addresses | 52 |

1. Introduction

In response to pervasive monitoring revelations and the IETF consensus that Pervasive Monitoring is an Attack [RFC7258], efforts are underway to increase encryption of Internet traffic. Pervasive Monitoring (PM) is of serious concern to users, operators, and application providers. RFC7258 discussed the critical need to protect users' privacy when developing IETF specifications and also recognized that making networks unmanageable to mitigate PM is not an acceptable outcome, but rather that an appropriate balance would emerge over time.

This document describes practices currently used by network operators to manage, operate, and secure their networks and how those practices may be impacted by a shift to increased use of encryption. It provides network operators' perspectives about the motivations and objectives of those practices as well as effects anticipated by operators as use of encryption increases. It is a summary of concerns of the operational community as they transition to managing networks with less visibility. The document does not endorse the use of the practices described herein. Nor does it aim to provide a comprehensive treatment of the effects of current practices, some of which have been considered controversial from a technical or business perspective or contradictory to previous IETF statements (e.g., [RFC1958], [RFC1984], [RFC2804]). The informational documents consider the end to end (e2e) architectural principle to be a guiding principle for the development of Internet protocols [RFC2775] [RFC3724] [RFC7754].

This document aims to help IETF participants understand network operators' perspectives about the impact of pervasive encryption, both opportunistic and strong end-to-end encryption, on operational practices. The goal is to help inform future protocol development to ensure that operational impact is part of the conversation. Perhaps, new methods could be developed to accomplish some of the goals of current practices despite changes in the extent to which cleartext will be available to network operators (including methods that rely on network endpoints where applicable). Discussion of current practices and the potential future changes is provided as a prerequisite to potential future cross-industry and cross-layer work to support the ongoing evolution towards a functional Internet with pervasive encryption.

Traditional network management, planning, security operations, and performance optimization have been developed in an Internet where a large majority of data traffic flows without encryption. While unencrypted traffic has made information that aids operations and troubleshooting at all layers accessible, it has also made pervasive monitoring by unseen parties possible. With broad support and increased awareness of the need to consider privacy in all aspects across the Internet, it is important to catalog existing management, operational, and security practices that have depended upon the availability of cleartext to function and to explore if critical operational practices can be met by less invasive means.

This document refers to several different forms of service providers, distinguished with adjectives. For example, network service providers (or network operators) provide IP-packet transport primarily, though they may bundle other services with packet transport. Alternatively, application service providers primarily offer systems that participate as an end-point in communications with the application user, and hosting service providers lease computing, storage, and communications systems in datacenters. In practice, many companies perform two or more service provider roles, but may be historically associated with one.

This document includes a sampling of current practices and does not attempt to describe every nuance. Some sections cover technologies used over a broad spectrum of devices and use cases.

1.1. Additional Background on Encryption Changes

Pervasive encryption in this document refers to all types of session encryption including Transport Layer Security (TLS), IP security (IPsec), TCPcrypt [TCPcrypt], QUIC [QUIC] and others that are increasing in deployment usage. It is well understood that session encryption helps to prevent both passive and active attacks on

transport protocols; more on pervasive monitoring can be found in Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement [RFC7624]. Active attacks have long been a motivation for increased encryption, and preventing pervasive monitoring became a focus just a few years ago. As such, the Internet Architecture Board (IAB) released a statement advocating for increased use of encryption in November 2014. Perspectives on encryption paradigms have shifted over time to incorporate ease of deployment as a high priority, and balance that against providing the maximum possible level of security regardless of deployment considerations.

One such shift is documented in "Opportunistic Security" (OS) [RFC7435], which suggests that when use of authenticated encryption is not possible, cleartext sessions should be upgraded to unauthenticated session encryption, rather than no encryption. OS encourages upgrading from cleartext, but cannot require or guarantee such upgrades. Once OS is used, it allows for an evolution to authenticated encryption. These efforts are necessary to improve end user's expectation of privacy, making pervasive monitoring cost prohibitive. With OS in use, active attacks are still possible on unauthenticated sessions. OS has been implemented as NULL Authentication with IPsec [RFC7619] and there are a number of infrastructure use cases such as server to server encryption where this mode is deployed. While OS is helpful in reducing pervasive monitoring by increasing the cost to monitor, it is recognized that risk profiles for some applications require authenticated and secure session encryption as well to prevent active attacks. IPsec, and other session encryption protocols, with authentication has many useful applications and usage has increased for infrastructure applications such as for virtual private networks between data centers. OS as well as other protocol developments, like the Automated Certificate Management Environment (ACME), have increased the usage of session encryption on the Internet.

Risk profiles vary and so do the types of session encryption deployed. To understand the scope of changes in visibility a few examples are highlighted. Work continues to improve the implementation, development and configuration of TLS and DTLS sessions to prevent active attacks used to monitor or intercept session data. The changes from TLS 1.2 to 1.3 enhance the security of TLS, while hiding more of the session negotiation and providing less visibility on the wire. The Using TLS in Applications (UTA) working group has been publishing documentation to improve the security of TLS and DTLS sessions. They have documented the known attack vectors in [RFC7457] and have documented Best Practices for TLS and DTLS in [RFC7525] and have other documents in the queue. The

recommendations from these documents were built upon for TLS 1.3 to provide a more inherently secure end-to-end protocol.

In addition to encrypted web site access (HTTP over TLS), there are other well-deployed application level transport encryption efforts such as mail transfer agent (MTA)-to-MTA session encryption transport for email (SMTP over TLS) and gateway-to-gateway for instant messaging (Extensible Messaging and Presence Protocol (XMPP) over TLS). Although this does provide protection from transport layer attacks, the servers could be a point of vulnerability if user-to-user encryption is not provided for these messaging protocols. User-to-user content encryption schemes, such as S/MIME and PGP for email and Off-the-Record (OTR) encryption for XMPP are used by those interested to protect their data as it crosses intermediary servers, preventing transport layer attacks by providing an end-to-end solution. User-to-user schemes are under review and additional options will emerge to ease the configuration requirements, making this type of option more accessible to non-technical users interested in protecting their privacy.

Increased use of encryption, either opportunistic or authenticated, at the transport, network or application layer, impacts how networks are operated, managed, and secured. In some cases, new methods to operate, manage, and secure networks will evolve in response. In other cases, currently available capabilities for monitoring or troubleshooting networks could become unavailable. This document lists a collection of functions currently employed by network operators that may be impacted by the shift to increased use of encryption. This draft does not attempt to specify responses or solutions to these impacts, but rather documents the current state.

1.2. Examples of Attempts to Preserve Functions

Following the Snowden [Snowden] revelations, application service providers responded by encrypting traffic between their data centers (IPsec) to prevent passive monitoring from taking place unbeknownst to them (Yahoo, Google, etc.). Infrastructure traffic carried over the public Internet has been encrypted for some time, this change for universal encryption was specific to their private backbones. Large mail service providers also began to encrypt session transport (TLS) to hosted mail services. This and other increases in the use of encryption had the immediate effect of providing confidentiality and integrity for protected data, but created a problem for some network management functions. Operators could no longer gain access to some session streams resulting in actions by several to regain their operational practices that previously depended on cleartext data sessions.

The EFF reported [EFF2014] several network service providers using a downgrade attack to prevent the use of SMTP over TLS by breaking STARTTLS (section 3.2 of [RFC7525]), essentially preventing the negotiation process resulting in fallback to the use of clear text. There have already been documented cases of service providers preventing STARTTLS to prevent session encryption negotiation on some session to inject a super cookie to enable tracking of users for advertisers, also considered an attack. These serve as examples of undesirable behavior that could be prevented through upfront discussions in protocol work for operators and protocol designers to understand the implications of such actions. In other cases, some service providers and enterprises have relied on middleboxes having access to clear text for the purposes of load balancing, monitoring for attack traffic, meeting regulatory requirements, or for other purposes. The implications for enterprises, who own the data on their networks or have explicit agreements that permit monitoring of user traffic is very different from service providers who may be accessing content in a way that violates privacy considerations. Additionally, service provider equipment is designed for accessing only the headers exposed for the data-link, network, and transport layers. Delving deeper into packets is possible, but there is typically a high degree of accuracy from the header information and packet sizes when limited to header information from these three layers. Service providers also have the option of adding routing overlay protocols to traffic. These middlebox implementations, whether performing functions considered legitimate by the IETF or not, have been impacted by increases in encrypted traffic. Only methods keeping with the goal of balancing network management and PM mitigation in [RFC7258] should be considered in solution work resulting from this document.

It is well known that national surveillance programs monitor traffic [JNSLP] [RFC2804] [RFC7258] monitor for criminal activities. Governments vary on their balance between monitoring versus the protection of user privacy, data, and assets. Those that favor unencrypted access to data ignore the real need to protect users' identity, financial transactions and intellectual property, which requires security and encryption to prevent crime. A clear understanding of technology, encryption, and monitoring goals will aid in the development of solutions as work continues towards finding an appropriate balance allowing for management while protecting users privacy with strong encryption solutions.

2. Network Service Provider Monitoring

Network Service Providers (SP) for this definition include the backbone Internet Service providers as well as those providing

infrastructure at scale for core Internet use (hosted infrastructure and services such as email).

Network service providers use various techniques to operate, manage, and secure their networks. The following subsections detail the purpose of several techniques and which protocol fields are used to accomplish each task. In response to increased encryption of these fields, some network service providers may be tempted to undertake undesirable security practices in order to gain access to the fields in unencrypted data flows. To avoid this situation, new methods could be developed to accomplish the same goals without service providers having the ability to see session data.

2.1. Passive Monitoring

2.1.1. Traffic Surveys

Internet traffic surveys are useful in many pursuits, such as input for Center for Applied Internet Data Analysis (CAIDA) studies [CAIDA], network planning and optimization. Tracking the trends in Internet traffic growth, from earlier peer-to-peer communication to the extensive adoption of unicast video streaming applications, has relied on a view of traffic composition with a particular level of assumed accuracy, based on access to cleartext by those conducting the surveys.

Passive monitoring makes inferences about observed traffic using the maximal information available, and is subject to inaccuracies stemming from incomplete sampling (of packets in a stream) or loss due to monitoring system overload. When encryption conceals more layers in each packet, reliance on pattern inferences and other heuristics grows, and accuracy suffers. For example, the traffic patterns between server and browser are dependent on browser supplier and version, even when the sessions use the same server application (e.g., web e-mail access). It remains to be seen whether more complex inferences can be mastered to produce the same monitoring accuracy.

2.1.2. Troubleshooting

Network operators use protocol-dissecting analyzers when responding to customer problems, to identify the presence of attack traffic, and to identify root causes of the problem such as misconfiguration. In limited cases, packet captures may also be used when a customer approves of access to their packets or provides packet captures close to the endpoint. The protocol dissection is generally limited to supporting protocols (e.g., DNS, DHCP), network and transport (e.g., IP, TCP), and some higher layer protocols (e.g., RTP, RTCP).

Troubleshooting will move closer to the endpoint with increased encryption and adjustments in practices to effectively troubleshoot using a 5-tuple may require education. Packet loss investigations, and those where access is limited to a 2-tuple (IPsec tunnel mode), rely on network and transport layer headers taken at the endpoint. In this case, captures on intermediate nodes are not reliable as there are far too many cases of aggregate interfaces and alternate paths in service provider networks.

Network operators are often the first ones called upon to investigate application problems (e.g., "my HD video is choppy"), to first rule out network and network services as a cause for the underlying issue. When diagnosing a customer problem, the starting point may be a particular application that isn't working. The ability to identify the problem application's traffic is important and packet capture provided from the customer close to the edge may be used for this purpose; IP address filtering is not useful for applications using content delivery networks (CDNs) or cloud providers. After identifying the traffic, an operator may analyze the traffic characteristics and routing of the traffic. This diagnostic step is important to help determine the root cause before exploring if the issue is directly with the application.

For example, by investigating packet loss (from TCP sequence and acknowledgement numbers), round-trip-time (from TCP timestamp options or application-layer transactions, e.g., DNS or HTTP response time), TCP receive-window size, packet corruption (from checksum verification), inefficient fragmentation, or application-layer problems, the operator can narrow the problem to a portion of the network, server overload, client or server misconfiguration, etc. Network operators may also be able to identify the presence of attack traffic as not conforming to the application the user claims to be using. In many instances, the exposed packet header is sufficient for this type of troubleshooting.

One way of quickly excluding the network as the bottleneck during troubleshooting is to check whether the speed is limited by the endpoints. For example, the connection speed might instead be limited by suboptimal TCP options, the sender's congestion window, the sender temporarily running out of data to send, the sender waiting for the receiver to send another request, or the receiver closing the receive window. All this information can be derived from the cleartext TCP header.

Packet captures and protocol-dissecting analyzers have been important tools. Automated monitoring has also been used to proactively identify poor network conditions, leading to maintenance and network upgrades before user experience declines. For example, findings of

loss and jitter in VoIP traffic can be a predictor of future customer dissatisfaction (supported by metadata from the RTP/RTCP protocol) [RFC3550], or increases in DNS response time can generally make interactive web browsing appear sluggish. But to detect such problems, the application or service stream must first be distinguished from others.

When increased encryption is used, operators lose a source of data that may be used to debug user issues. For example, IPsec obscures TCP and RTP header information, while TLS and SRTP do not. Because of this, application server operators using increased encryption might be called upon more frequently to assist with debugging and troubleshooting, and thus may want to consider what tools can be put in the hands of their clients or network operators.

Further, the performance of some services can be more efficiently managed and repaired when information on user transactions is available to the service provider. It may be possible to continue transaction monitoring activities without clear text access to the application layers of interest, but inaccuracy will increase and efficiency of repair activities will decrease. For example, an application protocol error or failure would be opaque to network troubleshooters when transport encryption is applied, making root cause location more difficult and therefore increasing the time-to-repair. Repair time directly reduces the availability of the service, and most network operators have made availability a key metric in their Service Level Agreements and/or subscription rebates. Also, there may be more cases of user communication failures when the additional encryption processes are introduced (e.g., key management at large scale), leading to more customer service contacts and (at the same time) less information available to network operations repair teams.

In mobile networks, knowledge about TCP's stream transfer progress (by observing ACKs, retransmissions, packet drops, and the Sector Utilization Level etc.) is further used to measure the performance of Network Segments (Sector, eNodeB (eNB) etc.). This information is used as key performance indicators (KPIs) and for the estimation of user/service key quality indicators at network edges for circuit emulation (CEM) as well as input for mitigation methods. If the make-up of active services per user and per sector are not visible to a server that provides Internet Access Point Names (APN), it cannot perform mitigation functions based on network segment view.

It is important to note that the push for encryption by application providers has been motivated by the application of the described techniques. Although network operators have noted performance improvements with network-based optimization or enhancement of user

traffic (otherwise, deployment would not have occurred), application providers have likewise noted some degraded performance and/or user experience, and such cases may result in additional operator troubleshooting. Further, encrypted application streams might avoid outdated optimization or enhancement techniques, where they exist.

A gap exists for vendors where built-in diagnostics and serviceability are not adequate to provide detailed logging and debugging capabilities that, when possible, could be accessed with cleartext network parameters. In addition to traditional logging and debugging methods, packet tracing and inspection along the service path provides operators the visibility to continue to diagnose problems reported both internally and by their customers. Logging of service path upon exit for routing overlay protocols will assist with policy management and troubleshooting capabilities for traffic flows on encrypted networks. Protocol trace logging and protocol data unit (PDU) logging should also be considered to improve visibility to monitor and troubleshoot application level traffic. Additional work on this gap would assist network operators to better troubleshoot and manage networks with increasing amounts of encrypted traffic.

2.1.1.3. Traffic Analysis Fingerprinting

Fingerprinting is used in traffic analysis and monitoring to identify traffic streams that match certain patterns. This technique can be used with both clear text or encrypted sessions. Some Distributed Denial of Service (DDoS) prevention techniques at the network provider level rely on the ability to fingerprint traffic in order to mitigate the effect of this type of attack. Thus, fingerprinting may be an aspect of an attack or part of attack countermeasures.

A common, early trigger for DDoS mitigation includes observing uncharacteristic traffic volumes or sources; congestion; or degradation of a given network or service. One approach to mitigate such an attack involves distinguishing attacker traffic from legitimate user traffic. The ability to examine layers and payloads above transport provides an increased range of filtering opportunities at each layer in the clear. If fewer layers are in the clear, this means that there are reduced filtering opportunities available to mitigate attacks. However, fingerprinting is still possible.

Passive monitoring of network traffic can lead to invasion of privacy by external actors at the endpoints of the monitored traffic. Encryption of traffic end-to-end is one method to obfuscate some of the potentially identifying information. For example, browser fingerprints are comprised of many characteristics, including User Agent, HTTP Accept headers, browser plug-in details, screen size and

color details, system fonts and time zone. A monitoring system could easily identify a specific browser, and by correlating other information, identify a specific user.

2.2. Traffic Optimization and Management

2.2.1. Load Balancers

A standalone load balancer is a function one can take off the shelf, place in front of a pool of servers, configure appropriately, and it will balance the traffic load among servers in the pool. This is a typical setup for load balancers. Standalone load balancers rely on the plainly observable information in the packets they are forwarding and rely on industry-accepted standards in interpreting the plainly observable information. Typically, this is a 5-tuple of the connection. This type of configuration terminates TLS sessions at the load balancer, making it the end point instead of the server. Standalone load balancers are considered middleboxes, but are an integral part of server infrastructure that scales.

In contrast, an integrated load balancer is developed to be an integral part of the service provided by the server pool behind that load balancer. These load balancers can communicate state with their pool of servers to better route flows to the appropriate servers. They rely on non-standard system-specific information and operational knowledge shared between the load balancer and its servers.

Both standalone and integrated load balancers can be deployed in pools for redundancy and load sharing. For high availability, it is important that when packets belonging to a flow start to arrive at a different load balancer in the load balancer pool, the packets continue to be forwarded to the original server in the server pool. The importance of this requirement increases as the chances of such load balancer change event increases.

Mobile operators deploy integrated load balancers to assist with maintaining connection state as devices migrate. With the proliferation of mobile connected devices, there is an acute need for connection-oriented protocols that maintain connections after a network migration by an endpoint. This connection persistence provides an additional challenge for multi-homed anycast-based services typically employed by large content owners and Content Distribution Networks (CDNs). The challenge is that a migration to a different network in the middle of the connection greatly increases the chances of the packets routed to a different anycast point-of-presence (POP) due to the new network's different connectivity and Internet peering arrangements. The load balancer in the new POP, potentially thousands of miles away, will not have information about

the new flow and would not be able to route it back to the original POP.

To help with the endpoint network migration challenges, anycast service operations are likely to employ integrated load balancers that, in cooperation with their pool servers, are able to ensure that client-to-server packets contain some additional identification in plainly-observable parts of the packets (in addition to the 5-tuple). As noted in Section 2 of [RFC7258], careful consideration in protocol design to mitigate PM is important, while ensuring manageability of the network.

An area for further research includes end-to-end solutions that would provide a simpler architecture and may solve the issue with CDN anycast. In this case, connections would be migrated to a CDN unicast address.

Current protocols, such as TCP, allow the development of stateless integrated load balancers by availing such load balancers of additional plain text information in client-to-server packets. In case of TCP, such information can be encoded by having server-generated sequence numbers (that are ACK'd by the client), segment values, lengths of the packet sent, etc. The use of some of these mechanisms for load balancing negates some of the security assumptions associated with those primitives (e.g., that an off-path attacker guessing valid sequence numbers for a flow is hard). Another possibility is a dedicated mechanism for storing load balancer state, such as QUIC's proposed connection ID to provide visibility to the load balancer. An identifier could be used for tracking purposes, but this may provide an option that is an improvement from bolting it on to an unrelated transport signal. This method allows for tight control by one of the endpoints and can be rotated to avoid roving client linkability: in other words, being a specific, separate signal, it can be governed in a way that is finely targeted at that specific use-case.

Some integrated load balancers have the ability to use additional plainly observable information even for today's protocols that are not network migration tolerant. This additional information allows for improved availability and scalability of the load balancing operation. For example, BGP reconvergence can cause a flow to switch anycast POPs even without a network change by any endpoint. Additionally, a system that is able to encode the identity of the pool server in plain text information available in each incoming packet is able to provide stateless load balancing. This ability confers great reliability and scalability advantages even if the flow remains in a single POP, because the load balancing system is not required to keep state of each flow. Even more importantly, there's

no requirement to continuously synchronize such state among the pool of load balancers. An integrated load balancer repurposing limited existing bits in transport flow state must maintain and synchronize per-flow state occasionally: using the sequence number as a cookie only works for so long given that there aren't that many bits available to divide across a pool of machines.

Mobile operators apply Self Organizing Networks (3GPP SON) for intelligent workflows such as content-aware MLB (Mobility Load Balancing). Where network load balancers have been configured to route according to application-layer semantics, an encrypted payload is effectively invisible. This has resulted in practices of intercepting TLS in front of load balancers to regain that visibility, but at a cost to security and privacy.

In future Network Function Virtualization (NFV) architectures, load balancing functions are likely to be more prevalent (deployed at locations throughout operators' networks). NFV environments will require some type of identifier (IPv6 flow identifiers, the proposed QUIC connection ID, etc.) for managing traffic using encrypted tunnels. The shift to increased encryption will have an impact to visibility of flow information and will require adjustments to perform similar load balancing functions within an NFV.

2.2.2. Differential Treatment based on Deep Packet Inspection (DPI)

Data transfer capacity resources in cellular radio networks tend to be more constrained than in fixed networks. This is a result of variance in radio signal strength as a user moves around a cell, the rapid ingress and egress of connections as users hand off between adjacent cells, and temporary congestion at a cell. Mobile networks alleviate this by queuing traffic according to its required bandwidth and acceptable latency: for example, a user is unlikely to notice a 20ms delay when receiving a simple Web page or email, or an instant message response, but will very likely notice a re-buffering pause in a video playback or a VoIP call de-jitter buffer. Ideally, the scheduler manages the queue so that each user has an acceptable experience as conditions vary, but inferences of the traffic type have been used to make bearer assignments and set scheduler priority.

Deep Packet Inspection (DPI) allows identification of applications based on payload signatures, in contrast to trusting well-known port numbers. Application and transport layer encryption make the traffic type estimation more complex and less accurate, and therefore it may not be effectual to use this information as input for queue management. With the use of WebSockets [RFC6455], for example, many forms of communications (from isochronous/real-time to bulk/elastic file transfer) will take place over HTTP port 80 or port 443, so only

the messages and higher-layer data will make application differentiation possible. If the monitoring system sees only "HTTP port 443", it cannot distinguish application streams that would benefit from priority queueing from others that would not.

Mobile networks especially rely on content/application based prioritization of Over-the-Top (OTT) services - each application-type or service has different delay/loss/throughput expectations, and each type of stream will be unknown to an edge device if encrypted; this impedes dynamic-QoS adaptation. An alternate way to achieve encrypted application separation is possible when the User Equipment (UE) requests a dedicated bearer for the specific application stream (known by the UE), using a mechanism such as the one described in Section 6.5 of 3GPP TS 24.301 [TS3GPP]. The UE's request includes the Quality Class Indicator (QCI) appropriate for each application, based on their different delay/loss/throughput expectations. However, UE requests for dedicated bearers and QCI may not be supported at the subscriber's service level, or in all mobile networks.

These effects and potential alternative solutions have been discussed at the accord BoF [ACCORD] at IETF95.

This section does not consider traffic discrimination by service providers related to NetNeutrality, where traffic may be favored according to the service provider preference as opposed to the user's preference. These use cases are considered out-of-scope for this document as controversial practices.

2.2.3. Network Congestion Management

For User Plane Congestion Management (3GPP UPCON) [UPCON], the ability to understand content and manage networks during periods of congestion is the focus of this 3GPP work item. Mitigating techniques such as deferred download, off-peak acceleration, and outbound roamers are a few examples of the areas explored in the associated 3GPP documents. The documents describe the issues, the data utilized in managing congestion, and make policy recommendations.

2.2.4. Performance-enhancing Proxies

Performance-enhancing TCP proxies may perform local retransmission at the network edge; this also applies to mobile networks. In TCP, duplicated ACKs are detected and potentially concealed when the proxy retransmits a segment that was lost on the mobile link without involvement of the far end (see section 2.1.1 of [RFC3135] and section 3.5 of [I-D.dolson-plus-middlebox-benefits]).

Operators report that this optimization at network edges improves real-time transmission over long delay Internet paths or networks with large capacity-variation (such as mobile/cellular networks). However, such optimizations can also cause problems with performance, for example if the characteristics of some packet streams begin to vary significantly from those considered in the proxy design.

In general some operators have stated that performance-enhancing proxies have a lower Round-Trip Time (RTT) to the client and therefore determine the responsiveness of flow control. A lower RTT makes the flow control loop more responsive to changes in the mobile network conditions and enables faster adaptation in a delay and capacity varying network due to user mobility.

Further, some use service-provider-operated proxies to reduce the control delay between the sender and a receiver on a mobile network where resources are limited. The RTT determines how quickly a user's attempt to cancel a video is recognized and therefore how quickly the traffic is stopped, thus keeping un-wanted video packets from entering the radio scheduler queue. If impacted by encryption, performance enhancing proxies could make use of routing overlay protocols to accomplish the same task, but this results in additional overhead.

An application-type-aware network edge (middlebox) can further control pacing, limit simultaneous HD videos, or prioritize active videos against new videos, etc. Services at this more granular level are limited with the use of encryption.

Performance enhancing proxies are primarily used on long delay links (satellite) with access to the TCP header to provide an early ACK and make the long delay link of the path seem shorter. With some specific forms of flow control, TCP can be more efficient than alternatives such as proxies. The editors cannot cite research on this point specific to the performance enhancing proxies described, but agree this area could be explored to determine if flow-control modifications could preserve the end-to-end performance on long delay paths session where the TCP header is exposed.

2.2.5. Caching and Content Replication Near the Network Edge

The features and efficiency of some Internet services can be augmented through analysis of user flows and the applications they provide. For example, network caching of popular content at a location close to the requesting user can improve delivery efficiency (both in terms of lower request response times and reduced use of International Internet links when content is remotely located), and the service provider through an authorized agreement acting on their

behalf use DPI in combination with content distribution networks to determine if they can intervene effectively. Encryption of packet contents at a given protocol layer usually makes DPI processing of that layer and higher layers impossible. That being said, it should be noted that some content providers prevent caching to control content delivery through the use of encrypted end-to-end sessions. CDNs vary in their deployment options of end-to-end encryption. The business risk of losing control of content is a motivation outside of privacy and pervasive monitoring that are driving end-to-end encryption for these content providers.

It should be noted that caching was first supported in [RFC1945] and continued in the recent update of "Hypertext Transfer Protocol (HTTP/1.1): Caching" in [RFC7234]. Some operators also operate transparent caches which neither the user nor the origin opt-in. The use of these caches is controversial within IETF and is generally precluded by the use of HTTPS.

Content replication in caches (for example live video, Digital Rights Management (DRM) protected content) is used to most efficiently utilize the available limited bandwidth and thereby maximize the user's Quality of Experience (QoE). Especially in mobile networks, duplicating every stream through the transit network increases backhaul cost for live TV. The Enhanced Multimedia Broadcast/Multicast Services (3GPP eMBMS) utilizes trusted edge proxies to facilitate delivering the same stream to different users, using either unicast or multicast depending on channel conditions to the user. There are on-going efforts to support multicast inside carrier networks while preserving end-to-end security: Automatic Multicast Tunneling (AMT), for instance, allows CDNs to deliver a single (potentially encrypted) copy of a live stream to a carrier network over the public internet and for the carrier to then distribute that live stream as efficiently as possible within its own network using multicast.

Alternate approaches are in the early phase of being explored to allow caching of encrypted content. These solutions require cooperation from content owners and fall outside the scope of what is covered in this document. Content delegation allows for replication with possible benefits, but any form of delegation has the potential to affect the expectation of client-server confidentiality.

2.2.6. Content Compression

In addition to caching, various applications exist to provide data compression in order to conserve the life of the user's mobile data plan or make delivery over the mobile link more efficient. The compression proxy access can be built into a specific user level

application, such as a browser, or it can be available to all applications using a system level application. The primary method is for the mobile application to connect to a centralized server as a transparent proxy (user does not opt-in), with the data channel between the client application and the server using compression to minimize bandwidth utilization. The effectiveness of such systems depends on the server having access to unencrypted data flows.

Aggregated data stream content compression that spans objects and data sources that can be treated as part of a unified compression scheme (e.g., through the use of a shared segment store) is often effective at providing data offload when there is a network element close to the receiver that has access to see all the content.

2.2.7. Service Function Chaining

Service Function Chaining (SFC) has been defined in RFC7665 [RFC7665] and RFC8300 [RFC8300]. As discussed in RFC7498 [RFC7498], common SFC deployments may use classifiers to direct traffic into VLANs instead of using NSH, as defined in RFC8300 [RFC8300]. As described in RFC7665 [RFC7665], the ordered steering of traffic to support specific optimizations depends upon the ability of a Classifier to determine the microflows. RFC2474 [RFC2474] defines "Microflow: a single instance of an application-to-application flow of packets which is identified by source address, destination address, protocol id, and source port, destination port (where applicable)." SFC currently depends upon a classifier to at least identify the microflow. As the classifier's visibility is reduced from a 5-tuple to a 2-tuple, or if information above the transport layer becomes inaccessible, then the SFC Classifier is not able to perform its job and the service functions of the path may be adversely affected.

There are also mechanisms provided to protect security and privacy. In the SFC case, the layer below a network service header can be protected with session encryption. A goal is protecting end-user data, while retaining the intended functions of RFC7665 [RFC7665] at the same time.

2.3. Content Filtering, Network Access, and Accounting

Mobile Networks and many ISPs operate under the regulations of their licensing government authority. These regulations include Lawful Intercept, adherence to Codes of Practice on content filtering, and application of court order filters. Such regulations assume network access to provide content filtering and accounting, as discussed below. As previously stated, the intent of this document is to document existing practices; the development of IETF protocols follows the guiding principles of [RFC1984] and [RFC2804] and

explicitly do not support tools and methods that could be used for wiretapping and censorship.

2.3.1. Content Filtering

There are numerous reasons why service providers might block content: to comply with requests from law enforcement or regulatory authorities, to effectuate parental controls, to enforce content-based billing, or for other reasons, possibly considered inappropriate by some. See RFC7754 [RFC7754] for a survey of Internet filtering techniques and motivations and the IAB consensus on those mechanisms. This section is intended to document a selection of current content blocking practices by operators and the effects of encryption on those practices. Content blocking may also happen at endpoints or at the edge of enterprise networks, but those are not addressed in this section.

In a mobile network content filtering usually occurs in the core network. With other networks, content filtering could occur in the core network or at the edge. A proxy is installed which analyses the transport metadata of the content users are viewing and either filters content based on a blacklist of sites or based on the user's pre-defined profile (e.g., for age sensitive content). Although filtering can be done by many methods, one commonly used method involves a trigger based on the proxy identifying a DNS lookup of a host name in a URL which appears on a blacklist being used by the operator. The subsequent requests to that domain will be re-routed to a proxy which checks whether the full URL matches a blocked URL on the list, and will return a 404 if a match is found. All other requests should complete. This technique does not work in situations where DNS traffic is encrypted (e.g., by employing [RFC7858]). This method is also used by other types of network providers enabling traffic inspection, but not modification.

Content filtering via a proxy can also utilize an intercepting certificate where the client's session is terminated at the proxy enabling for cleartext inspection of the traffic. A new session is created from the intercepting device to the client's destination; this is an opt-in strategy for the client, where the endpoint is configured to trust the intercepting certificate. Changes to TLSv1.3 do not impact this more invasive method of interception, that has the potential to expose every HTTPS session to an active man in the middle (MitM).

Another form of content filtering is called parental control, where some users are deliberately denied access to age-sensitive content as a feature to the service subscriber. Some sites involve a mixture of universal and age-sensitive content and filtering software. In these

cases, more granular (application layer) metadata may be used to analyze and block traffic. Methods that accessed cleartext application-layer metadata no longer work when sessions are encrypted. This type of granular filtering could occur at the endpoint or as a proxy service. However, the lack of ability to efficiently manage endpoints as a service reduces network service providers' ability to offer parental control.

2.3.2. Network Access and Data Usage

Approved access to a network is a prerequisite to requests for Internet traffic.

However, there are cases (beyond parental control) when a network service provider currently redirects customer requests for content (affecting content accessibility):

1. The network service provider is performing the accounting and billing for the content provider, and the customer has not (yet) purchased the requested content.
2. Further content may not be allowed as the customer has reached their usage limit and needs to purchase additional data service, which is the usual billing approach in mobile networks.

Currently, some network service providers redirect the customer using HTTP redirect to a captive portal page that explains to those customers the reason for the blockage and the steps to proceed. [RFC6108] describes one viable web notification system. When the HTTP headers and content are encrypted, this appropriately prevents mobile carriers from intercepting the traffic and performing an HTTP redirect. As a result, some mobile carriers block customer's encrypted requests, which impacts customer experience because the blocking reason must be conveyed by some other means. The customer may need to call customer care to find out the reason and/or resolve the issue, possibly extending the time needed to restore their network access. While there are well deployed alternate SMS-based solutions that do not involve out of specification protocol interception, this is still an unsolved problem for non-SMS users.

Further, when the requested service is about to consume the remainder of the user's plan limits, the transmission could be terminated and advance notifications may be sent to the user by their service provider to warn the user ahead of the exhausted plan. If web content is encrypted, the network provider cannot know the data transfer size at request time. Lacking this visibility of the application type and content size, the network would continue the transmission and stop the transfer when the limit was reached. A

partial transfer may not be usable by the client wasting both network and user resources, possibly leading to customer complaints. The content provider does not know user's service plans or current usage, and cannot warn the user of plan exhaustion.

In addition, some mobile network operators sell tariffs that allow free-data access to certain sites, known as 'zero rating'. A session to visit such a site incurs no additional cost or data usage to the user. For some implementations, zero rating is impacted if encryption hides the details of the content domain from the network.

2.3.3. Application Layer Gateways

Application Layer Gateways (ALG) assist applications to set connectivity across Network Address Translators (NAT), Firewalls, and/or Load Balancers for specific applications running across mobile networks. Section 2.9 of [RFC2663] describes the role of ALGs and their interaction with NAT and/or application payloads. ALG are deployed with an aim to improve connectivity. However, it is an IETF Best Common Practice recommendation that ALGs for UDP-based protocols should be turned off [RFC4787].

One example of an ALG in current use is aimed at video applications that use the Real Time Session Protocol (RTSP) [RFC7826] primary stream as a means to identify related Real Time Protocol/Real Time Control Protocol (RTP/RTCP) [RFC3550] flows at set-up. The ALG in this case relies on the 5-tuple flow information derived from RTSP to provision NAT or other middleboxes and provide connectivity. Implementations vary, and two examples follow:

1. Parse the content of the RTSP stream and identify the 5-tuple of the supporting streams as they are being negotiated.
2. Intercept and modify the 5-tuple information of the supporting media streams as they are being negotiated on the RTSP stream, which is more intrusive to the media streams.

When RTSP stream content is encrypted, the 5-tuple information within the payload is not visible to these ALG implementations, and therefore they cannot provision their associated middleboxes with that information.

The deployment of IPv6 may well reduce the need for NAT, and the corresponding requirement for Application Layer Gateways.

2.3.4. HTTP Header Insertion

Some mobile carriers use HTTP header insertion (see section 3.2.1 of [RFC7230]) to provide information about their customers to third parties or to their own internal systems [Enrich]. Third parties use the inserted information for analytics, customization, advertising, cross-site tracking of users, to bill the customer, or to selectively allow or block content. HTTP header insertion is also used to pass information internally between a mobile service provider's sub-systems, thus keeping the internal systems loosely coupled. When HTTP connections are encrypted to protect users privacy, mobile network service providers cannot insert headers to accomplish the, sometimes considered controversial, functions above.

Guidance from the Internet Architecture Board has been provided in RFC8165 [RFC8165] on Design Considerations for Metadata Insertion. The guidance asserts that designs that share metadata only by explicit actions at the host are preferable to designs in which middleboxes insert metadata. Alternate notification methods that follow this and other guidance would be helpful to mobile carriers.

3. Encryption in Hosting and Application SP Environments

Hosted environments have had varied requirements in the past for encryption, with many businesses choosing to use these services primarily for data and applications that are not business or privacy sensitive. A shift prior to the revelations on surveillance/passive monitoring began where businesses were asking for hosted environments to provide higher levels of security so that additional applications and service could be hosted externally. Businesses understanding the threats of monitoring in hosted environments increased that pressure to provide more secure access and session encryption to protect the management of hosted environments as well as for the data and applications.

3.1. Management Access Security

Hosted environments may have multiple levels of management access, where some may be strictly for the Hosting SP (infrastructure that may be shared among customers) and some may be accessed by a specific customer for application management. In some cases, there are multiple levels of hosting service providers, further complicating the security of management infrastructure and the associated requirements.

Hosting service provider management access is typically segregated from other traffic with a control channel and may or may not be encrypted depending upon the isolation characteristics of the

management session. Customer access may be through a dedicated connection, but discussion for that connection method is out-of-scope for this document.

In overlay networks (e.g. VXLAN, Geneve, etc.) that are used to provide hosted services, management access for a customer to support application management may depend upon the security mechanisms available as part of that overlay network. While overlay network data encapsulations may be used to indicate the desired isolation, this is not sufficient to prevent deliberate attacks that are aware of the use of the overlay network.

[I-D.mglt-nvo3-geneve-security-requirements] describes requirements to handle attacks. It is possible to use an overlay header in combination with IPsec or other encrypted traffic sessions, but this adds the requirement for authentication infrastructure and may reduce packet transfer performance. The use of an overlay header may also be deployed as a mechanism to manage encrypted traffic streams on the network by network service providers. Additional extension mechanisms to provide integrity and/or privacy protections are being investigated for overlay encapsulations. Section 7 of [RFC7348] describes some of the security issues possible when deploying VXLAN on Layer 2 networks. Rogue endpoints can join the multicast groups that carry broadcast traffic, for example.

3.1.1. Customer Access Monitoring

Hosted applications that allow some level of customer management access may also require monitoring by the hosting service provider. Monitoring could include access control restrictions such as authentication, authorization, and accounting for filtering and firewall rules to ensure they are continuously met. Customer access may occur on multiple levels, including user-level and administrative access. The hosting service provider may need to monitor access either through session monitoring or log evaluation to ensure security service level agreements (SLA) for access management are met. The use of session encryption to access hosted environments limits access restrictions to the metadata described below. Monitoring and filtering may occur at an:

2-tuple IP-level with source and destination IP addresses alone, or

5-tuple IP and protocol-level with source IP address, destination IP address, protocol number, source port number, and destination port number.

Session encryption at the application level, TLS for example, currently allows access to the 5-tuple. IP-level encryption, such as IPsec in tunnel mode prevents access to the original 5-tuple and may

limit the ability to restrict traffic via filtering techniques. This shift may not impact all hosting service provider solutions as alternate controls may be used to authenticate sessions or access may require that clients access such services by first connecting to the organization before accessing the hosted application. Shifts in access may be required to maintain equivalent access control management. Logs may also be used for monitoring that access control restrictions are met, but would be limited to the data that could be observed due to encryption at the point of log generation. Log analysis is out of scope for this document.

3.1.2. SP Content Monitoring of Applications

The following observations apply to any IT organization that is responsible for delivering services, whether to third-parties, for example as a web based service, or to internal customers in an enterprise, e.g. a data processing system that forms a part of the enterprise's business.

Organizations responsible for the operation of a data center have many processes which access the contents of IP packets (passive methods of measurement, as defined in [RFC7799]). These processes are typically for service assurance or security purposes as part of their data center operations.

Examples include:

- Network Performance Monitoring/Application Performance Monitoring
- Intrusion defense/prevention systems
- Malware detection
- Fraud Monitoring
- Application DDOS protection
- Cyber-attack investigation
- Proof of regulatory compliance
- Data Leakage Prevention

Many application service providers simply terminate sessions to/from the Internet at the edge of the data center in the form of SSL/TLS offload in the load balancer. Not only does this reduce the load on

application servers, it simplifies the processes to enable monitoring of the session content.

However, in some situations, encryption deeper in the data center may be necessary to protect personal information or in order to meet industry regulations, e.g. those set out by the Payment Card Industry (PCI). In such situations, various methods have been used to allow service assurance and security processes to access unencrypted data. These include SSL/TLS decryption in dedicated units, which then forward packets to SP-controlled tools, or by real-time or post-capture decryption in the tools themselves. A number of these tools provide passive decryption by providing the monitoring device with the server's private key. The move to increased use of forward-secret key exchange mechanism impacts the use of these techniques.

Data center operators may also maintain packet recordings in order to be able to investigate attacks, breach of internal processes, etc. In some industries, organizations may be legally required to maintain such information for compliance purposes. Investigations of this nature have used access to the unencrypted contents of the packet. Alternate methods to investigate attacks or breach of process will rely on endpoint information, such as logs. As previously noted, logs often lack complete information, and this is seen as a concern resulting in some relying on session access for additional information.

Application Service Providers may offer content-level monitoring options to detect intellectual property leakage, or other attacks. In service provider environments where Data Loss Prevention (DLP) has been implemented on the basis of the service provider having cleartext access to session streams, the use of encrypted streams prevents these implementations from conducting content searches for the keywords or phrases configured in the DLP system. DLP is often used to prevent the leakage of Personally Identifiable Information (PII) as well as financial account information, Personal Health Information (PHI), and Payment Card Information (PCI). If session encryption is terminated at a gateway prior to accessing these services, DLP on session data can still be performed. The decision of where to terminate encryption to hosted environments will be a risk decision made between the application service provider and customer organization according to their priorities. DLP can be performed at the server for the hosted application and on an end user's system in an organization as alternate or additional monitoring points of content; however, this is not frequently done in a service provider environment.

Application service providers, by their very nature, control the application endpoint. As such, much of the information gleaned from

sessions are still available on that endpoint. However, when a gap exists in the application's logging and debugging capabilities, this has led the application service provider to access data-in-transport for monitoring and debugging.

3.2. Hosted Applications

Organizations are increasingly using hosted applications rather than in-house solutions that require maintenance of equipment and software. Examples include Enterprise Resource Planning (ERP) solutions, payroll service, time and attendance, travel and expense reporting among others. Organizations may require some level of management access to these hosted applications and will typically require session encryption or a dedicated channel for this activity.

In other cases, hosted applications may be fully managed by a hosting service provider with service level agreement expectations for availability and performance as well as for security functions including malware detection. Due to the sensitive nature of these hosted environments, the use of encryption is already prevalent. Any impact may be similar to an enterprise with tools being used inside of the hosted environment to monitor traffic. Additional concerns were not reported in the call for contributions.

3.2.1. Monitoring Managed Applications

Performance, availability, and other aspects of a SLA are often collected through passive monitoring. For example:

- o Availability: ability to establish connections with hosts to access applications, and discern the difference between network or host-related causes of unavailability.
- o Performance: ability to complete transactions within a target response time, and discern the difference between network or host-related causes of excess response time.

Here, as with all passive monitoring, the accuracy of inferences are dependent on the cleartext information available, and encryption would tend to reduce the information and therefore, the accuracy of each inference. Passive measurement of some metrics will be impossible with encryption that prevents inferring packet correspondence across multiple observation points, such as for packet loss metrics.

Application logging currently lacks detail sufficient to make accurate inferences in an environment with increased encryption, and

so this constitutes a gap for passive performance monitoring (which could be closed if log details are enhanced in the future).

3.2.2. Mail Service Providers

Mail (application) service providers vary in what services they offer. Options may include a fully hosted solution where mail is stored external to an organization's environment on mail service provider equipment or the service offering may be limited to monitor incoming mail to remove spam [Section 5.1], malware [Section 5.6], and phishing attacks [Section 5.3] before mail is directed to the organization's equipment. In both of these cases, content of the messages and headers is monitored to detect spam, malware, phishing, and other messages that may be considered an attack.

STARTTLS should have zero effect on anti-spam efforts for SMTP traffic. Anti-spam services could easily be performed on an SMTP gateway, eliminating the need for TLS decryption services. The impact to anti-spam service providers should be limited to a change in tools, where middleboxes were deployed to perform these functions.

Many efforts are emerging to improve user-to-user encryption, including promotion of PGP and newer efforts such as Dark Mail [DarkMail]. Of course, content-based spam filtering will not be possible on encrypted content.

3.3. Data Storage

Numerous service offerings exist that provide hosted storage solutions. This section describes the various offerings and details the monitoring for each type of service and how encryption may impact the operational and security monitoring performed.

Trends in data storage encryption for hosted environments include a range of options. The following list is intentionally high-level to describe the types of encryption used in coordination with data storage that may be hosted remotely, meaning the storage is physically located in an external data center requiring transport over the Internet. Options for monitoring will vary with each encryption approach described below. In most cases, solutions have been identified to provide encryption while ensuring management capabilities were maintained through logging or other means.

3.3.1. Object-level Encryption

For higher security and/or privacy of data and applications, options that provide end-to-end encryption of the data from the user's desktop or server to the storage platform may be preferred. This

description includes any solution that encrypts data at the object level, not transport level. Encryption of data may be performed with libraries on the system or at the application level, which includes file encryption services via a file manager. Object-level encryption is useful when data storage is hosted, or scenarios when the storage location is determined based on capacity or based on a set of parameters to automate decisions. This could mean that large data sets accessed infrequently could be sent to an off-site storage platform at an external hosting service, data accessed frequently may be stored locally, or the decision could be based on the transaction type. Object-level encryption is grouped separately for the purpose of this document since data may be stored in multiple locations including off-site remote storage platforms. If session encryption is also used, the protocol is likely to be TLS.

Impacts to monitoring may include access to content inspection for data leakage prevention and similar technologies, depending on their placement in the network.

3.3.1.1. Monitoring for Hosted Storage

Monitoring of hosted storage solutions that use host-level (object) encryption is described in this subsection. Host-level encryption can be employed for backup services, and occasionally for external storage services (operated by a third party) when internal storage limits are exceeded.

Monitoring of data flows to hosted storage solutions is performed for security and operational purposes. The security monitoring may be to detect anomalies in the data flows that could include changes to destination, the amount of data transferred, or alterations in the size and frequency of flows. Operational considerations include capacity and availability monitoring.

3.3.2. Disk Encryption, Data at Rest

There are multiple ways to achieve full disk encryption for stored data. Encryption may be performed on data to be stored while in transit close to the storage media with solutions like Controller Based Encryption (CBE) or in the drive system with Self-Encrypting Drives (SED). Session encryption is typically coupled with encryption of these data at rest (DAR) solutions to also protect data in transit. Transport encryption is likely via TLS.

3.3.2.1. Monitoring Session Flows for Data at Rest (DAR) Solutions

Monitoring for transport of data to storage platforms, where object level encryption is performed close to or on the storage platform are similar to those described in the section on Monitoring for Hosted Storage. The primary difference for these solutions is the possible exposure of sensitive information, which could include privacy related data, financial information, or intellectual property if session encryption via TLS is not deployed. Session encryption is typically used with these solutions, but that decision would be based on a risk assessment. There are use cases where DAR or disk-level encryption is required. Examples include preventing exposure of data if physical disks are stolen or lost. In the case where TLS is in use, monitoring and the exposure of data is limited to a 5-tuple.

3.3.3. Cross Data Center Replication Services

Storage services also include data replication which may occur between data centers and may leverage Internet connections to tunnel traffic. The traffic may use iSCSI [RFC7143] or FC/IP [RFC7146] encapsulated in IPsec. Either transport or tunnel mode may be used for IPsec depending upon the termination points of the IPsec session, if it is from the storage platform itself or from a gateway device at the edge of the data center respectively.

3.3.3.1. Monitoring Of IPsec for Data Replication Services

Monitoring of data flows between data centers (for data replication) may be performed for security and operational purposes and would typically concentrate more on operational aspects since these flows are essentially virtual private networks (VPN) between data centers. Operational considerations include capacity and availability monitoring. The security monitoring may be to detect anomalies in the data flows, similar to what was described in the "Monitoring for Hosted Storage Section". If IPsec tunnel mode is in use, monitoring is limited to a 2-tuple, or with transport mode, a 5-tuple.

4. Encryption for Enterprises

Encryption of network traffic within the private enterprise is a growing trend, particularly in industries with audit and regulatory requirements. Some enterprise internal networks are almost completely TLS and/or IPsec encrypted.

For each type of monitoring, different techniques and access to parts of the data stream are part of current practice. As we transition to an increased use of encryption, alternate methods of monitoring for operational purposes may be necessary to reduce the practice of

breaking encryption (other policies may apply in some enterprise settings).

4.1. Monitoring Practices of the Enterprise

Large corporate enterprises are the owners of the platforms, data, and network infrastructure that provide critical business services to their user communities. As such, these enterprises are responsible for all aspects of the performance, availability, security, and quality of experience for all user sessions. In many such enterprises, users are required to consent to the enterprise monitoring all their activities as a condition of employment. Subsections of 4. Encryption for Enterprises may discuss techniques that access data beyond the data-link, network, and transport level headers typically used in SP networks since the corporate enterprise owns the data. These responsibilities break down into three basic areas:

1. Security Monitoring and Control
2. Application Performance Monitoring and Reporting
3. Network Diagnostics and Troubleshooting

In each of the above areas, technical support teams utilize collection, monitoring, and diagnostic systems. Some organizations currently use attack methods such as replicated TLS server RSA private keys to decrypt passively monitored copies of encrypted TLS packet streams.

For an enterprise to avoid costly application down time and deliver expected levels of performance, protection, and availability, some forms of traffic analysis, sometimes including examination of packet payloads, are currently used.

4.1.1. Security Monitoring in the Enterprise

Enterprise users are subject to the policies of their organization and the jurisdictions in which the enterprise operates. As such, proxies may be in use to:

1. intercept outbound session traffic to monitor for intellectual property leakage (by users, malware, and trojans),
2. detect viruses/malware entering the network via email or web traffic,

3. detect malware/Trojans in action, possibly connecting to remote hosts,
4. detect attacks (Cross site scripting and other common web related attacks),
5. track misuse and abuse by employees,
6. restrict the types of protocols permitted to/from the entire corporate environment,
7. detect and defend against Internet DDoS attacks, including both volumetric and layer 7 attacks.

A significant portion of malware hides its activity within TLS or other encryption protocols. This includes lateral movement, Command and Control, and Data Exfiltration.

The impact to a fully encrypted internal network would include cost and possible loss of detection capabilities associated with the transformation of the network architecture and tools for monitoring. The capabilities of detection through traffic fingerprinting, logs, host-level transaction monitoring, and flow analysis would vary depending on access to a 2-tuple or 5-tuple in the network as well.

Security monitoring in the enterprise may also be performed at the endpoint with numerous current solutions that mitigate the same problems as some of the above mentioned solutions. Since the software agents operate on the device, they are able to monitor traffic before it is encrypted, monitor for behavior changes, and lock down devices to use only the expected set of applications. Session encryption does not affect these solutions. Some might argue that scaling is an issue in the enterprise, but some large enterprises have used these tools effectively.

Use of Bring-your-own-device (BYOD) policies within organizations may limit the scope of monitoring permitted with these alternate solutions. Network endpoint assessment (NEA) or the use of virtual hosts could help to bridge the monitoring gap.

4.1.2. Application Performance Monitoring in the Enterprise

There are two main goals of monitoring:

1. Assess traffic volume on a per-application basis, for billing, capacity planning, optimization of geographical location for servers or proxies, and other goals.

2. Assess performance in terms of application response time and user perceived response time.

Network-based Application Performance Monitoring tracks application response time by user and by URL, which is the information that the application owners and the lines of business request. Content Delivery Networks (CDNs) add complexity in determining the ultimate endpoint destination. By their very nature, such information is obscured by CDNs and encrypted protocols -- adding a new challenge for troubleshooting network and application problems. URL identification allows the application support team to do granular, code level troubleshooting at multiple tiers of an application.

New methodologies to monitor user perceived response time and to separate network from server time are evolving. For example, the IPv6 Destination Option Header (DOH) implementation of Performance and Diagnostic Metrics (PDM) will provide this [RFC8250]. Using PDM with IPsec Encapsulating Security Payload (ESP) Transport Mode requires placement of the PDM DOH within the ESP encrypted payload to avoid leaking timing and sequence number information that could be useful to an attacker. Use of PDM DOH also may introduce some security weaknesses, including a timing attack, as described in Section 7 of [RFC8250]. For these and other reasons, [RFC8250] requires that the PDM DOH option be explicitly turned on by administrative action in each host where this measurement feature will be used.

4.1.3. Enterprise Network Diagnostics and Troubleshooting

One primary key to network troubleshooting is the ability to follow a transaction through the various tiers of an application in order to isolate the fault domain. A variety of factors relating to the structure of the modern data center and multi-tiered application have made it difficult to follow a transaction in network traces without the ability to examine some of the packet payload. Alternate methods, such as log analysis need improvement to fill this gap.

4.1.3.1. Address Sharing (NAT)

Content Delivery Networks (CDNs) and NATs and Network Address and Port Translators (NAPT) obscure the ultimate endpoint designation (See [RFC6269] for types of address sharing and a list of issues). Troubleshooting a problem for a specific end user requires finding information such as the IP address and other identifying information so that their problem can be resolved in a timely manner.

NAT is also frequently used by lower layers of the data center infrastructure. Firewalls, Load Balancers, Web Servers, App Servers,

and Middleware servers all regularly NAT the source IP of packets. Combine this with the fact that users are often allocated randomly by load balancers to all these devices, the network troubleshooter is often left with very few options in today's environment due to poor logging implementations in applications. As such, network troubleshooting is used to trace packets at a particular layer, decrypt them, and look at the payload to find a user session.

This kind of bulk packet capture and bulk decryption is frequently used when troubleshooting a large and complex application. Endpoints typically don't have the capacity to handle this level of network packet capture, so out-of-band networks of robust packet brokers and network sniffers that use techniques such as copies of TLS RSA private keys accomplish this task today.

4.1.3.2. TCP Pipelining/Session Multiplexing

TCP pipelining/session multiplexing used mainly by middleboxes today allows for multiple end user sessions to share the same TCP connection. This raises several points of interest with an increased use of encryption. TCP session multiplexing should still be possible when TLS or TCPcrypt is in use since the TCP header information is exposed leaving the 5-tuple accessible. The use of TCP session multiplexing of an IP layer encryption, e.g. IPsec, that only exposes a 2-tuple would not be possible. Troubleshooting capabilities with encrypted sessions from the middlebox may limit troubleshooting to the use of logs from the end points performing the TCP multiplexing or from the middleboxes prior to any additional encryption that may be added to tunnel the TCP multiplexed traffic.

Increased use of HTTP/2 will likely further increase the prevalence of session multiplexing, both on the Internet and in the private data center. HTTP pipelining requires both the client and server to participate; visibility of packets once encrypted will hide the use of HTTP pipelining for any monitoring that takes place outside of the endpoint or proxy solution. Since HTTP pipelining is between a client and server, logging capabilities may require improvement in some servers and clients for debugging purposes if this is not already possible. Visibility for middleboxes includes anything exposed by TLS and the 5-tuple.

4.1.3.3. HTTP Service Calls

When an application server makes an HTTP service call to back end services on behalf of a user session, it uses a completely different URL and a completely different TCP connection. Troubleshooting via network trace involves matching up the user request with the HTTP service call. Some organizations do this today by decrypting the TLS

packet and inspecting the payload. Logging has not been adequate for their purposes.

4.1.3.4. Application Layer Data

Many applications use text formats such as XML to transport data or application level information. When transaction failures occur and the logs are inadequate to determine the cause, network and application teams work together, each having a different view of the transaction failure. Using this troubleshooting method, the network packet is correlated with the actual problem experienced by an application to find a root cause. The inability to access the payload prevents this method of troubleshooting.

4.2. Techniques for Monitoring Internet Session Traffic

Corporate networks commonly monitor outbound session traffic to detect or prevent attacks as well as to guarantee service level expectations. In some cases, alternate options are available when encryption is in use through a proxy or a shift to monitoring at the endpoint. In both cases, scaling is a concern and advancements to support this shift in monitoring practices will assist the deployment of end-to-end encryption.

Some DLP tools intercept traffic at the Internet gateway or proxy services with the ability to man-in-the-middle (MiTM) encrypted session traffic (HTTP/TLS). These tools may monitor for key words important to the enterprise including business sensitive information such as trade secrets, financial data, personally identifiable information (PII), or personal health information (PHI). Various techniques are used to intercept HTTP/TLS sessions for DLP and other purposes, and can be misused as described in "Summarizing Known Attacks on TLS and DTLS" [RFC7457] Section 2.8. Note: many corporate policies allow access to personal financial and other sites for users without interception. Another option is to terminate a TLS session prior to the point where monitoring is performed. Aside from exposing user information to the enterprise MITM devices often are subject to severe security defects which can lead to exposure of user data to attackers outside the enterprise UserData [UserData]. In addition, implementation errors in middleboxes have led to major difficulties in deploying new versions of security protocols such as TLS [Ben17a][Ben17b][Res17a][Res17b]

Monitoring traffic patterns for anomalous behavior such as increased flows of traffic that could be bursty at odd times or flows to unusual destinations (small or large amounts of traffic) is common. This traffic may or may not be encrypted and various methods of encryption or just obfuscation may be used.

Web filtering devices are sometimes used to allow only access to well-known sites found to be legitimate and free of malware on last check by a web filtering service company. One common example of web filtering in a corporate environment is blocking access to sites that are not well-known to these tools for the purpose of blocking malware; this may be noticeable to those in research who are unable to access colleague's individual sites or new web sites that have not yet been screened. In situations where new sites are required for access, they can typically be added after notification by the user or log alerts and review. Home mail account access may be blocked in corporate settings to prevent another vector for malware to enter as well as for intellectual property to leak out of the network. This method remains functional with increased use of encryption and may be more effective at preventing malware from entering the network. Some enterprises may be more aggressive in their filtering and monitoring policy, causing undesirable outcomes. Web filtering solutions monitor and potentially restrict access based on the destination URL when available, server name, IP address, or the DNS name. A complete URL may be used in cases where access restrictions vary for content on a particular site or for the sites hosted on a particular server. In some cases, the enterprise may use a proxy to access this additional information based on their policy. This type of restriction is intended to be transparent to users in a corporate setting as the typical corporate user does not access sites which are not well-known to these tools. However, the mechanisms which these web filters use to do monitoring and enforcement have the potential to cause access issues or other user-visible failures.

Desktop DLP tools are used in some corporate environments as well. Since these tools reside on the desktop, they can intercept traffic before it is encrypted and may provide a continued method of monitoring intellectual property leakage from the desktop to the Internet or attached devices.

DLP tools can also be deployed by Network Service providers, as they have the vantage point of monitoring all traffic paired with destinations off the enterprise network. This makes an effective solution for enterprises that allow "bring-your-own" devices when the traffic is not encrypted, and for devices outside the desktop category (such as mobile phones) that are used on corporate networks nonetheless.

Enterprises may wish to reduce the traffic on their Internet access facilities by monitoring requests for within-policy content and caching it. In this case, repeated requests for Internet content spawned by URLs in e-mail trade newsletters or other sources can be served within the enterprise network. Gradual deployment of end to end encryption would tend to reduce the cacheable content over time,

owing to concealment of critical headers and payloads. Many forms of enterprise performance management may be similarly affected. It should be noted that transparent caching is considered an anti-pattern.

5. Security Monitoring for Specific Attack Types

Effective incident response today requires collaboration at Internet scale. This section will only focus on efforts of collaboration at Internet scale that are dedicated to specific attack types. They may require new monitoring and detection techniques in an increasingly encrypted Internet. As mentioned previously, some service providers have been interfering with STARTTLS to prevent session encryption to be able to perform functions they are used to (injecting ads, monitoring, etc.). By detailing the current monitoring methods used for attack detection and response, this information can be used to devise new monitoring methods that will be effective in the changed Internet via collaboration and innovation.

Changes to improve encryption or to deploy OS methods have little impact on the detection of malicious actors. Malicious actors have had access to strong encryption for quite some time. Incident responders, in many cases, have developed techniques to locate malicious traffic within encrypted sessions. The following section will note some examples where detection and mitigation of such traffic has been successful.

5.1. Mail Abuse and spam

The largest operational effort to prevent mail abuse is through the Messaging, Malware, Mobile Anti-Abuse Working Group (M3AAWG)[M3AAWG]. Mail abuse is combatted directly with mail administrators who can shut down or stop continued mail abuse originating from large scale providers that participate in using the Abuse Reporting Format (ARF) agents standardized in the IETF [RFC5965], [RFC6430], [RFC6590], [RFC6591], [RFC6650], [RFC6651], and [RFC6652]. The ARF agent directly reports abuse messages to the appropriate service provider who can take action to stop or mitigate the abuse. Since this technique uses the actual message, the use of SMTP over TLS between mail gateways will not affect its usefulness. As mentioned previously, SMTP over TLS only protects data while in transit and the messages may be exposed on mail servers or mail gateways if a user-to-user encryption method is not used. Current user-to-user message encryption methods on email (S/MIME and PGP) do not encrypt the email header information used by ARF and the service provider operators in their abuse mitigation efforts.

Another effort, Domain-based Message Authentication, Reporting, and Conformance (DMARC) [RFC7489] is a mechanism for policy distribution that enables increasingly strict handling of messages that fail authentication checks, ranging from no action, through altered delivery, up to message rejection. DMARC is also not affected by the use of STARTTLS.

5.2. Denial of Service

Response to Denial of Service (DoS) attacks are typically coordinated by the SP community with a few key vendors who have tools to assist in the mitigation efforts. Traffic patterns are determined from each DoS attack to stop or rate limit the traffic flows with patterns unique to that DoS attack.

Data types used in monitoring traffic for DDoS are described in the DDoS Open Threat Signaling (DOTS) [DOTS] working group documents in development. The impact of encryption can be understood from their documented use cases[I-D.ietf-dots-use-cases].

Data types used in DDoS attacks have been detailed in the IODEF Guidance draft [RFC8274], Appendix A.2, with the help of several members of the service provider community. The examples provided are intended to help identify the useful data in detecting and mitigating these attacks independent of the transport and protocol descriptions in the drafts.

5.3. Phishing

Investigations and response to phishing attacks follow well-known patterns, requiring access to specific fields in email headers as well as content from the body of the message. When reporting phishing attacks, the recipient has access to each field as well as the body to make content reporting possible, even when end-to-end encryption is used. The email header information is useful to identify the mail servers and accounts used to generate or relay the attack messages in order to take the appropriate actions. The content of the message often contains an embedded attack that may be in an infected file or may be a link that results in the download of malware to the user's system.

Administrators often find it helpful to use header information to track down similar message in their mail queue or users inboxes to prevent further infection. Combinations of To:, From:, Subject:, Received: from header information might be used for this purpose. Administrators may also search for document attachments of the same name, size, or containing a file with a matching hash to a known phishing attack. Administrators might also add URLs contained in

messages to block lists locally or this may also be done by browser vendors through larger scales efforts like that of the Anti-Phishing Working Group (APWG). See the Coordinating Attack Response at Internet Scale (CARIS) workshop Report [RFC8073] for additional information and pointers to the APWG's efforts on anti-phishing.

A full list of the fields used in phishing attack incident response can be found in RFC5901. Future plans to increase privacy protections may limit some of these capabilities if some email header fields are encrypted, such as To:, From:, and Subject: header fields. This does not mean that those fields should not be encrypted, only that we should be aware of how they are currently used.

Some products protect users from phishing by maintaining lists of known phishing domains (such as misspelled bank names) and blocking access. This can be done by observing DNS, clear-text HTTP, or server name indication (SNI) in TLS, in addition to analyzing email. Alternate options to detect and prevent phishing attacks may be needed. More recent examples of data exchanged in spear phishing attacks has been detailed in the IODEF Guidance draft [RFC8274], Appendix A.3.

5.4. Botnets

Botnet detection and mitigation is complex as botnets may involve hundreds or thousands of hosts with numerous Command and Control (C&C) servers. The techniques and data used to monitor and detect each may vary. Connections to C&C servers are typically encrypted, therefore a move to an increasingly encrypted Internet may not affect the detection and sharing methods used.

5.5. Malware

Malware monitoring and detection techniques vary. As mentioned in the enterprise section, malware monitoring may occur at gateways to the organization analyzing email and web traffic. These services can also be provided by service providers, changing the scale and location of this type of monitoring. Additionally, incident responders may identify attributes unique to types of malware to help track down instances by their communication patterns on the Internet or by alterations to hosts and servers.

Data types used in malware investigations have been summarized in an example of the IODEF Guidance draft [RFC8274], Appendix A.1.

5.6. Spoofed Source IP Address Protection

The IETF has reacted to spoofed source IP address-based attacks, recommending the use of network ingress filtering BCP 38 [RFC2827] and the unicast Reverse Path Forwarding (uRPF) mechanism [RFC2504]. But uRPF suffers from limitations regarding its granularity: a malicious node can still use a spoofed IP address included inside the prefix assigned to its link. The Source Address Validation Improvements (SAVI) mechanisms try to solve this issue. Basically, a SAVI mechanism is based on the monitoring of a specific address assignment/management protocol (e.g., SLAAC [RFC4862], SEND [RFC3971], DHCPv4/v6 [RFC2131][RFC3315]) and, according to this monitoring, set-up a filtering policy allowing only the IP flows with a correct source IP address (i.e., any packet with a source IP address, from a node not owning it, is dropped). The encryption of parts of the address assignment/management protocols, critical for SAVI mechanisms, can result in a dysfunction of the SAVI mechanisms.

5.7. Further work

Although incident response work will continue, new methods to prevent system compromise through security automation and continuous monitoring [SACM] may provide alternate approaches where system security is maintained as a preventative measure.

6. Application-based Flow Information Visible to a Network

This section describes specific techniques used in monitoring applications that is visible to the network if a 5-tuple is exposed and as such can potentially be used as an input for future network management approaches. It also includes an overview of IPFIX, a flow-based protocol used to export information about network flows.

6.1. IP Flow Information Export

Many of the accounting, monitoring and measurement tasks described in this document, especially Section 2.3.2, Section 3.1.1, Section 4.1.3, Section 4.2, and Section 5.2 use the IPFIX protocol [RFC7011] for export and storage of the monitored information. IPFIX evolved from the widely-deployed NetFlow protocol [RFC3954], which exports information about flows identified by 5-tuple. While NetFlow was largely concerned with exporting per-flow byte and packet counts for accounting purposes, IPFIX's extensible information model [RFC7012] provides a variety of Information Elements (IEs) [IPFIX-IANA] for representing information above and below the traditional network layer flow information. Enterprise-specific IEs allow exporter vendors to define their own non-standard IEs, as well,

and many of these are driven by header and payload inspection at the metering process.

While the deployment of encryption has no direct effect on the use of IPFIX, certain defined IEs may become unavailable when the metering process observing the traffic cannot decrypt formerly cleartext information. For example, HTTPS renders HTTP header analysis impossible, so IEs derived from the header (e.g. `httpContentType`, `httpUserAgent`) cannot be exported.

The collection of IPFIX data itself, of course, provides a point of centralization for potentially business- and privacy-critical information. The IPFIX File Format specification [RFC5655] recommends encryption for this data at rest, and the IP Flow Anonymization specification [RFC6235] defines a metadata format for describing the anonymization functions applied to an IPFIX dataset, if anonymization is employed for data sharing of IPFIX information between enterprises or network operators.

6.2. TLS Server Name Indication

When initiating the TLS handshake, the Client may provide an extension field (`server_name`) which indicates the server to which it is attempting a secure connection. TLS SNI was standardized in 2003 to enable servers to present the "correct TLS certificate" to clients in a deployment of multiple virtual servers hosted by the same server infrastructure and IP-address. Although this is an optional extension, it is today supported by all modern browsers, web servers and developer libraries. Akamai [Nygren] reports that many of their customer see client TLS SNI usage over 99%. It should be noted that HTTP/2 introduces the Alt-SVC method for upgrading the connection from HTTP/1 to either unencrypted or encrypted HTTP/2. If the initial HTTP/1 request is unencrypted, the destination alternate service name can be identified before the communication is potentially upgraded to encrypted HTTP/2 transport. HTTP/2 requires the TLS implementation to support the Server Name Indication (SNI) extension (see section 9.2 of [RFC7540]). It is also worth noting that [RFC7838] "allows an origin server to nominate additional means of interacting with it on the network", while [RFC8164] allows for a URI to be accessed with HTTP/2 and TLS using Opportunistic Security (on an experimental basis).

This information is only available if the client populates the Server Name Indication extension. Doing so is an optional part of the TLS standard and as stated above this has been implemented by all major browsers. Due to its optional nature, though, existing network filters that examine a TLS ClientHello for a SNI extension cannot expect to always find one. The SNI Encryption in TLS Through

Tunneling [I-D.ietf-tls-sni-encryption] draft has been adopted by the TLS working group, which provides solutions to encrypt SNI. As such, there will be an option to encrypt SNI in future versions of TLS. The per-domain nature of SNI may not reveal the specific service or media type being accessed, especially where the domain is of a provider offering a range of email, video, Web pages etc. For example, certain blog or social network feeds may be deemed 'adult content', but the Server Name Indication will only indicate the server domain rather than a URL path.

There are additional issues for identification of content using SNI: [RFC7540] includes connection coalescing, [I-D.ietf-httpbis-origin-frame] defines the ORIGIN frame, and the [I-D.bishop-httpbis-http2-additional-certs] proposal will increase the difficulty of passive monitoring.

6.3. Application Layer Protocol Negotiation (ALPN)

ALPN is a TLS extension which may be used to indicate the application protocol within the TLS session. This is likely to be of more value to the network where it indicates a protocol dedicated to a particular traffic type (such as video streaming) rather than a multi-use protocol. ALPN is used as part of HTTP/2 'h2', but will not indicate the traffic types which may make up streams within an HTTP/2 multiplex. ALPN is sent clear text in the ClientHello and the server returns it in Encrypted Extensions in TLS 1.3.

6.4. Content Length, BitRate and Pacing

The content length of encrypted traffic is effectively the same as that of the cleartext. Although block ciphers utilize padding, this makes a negligible difference. Bitrate and pacing are generally application specific, and do not change much when the content is encrypted. Multiplexed formats (such as HTTP/2 and QUIC [QUIC]) may however incorporate several application streams over one connection, which makes the bitrate/pacing no longer application-specific. Also, packet padding is available in HTTP/2, TLS 1.3, and many other protocols. Traffic analysis is made more difficult by such countermeasures.

7. Effect of Encryption on Mobile Network Evolution

Transport header encryption prevents the use of transit proxies in center of the network and the use of some edge proxies by preventing the proxies from taking action on the stream. It may be that the claimed benefits of such proxies could be achieved by end-to-end client and server optimizations, distribution using CDNs, plus the ability to continue connections across different access technologies

(across dynamic user IP addresses). The following aspects should be considered in this approach:

1. In a wireless mobile network, the delay and channel capacity per user and sector varies due to coverage, contention, user mobility, scheduling balances fairness, capacity, and service QoE. If most users are at the cell edge, the controller cannot use more complex QAM, thus reducing total cell capacity; similarly if a UMTS edge is serving some number of CS-Voice Calls, the remaining capacity for packet services is reduced.
 2. Mobile wireless networks service in-bound roamers (Users of Operator A in a foreign operator Network B) by backhauling their traffic through Operator B's network to Operator A's Network and then serving through the P-Gateway (PGW), General GPRS Support Node (GGSN), Content Distribution Network (CDN) etc., of Operator A (User's Home Operator). Increasing window sizes to compensate for the path RTT will have the limitations outlined earlier for TCP. The outbound roamer scenario has a similar TCP performance impact.
 3. Issues in deploying CDNs in Radio Access Networks (RAN) include decreasing client-server control loop that requires deploying CDNs/Cloud functions that terminate encryption closer to the edge. In Cellular RAN, the user IP traffic is encapsulated into General Packet Radio Service (GPRS) Tunneling Protocol-User Plane (GTP-U in UMTS and LTE) tunnels to handle user mobility; the tunnels terminate in APN/GGSN/PGW that are in central locations. One user's traffic may flow through one or more APN's (for example Internet APN, Roaming APN for Operator X, Video-Service APN, OnDeckAPN etc.). The scope of operator private IP addresses may be limited to specific APNs. Since CDNs generally operate on user IP flows, deploying them would require enhancing them with tunnel translation, tunnel management functions etc.
 4. While CDNs that de-encrypt flows or split-connection proxy (similar to split-tcp) could be deployed closer to the edges to reduce control loop RTT, with transport header encryption, such CDNs perform optimization functions only for partner client flows. Therefore, content from some Small-Medium Businesses (SMBs) would not get such CDN benefits.
8. Response to Increased Encryption and Looking Forward

As stated in [RFC7258], "an appropriate balance (between network management and PM mitigations) will emerge over time as real instances of this tension are considered." Numerous operators made it clear in their response to this document that they fully support

strong encryption and providing privacy for end users, this is a common goal. Operators recognize not all the practices documented need to be supported going forward, either because of the risk to end user privacy or alternate technologies and tools have already emerged. This document is intended to support network engineers and other innovators to work toward solving network and security management problems with protocol designers and application developers in new ways that facilitate adoption of strong encryption rather than preventing the use of encryption. By having the discussions on network and security management practices with application developers and protocol designers, each side of the debate can understand each others goals, work toward alternate solutions, and disband with practices that should no longer be supported. A goal of this document is to assist the IETF to understand some of the current practices so as to identify new work items for IETF-related use cases which can help facilitate the adoption of strong session encryption and support network and security management.

9. Security Considerations

There are no additional security considerations as this is a summary and does not include a new protocol or functionality.

10. IANA Considerations

This memo makes no requests of IANA.

11. Acknowledgements

Thanks to our reviewers, Natasha Rooney, Kevin Smith, Ashutosh Dutta, Brandon Williams, Jean-Michel Combes, Nalini Elkins, Paul Barrett, Badri Subramanyan, Igor Lubashev, Suresh Krishnan, Dave Dolson, Mohamed Boucadair, Stephen Farrell, Warren Kumari, Alia Atlas, Roman Danyliw, Mirja Kuhlewind, Ines Robles, Joe Clarke, Kyle Rose, Christian Huitema, and Chris Morrow for their editorial and content suggestions. Surya K. Kovvali provided material for section 7. Chris Morrow and Nik Teague provided reviews and updates specific to the DoS fingerprinting text. Brian Trammell provided the IPFIX text.

12. Informative References

[ACCORD] "Acord BoF IETF95
<https://www.ietf.org/proceedings/95/accord.html>".

- [CAIDA] "CAIDA *Anonymized Internet Traces*
[<http://www.caida.org/data/overview/> and
[http://www.caida.org/data/passive/
passive_2016_dataset.xml](http://www.caida.org/data/passive/passive_2016_dataset.xml)]".
- [DarkMail] "The Dark Mail Technical Alliance <https://darkmail.info/>".
- [DOTS] <https://datatracker.ietf.org/wg/dots/charter/>, "DDoS Open Threat Signaling IETF Working Group".
- [EFF2014] "EFF Report on STARTTLS Downgrade Attacks
[https://www.eff.org/deeplinks/2014/11/
starttls-downgrade-attacks](https://www.eff.org/deeplinks/2014/11/starttls-downgrade-attacks)".
- [Enrich] Narseo Vallina-Rodriguez, et al., "Header Enrichment or ISP Enrichment, Emerging Privacy Threats in Mobile Networks, Hot Middlebox, August 17-21 2015, London, United Kingdom", 2015.
- [I-D.bishop-httpbis-http2-additional-certs]
Bishop, M., Sullivan, N., and M. Thomson, "Secondary Certificate Authentication in HTTP/2", draft-bishop-httpbis-http2-additional-certs-05 (work in progress), October 2017.
- [I-D.dolson-plus-middlebox-benefits]
Dolson, D., Snellman, J., Boucadair, M., and C. Jacquenet, "Beneficial Functions of Middleboxes", draft-dolson-plus-middlebox-benefits-03 (work in progress), March 2017.
- [I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-09 (work in progress), November 2017.
- [I-D.ietf-httpbis-origin-frame]
Nottingham, M. and E. Nygren, "The ORIGIN HTTP/2 Frame", draft-ietf-httpbis-origin-frame-06 (work in progress), January 2018.
- [I-D.ietf-tls-sni-encryption]
Huitema, C. and E. Rescorla, "SNI Encryption in TLS Through Tunneling", draft-ietf-tls-sni-encryption-02 (work in progress), March 2018.

- [I-D.mglt-nvo3-geneve-security-requirements]
Migault, D., Boutros, S., Wing, D., and S. Krishnan,
"Geneve Protocol Security Requirements", draft-mglt-nvo3-
geneve-security-requirements-03 (work in progress),
February 2018.
- [IPFIX-IANA]
"IP Flow Information Export (IPFIX) Entities
<https://www.iana.org/assignments/ipfix/>".
- [JNSLP] Surveillance, Vol. 8 No. 3, "10 Standards for Oversight
and Transparency of National Intelligence Services
<http://jnslp.com/>".
- [M3AAWG] "Messaging, Malware, Mobile Anti-Abuse Working Group
(M3AAWG) <https://www.maawg.org/>".
- [Nygren] <https://blogs.akamai.com/2017/03/reaching-toward-universal-tls-sni.html>, "Erik Nygren, personal reference".
- [QUIC] <https://datatracker.ietf.org/wg/quic/charter/>, "QUIC
(quic)".
- [RFC1945] Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext
Transfer Protocol -- HTTP/1.0", RFC 1945,
DOI 10.17487/RFC1945, May 1996,
<<https://www.rfc-editor.org/info/rfc1945>>.
- [RFC1958] Carpenter, B., Ed., "Architectural Principles of the
Internet", RFC 1958, DOI 10.17487/RFC1958, June 1996,
<<https://www.rfc-editor.org/info/rfc1958>>.
- [RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic
Technology and the Internet", BCP 200, RFC 1984,
DOI 10.17487/RFC1984, August 1996,
<<https://www.rfc-editor.org/info/rfc1984>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol",
RFC 2131, DOI 10.17487/RFC2131, March 1997,
<<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,
"Definition of the Differentiated Services Field (DS
Field) in the IPv4 and IPv6 Headers", RFC 2474,
DOI 10.17487/RFC2474, December 1998,
<<https://www.rfc-editor.org/info/rfc2474>>.

- [RFC2504] Guttman, E., Leong, L., and G. Malkin, "Users' Security Handbook", FYI 34, RFC 2504, DOI 10.17487/RFC2504, February 1999, <<https://www.rfc-editor.org/info/rfc2504>>.
- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <<https://www.rfc-editor.org/info/rfc2663>>.
- [RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <<https://www.rfc-editor.org/info/rfc2775>>.
- [RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, DOI 10.17487/RFC2804, May 2000, <<https://www.rfc-editor.org/info/rfc2804>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <<https://www.rfc-editor.org/info/rfc3135>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3724] Kempf, J., Ed., Austein, R., Ed., and IAB, "The Rise of the Middle and the Future of End-to-End: Reflections on the Evolution of the Internet Architecture", RFC 3724, DOI 10.17487/RFC3724, March 2004, <<https://www.rfc-editor.org/info/rfc3724>>.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <<https://www.rfc-editor.org/info/rfc3954>>.

- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SECure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<https://www.rfc-editor.org/info/rfc4787>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, DOI 10.17487/RFC5655, October 2009, <<https://www.rfc-editor.org/info/rfc5655>>.
- [RFC5965] Shafranovich, Y., Levine, J., and M. Kucherawy, "An Extensible Format for Email Feedback Reports", RFC 5965, DOI 10.17487/RFC5965, August 2010, <<https://www.rfc-editor.org/info/rfc5965>>.
- [RFC6108] Chung, C., Kasyanov, A., Livingood, J., Mody, N., and B. Van Lieu, "Comcast's Web Notification System Design", RFC 6108, DOI 10.17487/RFC6108, February 2011, <<https://www.rfc-editor.org/info/rfc6108>>.
- [RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, DOI 10.17487/RFC6235, May 2011, <<https://www.rfc-editor.org/info/rfc6235>>.
- [RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <<https://www.rfc-editor.org/info/rfc6269>>.
- [RFC6430] Li, K. and B. Leiba, "Email Feedback Report Type Value: not-spam", RFC 6430, DOI 10.17487/RFC6430, November 2011, <<https://www.rfc-editor.org/info/rfc6430>>.
- [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <<https://www.rfc-editor.org/info/rfc6455>>.

- [RFC6590] Falk, J., Ed. and M. Kucherawy, Ed., "Redaction of Potentially Sensitive Data from Mail Abuse Reports", RFC 6590, DOI 10.17487/RFC6590, April 2012, <<https://www.rfc-editor.org/info/rfc6590>>.
- [RFC6591] Fontana, H., "Authentication Failure Reporting Using the Abuse Reporting Format", RFC 6591, DOI 10.17487/RFC6591, April 2012, <<https://www.rfc-editor.org/info/rfc6591>>.
- [RFC6650] Falk, J. and M. Kucherawy, Ed., "Creation and Use of Email Feedback Reports: An Applicability Statement for the Abuse Reporting Format (ARF)", RFC 6650, DOI 10.17487/RFC6650, June 2012, <<https://www.rfc-editor.org/info/rfc6650>>.
- [RFC6651] Kucherawy, M., "Extensions to DomainKeys Identified Mail (DKIM) for Failure Reporting", RFC 6651, DOI 10.17487/RFC6651, June 2012, <<https://www.rfc-editor.org/info/rfc6651>>.
- [RFC6652] Kitterman, S., "Sender Policy Framework (SPF) Authentication Failure Reporting Using the Abuse Reporting Format", RFC 6652, DOI 10.17487/RFC6652, June 2012, <<https://www.rfc-editor.org/info/rfc6652>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <<https://www.rfc-editor.org/info/rfc7012>>.
- [RFC7143] Chadalapaka, M., Satran, J., Meth, K., and D. Black, "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)", RFC 7143, DOI 10.17487/RFC7143, April 2014, <<https://www.rfc-editor.org/info/rfc7143>>.
- [RFC7146] Black, D. and P. Koning, "Securing Block Storage Protocols over IP: RFC 3723 Requirements Update for IPsec v3", RFC 7146, DOI 10.17487/RFC7146, April 2014, <<https://www.rfc-editor.org/info/rfc7146>>.

- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <<https://www.rfc-editor.org/info/rfc7457>>.
- [RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <<https://www.rfc-editor.org/info/rfc7489>>.
- [RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <<https://www.rfc-editor.org/info/rfc7498>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.

- [RFC7540] Belshé, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <<https://www.rfc-editor.org/info/rfc7619>>.
- [RFC7624] Barnes, R., Schneier, B., Jennings, C., Hardie, T., Trammell, B., Huitema, C., and D. Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement", RFC 7624, DOI 10.17487/RFC7624, August 2015, <<https://www.rfc-editor.org/info/rfc7624>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC7754] Barnes, R., Cooper, A., Kolkman, O., Thaler, D., and E. Nordmark, "Technical Considerations for Internet Service Blocking and Filtering", RFC 7754, DOI 10.17487/RFC7754, March 2016, <<https://www.rfc-editor.org/info/rfc7754>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <<https://www.rfc-editor.org/info/rfc7826>>.
- [RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <<https://www.rfc-editor.org/info/rfc7838>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.

- [RFC8073] Moriarty, K. and M. Ford, "Coordinating Attack Response at Internet Scale (CARIS) Workshop Report", RFC 8073, DOI 10.17487/RFC8073, March 2017, <<https://www.rfc-editor.org/info/rfc8073>>.
- [RFC8164] Nottingham, M. and M. Thomson, "Opportunistic Security for HTTP/2", RFC 8164, DOI 10.17487/RFC8164, May 2017, <<https://www.rfc-editor.org/info/rfc8164>>.
- [RFC8165] Hardie, T., "Design Considerations for Metadata Insertion", RFC 8165, DOI 10.17487/RFC8165, May 2017, <<https://www.rfc-editor.org/info/rfc8165>>.
- [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>>.
- [RFC8274] Kampanakis, P. and M. Suzuki, "Incident Object Description Exchange Format Usage Guidance", RFC 8274, DOI 10.17487/RFC8274, November 2017, <<https://www.rfc-editor.org/info/rfc8274>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.
- [SACM] <https://datatracker.ietf.org/wg/sacm/charter/>, "Security Automation and Continuous Monitoring (sacm) IETF Working Group".
- [Snowden] <http://www.jjsylvia.com/bigdatacourse/wp-content/uploads/2016/04/pl4-verble-1.pdf>, "The NSA and Edward Snowden: Surveillance In The 21st Century", 2014.
- [TCPcrypt] <https://datatracker.ietf.org/wg/tcpinc/charter/>, "TCPcrypt".
- [TLS100Proceedings] IETF 100, TLS Working Group Session, "Presentation before the TLS WG at IETF" <https://datatracker.ietf.org/meeting/100/materials/slides-100-tls-sessa-tls13/>, 2017.
- [TS3GPP] "3GPP TS 24.301, "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3"", 2017.

[UPCON] 3GPP, "User Plane Congestion Management
[http://www.3gpp.org/DynaReport/
FeatureOrStudyItemFile-570029.htm](http://www.3gpp.org/DynaReport/FeatureOrStudyItemFile-570029.htm)", 2014.

[UserData] Network and Distributed Systems Symposium, The Internet
Society, "The Security Impact of HTTPS Interception",
2017.

Authors' Addresses

Kathleen Moriarty (editor)
Dell EMC
176 South St
Hopkinton, MA
USA

Phone: +1
Email: Kathleen.Moriarty@dell.com

Al Morton (editor)
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 23, 2018

S. Thomas
R. Reginelli
A. Hope-Bailie
Ripple
January 19, 2018

Crypto-Conditions
draft-thomas-crypto-conditions-04

Abstract

The crypto-conditions specification defines a set of encoding formats and data structures for **conditions** and **fulfillments**. A condition uniquely identifies a logical "boolean circuit" constructed from one or more logic gates, evaluated by either validating a cryptographic signature or verifying the preimage of a hash digest. A fulfillment is a data structure encoding one or more cryptographic signatures and hash digest preimages that define the structure of the circuit and provide inputs to the logic gates allowing for the result of the circuit to be evaluated.

A fulfillment is validated by evaluating that the circuit output is TRUE but also that the provided fulfillment matches the circuit fingerprint, the condition.

Since evaluation of some of the logic gates in the circuit (those that are signatures) also take a message as input the evaluation of the entire fulfillment takes an optional input message which is passed to each logic gate as required. As such the algorithm to validate a fulfillment against a condition and a message matches that of other signature schemes and a crypto-condition can serve as a sophisticated and flexible replacement for a simple signature where the condition is used as the public key and the fulfillment as the signature.

Feedback

This specification is a part of the Interledger Protocol [1] work. Feedback related to this specification should be sent to ledger@ietf.org [2].

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 July 23, 2018.

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 | 4 |
| 2. Terminology | 4 |
| 3. Types | 5 |
| 3.1. Simple and Compound Types | 5 |
| 3.2. Defining and Supporting New types | 6 |
| 4. Features | 6 |
| 4.1. Multi-Algorithm | 6 |
| 4.2. Multi-Signature | 6 |
| 4.3. Multi-Level | 7 |
| 4.4. Crypto-conditions as a signature scheme | 7 |
| 4.5. Crypto-conditions as a trigger in distributed systems . . | 8 |
| 4.6. Smart signatures | 9 |
| 5. Validation of a fulfillment | 9 |
| 5.1. Subfulfillments | 10 |
| 6. Deriving the Condition | 10 |
| 6.1. Conditions as Public Keys | 11 |
| 7. Format | 11 |
| 7.1. Encoding Rules | 11 |
| 7.2. Condition | 11 |

| | | |
|--------|---|----|
| 7.2.1. | Fingerprint | 12 |
| 7.2.2. | Cost | 13 |
| 7.2.3. | Subtypes | 13 |
| 7.3. | Fulfillment | 14 |
| 8. | Crypto-Condition Types | 15 |
| 8.1. | PREIMAGE-SHA-256 | 15 |
| 8.1.1. | Cost | 16 |
| 8.1.2. | ASN.1 | 16 |
| 8.1.3. | Condition Format | 16 |
| 8.1.4. | Fulfillment Format | 16 |
| 8.1.5. | Validating | 16 |
| 8.1.6. | Example | 16 |
| 8.2. | PREFIX-SHA-256 | 17 |
| 8.2.1. | Cost | 17 |
| 8.2.2. | ASN.1 | 18 |
| 8.2.3. | Condition Format | 18 |
| 8.2.4. | Fulfillment Format | 18 |
| 8.2.5. | Validating | 19 |
| 8.2.6. | Example | 19 |
| 8.3. | THRESHOLD-SHA-256 | 20 |
| 8.3.1. | Cost | 20 |
| 8.3.2. | ASN.1 | 20 |
| 8.3.3. | Condition Format | 20 |
| 8.3.4. | Fulfillment Format | 21 |
| 8.3.5. | Validating | 21 |
| 8.3.6. | Example | 21 |
| 8.4. | RSA-SHA-256 | 23 |
| 8.4.1. | RSA Keys | 23 |
| 8.4.2. | Cost | 24 |
| 8.4.3. | ASN.1 | 24 |
| 8.4.4. | Condition Format | 24 |
| 8.4.5. | Fulfillment Format | 24 |
| 8.4.6. | Validating | 25 |
| 8.4.7. | Example | 25 |
| 8.5. | ED25519-SHA256 | 26 |
| 8.5.1. | Cost | 27 |
| 8.5.2. | ASN.1 | 27 |
| 8.5.3. | Condition Format | 27 |
| 8.5.4. | Fulfillment | 27 |
| 8.5.5. | Validating | 27 |
| 8.5.6. | Example | 28 |
| 9. | URI Encoding Rules | 28 |
| 9.1. | Condition URI Format | 28 |
| 9.2. | New URI Parameter Definitions | 29 |
| 9.2.1. | Parameter: Fingerprint Type (fpt) | 29 |
| 9.2.2. | Parameter: Cost (cost) | 29 |
| 9.2.3. | Parameter: Subtypes (subtypes) | 29 |
| 9.3. | Condition URI Parameter Ordering | 30 |

| | |
|---|----|
| 10. Example Condition | 30 |
| 11. References | 31 |
| 11.1. Normative References | 31 |
| 11.2. Informative References | 32 |
| 11.3. URIs | 33 |
| Appendix A. Security Considerations | 34 |
| Appendix B. Test Values | 34 |
| Appendix C. Implementations | 34 |
| Appendix D. ASN.1 Module | 35 |
| Appendix E. IANA Considerations | 37 |
| E.1. Crypto-Condition Type Registry | 37 |
| Authors' Addresses | 37 |

1. Introduction

Crypto-conditions is a scheme for composing signature-like structures from one or more existing signature schemes or hash digest primitives. It defines a mechanism for these existing primitives to be combined and grouped to create complex signature arrangements but still maintain the useful properties of a simple signature, most notably, that a deterministic algorithm exists to verify the signature against a message given a public key.

Using crypto-conditions, existing primitives such as RSA and ED25519 signature schemes and SHA256 digest algorithms can be used as logic gates to construct complex boolean circuits which can then be used as compound signatures. The validation function for these compound signatures takes as input the fingerprint of the circuit, called the condition, the circuit definition and minimum required logic gates with their inputs, called the fulfillment, and a message.

The function returns a boolean indicating if the compound signature is valid or not. This property of crypto-conditions means they can be used in most scenarios as a replacement for existing signature schemes which also take as input, a public key (the condition), a signature (the fulfillment), and a message and return a boolean result.

2. Terminology

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

3. Types

Crypto-conditions are a standard format for expressing conditions and fulfillments. The format supports multiple algorithms, including different hash functions and cryptographic signing schemes. Crypto-conditions can be nested in multiple levels, with each level possibly having multiple signatures.

The different types of crypto-conditions each have different internal structures and employ different cryptographic algorithms as primitives.

3.1. Simple and Compound Types

Two categories of crypto-condition type exist. Simple crypto-conditions provide a standard encoding of common cryptographic primitives with hardcoded parameters, e.g RSA and ED25519 signature or SHA256 hash digests. As such, simple types that use the same underlying scheme (e.g. SHA) with different parameters (e.g. 256 or 512 bits) are considered different crypto-condition types.

As an example, the types defined in this version of the specification all use the SHA-256 digest algorithm to generate the condition fingerprint. If a future version were to introduce SHA-512 as an alternative this would require that new types be defined for each existing type that must have its condition generated using SHA-512.

Compound crypto-conditions contain one or more sub-crypto-conditions. The compound crypto-condition will evaluate to TRUE or FALSE based on the output of the evaluation of the sub-crypto-conditions. In this way compound crypto-conditions are used to construct branches of a boolean circuit.

To validate a compound crypto-condition all sub-crypto-conditions are provided in the fulfillment so that the fingerprint of the compound condition can be generated. However, some of these sub-crypto-conditions may be sub-fulfillments and some may be sub-conditions, depending on the type and properties of the compound crypto-condition.

As an example, in the case of an m-of-n signature scheme, only m sub-fulfillments are needed to validate the compound signature, but the remaining n-m sub-conditions must still be provided to validate that the complete fulfillment matches the originally provided condition. This is an important feature for multi-party signing, when not all parties are ready to provide fulfillment yet all parties still desire fulfillment of the overall condition if enough counter-parties do provide fulfillment.

3.2. Defining and Supporting New types

The crypto-conditions format has been designed so that it can be expanded. For example, you can add new cryptographic signature schemes or hash functions. This is important because advances in cryptography frequently render old algorithms insecure or invent newer, more effective algorithms.

Implementations are not required to support all condition types therefore it is necessary to indicate which types an implementation must support in order to validate a fulfillment. For this reason, compound conditions are encoded with an additional field, subtypes, indicating the set of types and subtypes of all sub-crypto-conditions.

4. Features

Crypto-conditions offer many of the features required of a regular signature scheme but also others which make them useful in a variety of new use cases.

4.1. Multi-Algorithm

Each condition type uses one or more cryptographic primitives such as digest or signature algorithms. Compound types may contain sub-crypto-conditions of any type and indicate the set of underlying types in the subtypes field of the condition

To verify that a given implementation can verify a fulfillment for a given condition, implementations **MUST** ensure they are able to validate fulfillments of all types indicated in the subtypes field of a compound condition. If an implementation encounters an unknown type it **MUST** reject the condition as it will almost certainly be unable to validate the fulfillment.

4.2. Multi-Signature

Crypto-conditions can abstract away many of the details of multi-sign. When a party provides a condition, other parties can treat it opaquely and do not need to know about its internal structure. That allows parties to define arbitrary multi-signature setups without breaking compatibility. That said, it is important that implementations must inspect the types and subtypes of any crypto-conditions they encounter to ensure they do not pass on a condition they will not be able to verify at a later stage.

In many instances protocol designers can use crypto-conditions as a drop-in replacement for public key signature algorithms and add

multi-signature support to their protocols without adding any additional complexity.

4.3. Multi-Level

Crypto-conditions elegantly support weighted multi-signatures and multi-level signatures. A threshold condition has a number of subconditions, and a target threshold. Each subcondition can be a signature or another threshold condition. This provides flexibility in forming complex conditions.

For example, consider a threshold condition that consists of two subconditions, one each from Wayne and Alf. Alf's condition can be a signature condition while Wayne's condition is a threshold condition, requiring both Claude and Dan to sign for him.

Multi-level signatures allow more complex relationships than simple M-of-N signing. For example, a weighted condition can support an arrangement of subconditions such as, "Either Ron, Mac, and Ped must approve; or Smithers must approve."

4.4. Crypto-conditions as a signature scheme

Crypto-conditions is a signature scheme for compound signatures which has similar properties to most other signature schemes, such as:

1. Validation of the signature (the fulfillment) is done using a public key (the condition) and a message as input
2. The same public key can be used to validate multiple different signatures, each against a different message
3. It is not possible to derive the signature from the public key

However, the scheme also has a number of features that make it unique such as:

1. It is possible to derive the same public key from any valid signature without the message
2. It is possible for the same public key and message to be used to validate multiple signatures. For example, the fulfillment of an m-of-n condition will be different for each combination of n signatures.
3. Composite signatures use one or more other signatures as components allowing for recursive signature validation logic to be defined.

4. A valid signature can be produced using different combinations of private keys if the structure of the compound signature requires only specific combinations of internal signatures to be valid (m of n signature scheme).

4.5. Crypto-conditions as a trigger in distributed systems

One of the challenges facing a distributed system is achieving atomic execution of a transaction across the system. A common pattern for solving this problem is two-phase commit in which the most time and resource-consuming aspects of the transaction are prepared by all participants following which a simple trigger is sufficient to either commit or abort the transaction. Described in more abstract terms, the system consists of a number of participants that have prepared a transaction pending the fulfillment of a predefined condition.

Crypto-conditions defines a mechanism for expressing these triggers as pairs of unique trigger identifiers (conditions) and cryptographically verifiable triggers (fulfillments) that can be deterministically verified by all participants.

It is also important that all participants in such a distributed system are able to evaluate, prior to the trigger being fired, that they will be capable of verifying the trigger. Determinism is useless if validation of the trigger requires algorithms or resources that are not available to all participants.

Therefore conditions may be used as **distributable event descriptions** in the form of a `_fingerprint_`, but also `_event meta-data_` that allows the event verification system to determine if they have the necessary capabilities (such as required crypto-algorithms) and resources (such as heap size or memory) to verify the event notification later.

Fulfillments are therefore **cryptographically verifiable event notifications** that can be used to verify the event occurred but also that it matches the given description.

When using crypto-conditions as a trigger it will often make sense for the message that is used for validation to be empty to match the signature of the trigger processing system's API. This makes crypto-conditions compatible with systems that use simple hash-locks as triggers.

If a PKI signature scheme is being used for the triggers this would require a new key pair for each trigger which is impractical. Therefore the PREFIX compound type wraps a sub-crypto-condition with a message prefix that is applied to the message before signature

validation. In this way a unique condition can be derived for each trigger even if the same key pair is re-used with an empty message.

4.6. Smart signatures

In the Interledger protocol, fulfillments provide non-repudiable proof that a transaction has been completed on a ledger. They are simple messages that can be easily shared with other ledgers. This allows ledgers to escrow funds or hold a transfer conditionally, then execute the transfer automatically when the ledger sees the fulfillment of the stated condition. In this way the Interledger protocol synchronizes multiple transfers on distinct ledgers in an almost atomic end-to-end transaction.

Crypto-conditions may also be useful in other contexts where a system needs to make a decision based on predefined criteria, and the proof from a trusted oracle(s) that the criteria have been met, such as smart contracts.

The advantage of using crypto-conditions for such use cases as opposed to a turing complete contract scripting language is the fact that the outcome of a crypto-condition validation is deterministic across platforms as long as the underlying cryptographic primitives are correctly implemented.

5. Validation of a fulfillment

Validation of a fulfillment (F) against a condition (C) and a message (M), in the majority of cases, follows these steps:

1. The implementation must derive a condition from the fulfillment and ensure that the derived condition (D) matches the given condition (C).
2. If the fulfillment is a simple crypto-condition AND is based upon a signature scheme (such as RSA-PSS or ED25519) then any signatures in the fulfillment (F) must be verified, using the appropriate signature verification algorithm, against the corresponding public key, also provided in the fulfillment and the message (M) (which may be empty).
3. If the fulfillment is a compound crypto-condition then the sub-fulfillments MUST each be validated. In the case of the PREFIX-SHA-256 type the sub-fulfillment MUST be valid for F to be valid and in the case of the THRESHOLD-SHA-256 type the number of valid sub-fulfillments must be equal or greater than the threshold defined in F.

If the derived condition (D) matches the input condition (C) AND the boolean circuit defined by the fulfillment evaluates to TRUE then the fulfillment (F) fulfills the condition (C).

A more detailed validation algorithm for each crypto-condition type is provided with the details of the type later in this document. In each case the notation F.x or C.y implies; the decoded value of the field named x of the fulfillment and the decoded value of the field named y of the Condition respectively.

5.1. Subfulfillments

In validating a fulfillment for a compound crypto-condition it is necessary to validate one or more sub-fulfillments per step 3 above. In this instance the condition for one or more of these sub-fulfillments is often not available for comparison with the derived condition. Implementations MUST skip the first fulfillment validation step as defined above and only perform steps 2 and 3 of the validation.

The message (M) used to validate sub-fulfillments is the same message (M) used to validate F however in the case of the PREFIX-SHA-256 type this is prefixed with F.prefix before validation of the sub-fulfillment is performed.

6. Deriving the Condition

Since conditions provide a unique fingerprint for fulfillments it is important that a deterministic algorithm is used to derive a condition. For each crypto-condition type details are provided on how to:

1. Assemble the fingerprint content and calculate the hash digest of this data.
2. Calculate the maximum cost of validating a fulfillment

For compound types the fingerprint content will contain the complete, encoded, condition for all sub-crypto-conditions. Implementations MUST abide by the ordering rules provided when assembling the fingerprint content.

When calculating the fingerprint of a compound crypto-condition implementations MUST first derive the condition for all sub-fulfillments and include these conditions when assembling the fingerprint content.

6.1. Conditions as Public Keys

Since the condition is just a fingerprint and meta-data about the crypto-condition it can be transmitted freely in the same way a public key is shared publicly. It's not possible to derive the fulfillment from the condition.

7. Format

A description of crypto-conditions is provided in this document using Abstract Syntax Notation One (ASN.1) as defined in [itu.X680.2015].

7.1. Encoding Rules

Implementations of this specification MUST support encoding and decoding using Distinguished Encoding Rules (DER) as defined in [itu.X690.2015]. This is the canonical encoding format.

Alternative encodings may be used to represent top-level conditions and fulfillments but to ensure a deterministic outcome in producing the condition fingerprint content, including any sub-conditions, MUST be DER encoded prior to hashing.

The exception is the PREIMAGE-SHA-256 condition where the fingerprint content is the raw preimage which is not encoded prior to hashing. This is to allow a PREIMAGE-SHA-256 crypto-condition to be used in systems where "hash-locks" are already in use.

7.2. Condition

The binary encoding of conditions differs based on their type. All types define at least a fingerprint and cost sub-field. Some types, such as the compound condition types, define additional sub-fields that are required to convey essential properties of the crypto-condition (such as the sub-types used by sub-conditions in the case of the compound types).

Each crypto-condition type has a type ID. The list of known types is the IANA-maintained Crypto-Condition Type Registry (Appendix E.1).

Conditions are encoded as follows:

```
Condition ::= CHOICE {
    preimageSha256    [0] SimpleSha256Condition,
    prefixSha256      [1] CompoundSha256Condition,
    thresholdSha256   [2] CompoundSha256Condition,
    rsaSha256         [3] SimpleSha256Condition,
    ed25519Sha256     [4] SimpleSha256Condition
}

SimpleSha256Condition ::= SEQUENCE {
    fingerprint      OCTET STRING (SIZE(32)),
    cost              INTEGER (0..4294967295)
}

CompoundSha256Condition ::= SEQUENCE {
    fingerprint      OCTET STRING (SIZE(32)),
    cost              INTEGER (0..4294967295),
    subtypes          ConditionTypes
}

ConditionTypes ::= BIT STRING {
    preImageSha256    (0),
    prefixSha256      (1),
    thresholdSha256   (2),
    rsaSha256         (3),
    ed25519Sha256     (4)
}
```

7.2.1. Fingerprint

The fingerprint is an octet string uniquely representing the condition with respect to other conditions **of the same type**.

Implementations which index conditions **MUST** use the complete encoded condition as the key, not just the fingerprint - as different conditions of different types may have the same fingerprint.

For most condition types, the fingerprint is a cryptographically secure hash of the data which defines the condition, such as a public key.

For types that use PKI signature schemes, the signature is intentionally not included in the content that is used to compose the fingerprint. This means the fingerprint can be calculated without needing to know the message or having access to the private key.

Future types may use different functions to produce the fingerprint, which may have different lengths, therefore the field is encoded as a variable length string.

7.2.2. Cost

For each type, a cost function is defined which produces a deterministic cost value based on the properties of the condition.

The cost functions are designed to produce a number that will increase rapidly if the structure and properties of a crypto-condition are such that they increase the resource requirements of a system that must validate the fulfillment.

The constants used in the cost functions are selected in order to provide some consistency across types for the cost value and the expected "real cost" of validation. This is not an exact science given that some validations will require signature verification (such as RSA and ED25519) and others will simply require hashing and storage of large values therefore the cost functions are roughly configured (through selection of constants) to be the number of bytes that would need to be processed by the SHA-256 hash digest algorithm to produce the equivalent amount of work.

The goal is to produce an indicative number that implementations can use to protect themselves from attacks involving crypto-conditions that would require massive resources to validate (denial of service type attacks).

Since dynamic heuristic measures can't be used to achieve this a deterministic value is required that can be produced consistently by any implementation, therefore for each crypto-condition type, an algorithm is provided for consistently calculating the cost.

Implementations MUST determine a safe cost ceiling based on the expected cost value of crypto-conditions they will need to process. When a crypto-condition is submitted to an implementation, the implementation MUST verify that it will be able to process a fulfillment with the given cost (i.e. the cost is lower than the allowed ceiling) and reject it if not.

Cost function constants have been rounded to numbers that have an efficient base-2 representation to facilitate efficient arithmetic operations.

7.2.3. Subtypes

Subtypes is a bitmap that indicates the set of types an implementation must support in order to be able to successfully validate the fulfillment of this condition. This is the set of types and subtypes of all sub-crypto-conditions, recursively excluding the type of the root crypto-condition.

It must be possible to verify that all types used in a crypto-condition are supported (including the types and subtypes of any sub-crypto-conditions) even if the fulfillment is not available to be analysed yet. Therefore, all compound conditions set the bits in this bitmap that correspond to the set of types and subtypes of all sub-crypto-conditions.

The field is encoded as a variable length BIT STRING, as defined in ASN.1, to accommodate new types that may be defined.

Each bit in the bitmap represents a type from the list of known types in the IANA-maintained Crypto-Condition Type Registry (Appendix E.1) and the bit corresponding to each type is the bit at position X where X is the type ID of the type.

The presence of one or more sub-crypto-conditions of a specific type is indicated by setting the numbered bit corresponding to the type ID of that type.

In DER encoding, the bits in a bitstring are numbered from the MOST significant bit (bit 0) to least significant (bit 7) of the first byte and then continue with the MOST significant bit (bit 8) of the next byte, and so on. For example, a compound condition that contains an ED25519-SHA-256 crypto-condition as a sub-crypto-condition will set the bit at position 4 and the BITSTRING will be DER encoded with an appropriate tag byte followed by the three bytes 0x02 0x03 and 0x80, where 0x02 indicates the length (2 bytes, the first being the padding indicator), 0x03 indicates that there are 3 padding bits in the last byte and 0x80 indicates the 5 bits in the string are set to 00001.

7.3. Fulfillment

The ASN.1 definition for fulfillments is defined as follows:

```

Fulfillment ::= CHOICE {
    preimageSha256      [0] PreimageFulfillment ,
    prefixSha256        [1] PrefixFulfillment,
    thresholdSha256     [2] ThresholdFulfillment,
    rsaSha256           [3] RsaSha256Fulfillment,
    ed25519Sha256       [4] Ed25519Sha512Fulfillment
}

PreimageFulfillment ::= SEQUENCE {
    preimage            OCTET STRING
}

PrefixFulfillment ::= SEQUENCE {
    prefix              OCTET STRING,
    maxLength           INTEGER (0..4294967295),
    subfulfillment      Fulfillment
}

ThresholdFulfillment ::= SEQUENCE {
    subfulfillments     SET OF Fulfillment,
    subconditions       SET OF Condition
}

RsaSha256Fulfillment ::= SEQUENCE {
    modulus             OCTET STRING,
    signature           OCTET STRING
}

Ed25519Sha512Fulfillment ::= SEQUENCE {
    publicKey           OCTET STRING (SIZE(32)),
    signature           OCTET STRING (SIZE(64))
}

```

8. Crypto-Condition Types

The following condition types are defined in this version of the specification. While support for additional crypto-condition types may be added in the future and will be registered in the IANA maintained Crypto-Condition Type Registry (Appendix E.1), no other types are supported by this specification.

8.1. PREIMAGE-SHA-256

PREIMAGE-SHA-256 is assigned the type ID 0. It relies on the availability of the SHA-256 digest algorithm.

This type of condition is also called a "hashlock". By creating a hash of a difficult-to-guess 256-bit random or pseudo-random integer

it is possible to create a condition which the creator can trivially fulfill by publishing the random value. However, for anyone else, the condition is cryptographically hard to fulfill, because they would have to find a preimage for the given condition hash.

Implementations MUST ignore any input message when validating a PREIMAGE-SHA-256 fulfillment as the validation of this crypto-condition type only requires that the SHA-256 digest of the preimage, taken from the fulfillment, matches the fingerprint, taken from the condition.

8.1.1. Cost

The cost is the size, in bytes, of the **unencoded** preimage.

cost = preimage length

8.1.2. ASN.1

```
-- Condition Fingerprint
-- The PREIMAGE-SHA-256 condition fingerprint content is not DER encoded
-- The fingerprint content is the preimage
```

```
-- Fulfillment
PreimageFulfillment ::= SEQUENCE {
    preimage          OCTET STRING
}
```

8.1.3. Condition Format

The fingerprint of a PREIMAGE-SHA-256 condition is the SHA-256 hash of the **unencoded** preimage.

8.1.4. Fulfillment Format

The fulfillment simply contains the preimage (encoded into a SEQUENCE of one element for consistency).

8.1.5. Validating

A PREIMAGE-SHA-256 fulfillment is valid iff C.fingerprint is equal to the SHA-256 hash digest of F.

8.1.6. Example

```
examplePreimageCondition Condition ::=
  preimageSha256 : {
    fingerprint '7F83B165 7FF1FC53 B92DC181 48A1D65D FC2D4B1F A3D67728 4ADDD200
126D9069'H,
    cost          12
  }

examplePreimageFulfillment Fulfillment ::=
  preimageSha256 : {
    preimage '48656C6C 6F20576F 726C6421'H
  }
```

8.2. PREFIX-SHA-256

PREFIX-SHA-256 is assigned the type ID 1. It relies on the availability of the SHA-256 digest algorithm and any other algorithms required by its sub-crypto-condition as it is a compound crypto-condition type.

Prefix crypto-conditions provide a way to narrow the scope of other crypto-conditions that are used inside the prefix crypto-condition as a sub-crypto-condition.

Because a condition is the fingerprint of a public key, by creating a prefix crypto-condition that wraps another crypto-condition we can narrow the scope from signing an arbitrary message to signing a message with a specific prefix.

We can also use the prefix condition in contexts where there is an empty message used for validation of the fulfillment so that we can reuse the same key pair for multiple crypto-conditions, each with a different prefix, and therefore generate a unique condition and fulfillment each time.

Implementations MUST prepend the prefix to the provided message and will use the resulting value as the message to validate the sub-fulfillment.

8.2.1. Cost

The cost is the size, in bytes, of the **unencoded** prefix, plus the maximum message that will be accepted to be prefixed and validated by the subcondition, plus the cost of the sub-condition, plus the constant 1024.

$$\text{cost} = \text{prefix.length (in bytes)} + \text{max_message_length} + \text{subcondition_cost} + 1024$$

8.2.2. ASN.1

```
-- Condition Fingerprint
PrefixFingerprintContents ::= SEQUENCE {
    prefix          OCTET STRING,
    maxMessageLength INTEGER (0..4294967295),
    subcondition    Condition
}

-- Fulfillment
PrefixFulfillment ::= SEQUENCE {
    prefix          OCTET STRING,
    maxMessageLength INTEGER (0..4294967295),
    subfulfillment  Fulfillment
}
```

8.2.3. Condition Format

The fingerprint of a PREFIX-SHA-256 condition is the SHA-256 digest of the DER encoded fingerprint contents which are a SEQUENCE of:

prefix An arbitrary octet string which will be prepended to the message during validation of the sub-fulfillment.

maxMessageLength The maximum size, in bytes, of the message that will be accepted during validation of the fulfillment of this condition.

subcondition The condition derived from the sub-fulfillment of this crypto-condition.

8.2.4. Fulfillment Format

The fulfillment of a PREFIX-SHA-256 crypto-condition is a PrefixFulfillment which is a SEQUENCE of:

prefix An arbitrary octet string which will be prepended to the message during validation of the sub-fulfillment.

maxMessageLength The maximum size, in bytes, of the message that will be accepted during validation of the fulfillment of this condition.

subfulfillment A fulfillment that will be verified against the prefixed message.

8.2.5. Validating

A PREFIX-SHA-256 fulfillment is valid iff:

1. The size of M, in bytes, is less than or equal to F.maxMessageLength AND
2. F.subfulfillment is valid, where the message used for validation of f is M prefixed by F.prefix AND
3. D is equal to C

8.2.6. Example

examplePrefixCondition Condition ::=

```
prefixSha256 : {
  fingerprint 'BB1AC526 0C0141B7 E54B26EC 2330637C 5597BF81 1951AC09 E744AD20
FF77E287'H,
  cost          1024,
  subtypes      { preimageSha256 }
}
```

examplePrefixFulfillment Fulfillment ::=

```
prefixSha256 : {
  prefix          ''H,
  maxMessageLength 0,
  subfulfillment  preimageSha256 : { preimage ''H }
}
```

examplePrefixFingerprintContents PrefixFingerprintContents ::= {

```
prefix          ''H,
maxMessageLength 0,
subcondition     preimageSha256 : {
  fingerprint    'E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B78
52B855'H,
  cost           0
}
}
```

Note that the example given, while useful to demonstrate the structure, has less practical security value than the use of an RSA-SHA-256 or ED25519-SHA-256 subfulfillment. Since the subfulfillment is a PREIMAGE-SHA-256, the validation of which ignores the incoming message, as long as the prefix, maxMessageLength and preimage provided in the subfulfillment are correct, the parent PREFIX-SHA-256 fulfillment will validate.

In this case, wrapping the PREIMAGE-SHA-256 crypto-condition in the PREFIX-SHA-256 crypto-condition, has the effect of enforcing a message length of 0 bytes.

Note also, any change to the PREFIX-SHA-256 crypto-condition's prefix and maxMessageLength values result in a different fingerprint value, effectively namespacing the underlying preimage and re-hashing it. The result is a new crypto-condition with a new and unique fingerprint with no change to the underlying sub-crypto-condition.

8.3. THRESHOLD-SHA-256

THRESHOLD-SHA-256 is assigned the type ID 2. It relies on the availability of the SHA-256 digest algorithm and any other algorithms required by any of its sub-crypto-conditions as it is a compound crypto-condition type.

8.3.1. Cost

The cost is the sum of the F.threshold largest cost values of all sub-conditions, added to 1024 times the total number of sub-conditions.

$$\text{cost} = (\text{sum of largest F.threshold subcondition.cost values}) + 1024 * \text{F.subconditions.count}$$

For example, if a threshold crypto-condition contains 5 sub-conditions with costs of 64, 64, 82, 84 and 84 and has a threshold of 3, the cost is equal to the sum of the largest three sub-condition costs ($82 + 84 + 84 = 250$) plus 1024 times the number of sub-conditions ($1024 * 5 = 5120$): 5370

8.3.2. ASN.1

```
-- Condition Fingerprint
ThresholdFingerprintContents ::= SEQUENCE {
    threshold          INTEGER (1..65535),
    subconditions      SET OF Condition
}

-- Fulfillment
ThresholdFulfillment ::= SEQUENCE {
    subfulfillments    SET OF Fulfillment,
    subconditions      SET OF Condition
}
```

8.3.3. Condition Format

The fingerprint of a THRESHOLD-SHA-256 condition is the SHA-256 digest of the DER encoded fingerprint contents which are a SEQUENCE of:

threshold A number that MUST be an integer in the range 1 ... 65535. In order to fulfill a threshold condition, the count of the sub-fulfillments MUST be equal to the threshold.

subconditions The set of sub-conditions, F.threshold of which MUST be satisfied by valid sub-fulfillments provided in the fulfillment. The SET of DER encoded sub-conditions is sorted according to the DER encoding rules for a SET, in lexicographic (big-endian) order, smallest first as defined in section 11.6 of [itu.X690.2015].

8.3.4. Fulfillment Format

The fulfillment of a THRESHOLD-SHA-256 crypto-condition is a ThresholdFulfillment which is a SEQUENCE of:

subfulfillments A SET OF fulfillments. The number of elements in this set is equal to the threshold therefore implementations must use the length of this SET as the threshold value when deriving the fingerprint of this crypto-condition.

subconditions A SET OF conditions. This is the list of unfulfilled sub-conditions. This list must be combined with the list of conditions derived from the subfulfillments and the combined list, sorted, and used as the subconditions value when deriving the fingerprint of this crypto-condition.

This may be an empty list.

8.3.5. Validating

A THRESHOLD-SHA-256 fulfillment is valid iff :

1. All F.subfulfillments are valid.
2. D is equal to C.

8.3.6. Example

```
exampleThresholdCondition Condition ::=
  thresholdSha256 : {
    fingerprint 'B4B84136 DF48A71D 73F4985C 04C6767A 778ECB65 BA7023B4 506823BE
EE7631B9'H,
    cost          1024,
    subtypes      { preimageSha256 }
  }
```

```
exampleThresholdFulfillment Fulfillment ::=
  thresholdSha256 : {
```

```

        subfulfillments { preimageSha256 : { preimage ''H } },
        subconditions   { }
    }

exampleThresholdFingerprintContents ThresholdFingerprintContents ::= {
    threshold 1,
    subconditions {
        preimageSha256 : {
            fingerprint 'E3B0C442 98FC1C14 9AFBF4C8 996FB924 27AE41E4 649B934C A495991
B 7852B855'H,
            cost          0
        }
    }
}

exampleThresholdCondition2 Condition ::=
    thresholdSha256 : {
        fingerprint '5A218ECE 7AC4BC77 157F04CB 4BC8DFCD 5C9D225A 55BD0AA7 60BCA2A4
F1773DC6'H,
        cost          2060,
        subtypes      { preimageSha256 }
    }

exampleThresholdFulfillment2 Fulfillment ::=
    thresholdSha256 : {
        subfulfillments { preimageSha256 : { preimage ''H } },
        subconditions {
            preimageSha256 : {
                fingerprint '7F83B165 7FF1FC53 B92DC181 48A1D65D FC2D4B1F A3D67728 4ADDD
200 126D9069'H,
                cost          12
            }
        }
    }

exampleThresholdFingerprintContents2 ThresholdFingerprintContents ::= {
    threshold 1,
    subconditions {
        preimageSha256 : {
            fingerprint 'E3B0C442 98FC1C14 9AFBF4C8 996FB924 27AE41E4 649B934C A495991
B 7852B855'H,
            cost          0
        },
        preimageSha256 : {
            fingerprint '7F83B165 7FF1FC53 B92DC181 48A1D65D FC2D4B1F A3D67728 4ADDD20
0 126D9069'H,
            cost          12
        }
    }
}

```

8.4. RSA-SHA-256

RSA-SHA-256 is assigned the type ID 3. It relies on the SHA-256 digest algorithm and the RSA-PSS signature scheme.

The signature algorithm used is RSASSA-PSS as defined in PKCS#1 v2.2. [RFC8017]

Implementations MUST NOT use the default RSASSA-PSS-params. Implementations MUST use the SHA-256 hash algorithm and therefore, the same algorithm in the mask generation algorithm, as recommended in [RFC8017]. The algorithm parameters to use, as defined in [RFC4055] are:

```
pkcs-1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }
```

```
id-sha256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) 1 }
```

```
sha256Identifier AlgorithmIdentifier ::= {  
    algorithm          id-sha256,  
    parameters         nullParameters  
}
```

```
id-mgf1 OBJECT IDENTIFIER ::= { pkcs-1 8 }
```

```
mgf1SHA256Identifier AlgorithmIdentifier ::= {  
    algorithm          id-mgf1,  
    parameters         sha256Identifier  
}
```

```
rsassa-pss-sha256-params RSASSA-PSS-params ::= {  
    hashAlgorithm      sha256Identifier,  
    maskGenAlgorithm   mgf1SHA256Identifier,  
    saltLength         20,  
    trailerField       1  
}
```

8.4.1. RSA Keys

To optimize the RsaFulfillment, and enforce a public exponent value of 65537, only the RSA Public Key modulus is stored in the RsaFingerprintContents and RsaFulfillment.

The modulus is stored as an OCTET STRING representing an unsigned integer (i.e. no sign byte) in big-endian byte-order, the most significant byte being the first in the string.

Implementations MUST use moduli greater than 128 bytes (1017 bits) and smaller than or equal to 512 bytes (4096 bits.) Large moduli slow down signature verification which can be a denial-of-service vector. DNSSEC also limits the modulus to 4096 bits [RFC3110]. OpenSSL supports up to 16384 bits [OPENSSL-X509-CERT-EXAMPLES].

Implementations MUST use the value 65537 for the public exponent e as recommended in [RFC4871]. Very large exponents can be a DoS vector [LARGE-RSA-EXPONENTS] and 65537 is the largest Fermat prime, which has some nice properties [USING-RSA-EXPONENT-OF-65537].

The recommended modulus size as of 2016 is 2048 bits [KEYLENGTH-RECOMMENDATION]. In the future we anticipate an upgrade to 3072 bits which provides approximately 128 bits of security [NIST-KEYMANAGEMENT] (p. 64), about the same level as SHA-256.

8.4.2. Cost

The cost is the square of the RSA key modulus size (in bytes).

$\text{cost} = (\text{modulus size in bytes})^2$

8.4.3. ASN.1

```
-- Condition Fingerprint
RsaFingerprintContents ::= SEQUENCE {
    modulus          OCTET STRING
}

-- Fulfillment
RsaSha256Fulfillment ::= SEQUENCE {
    modulus          OCTET STRING,
    signature        OCTET STRING
}
```

8.4.4. Condition Format

The fingerprint of an RSA-SHA-256 condition is the SHA-256 digest of the DER encoded fingerprint contents which is a SEQUENCE of a single element, the modulus of the RSA Key Pair.

8.4.5. Fulfillment Format

The fulfillment of an RSA-SHA-256 crypto-condition is an RsaSha256Fulfillment which is a SEQUENCE of:

modulus The modulus of the RSA key pair used to sign and verify the signature provided.

signature An octet string representing the RSA signature on the message M.

Implementations MUST verify that the signature is numerically less than the modulus.

Note that the message that has been signed is provided separately. If no message is provided, the message is assumed to be an octet string of length zero.

8.4.6. Validating

An RSA-SHA-256 fulfillment is valid iff :

1. F.signature is valid for the message M, using the RSA public key with modulus = F.modulus and exponent = 65537 for verification.
2. D is equal to C.

8.4.7. Example

```
exampleRsaCondition Condition ::=
```

```
  rsaSha256 : {
    fingerprint 'B31FA820 6E4EA7E5 15337B3B 33082B87 76518010 85ED84FB 4DAEB247
BF698D7F'H,
    cost          65536
  }
```

```
exampleRsaSha256Fulfillment Fulfillment ::=
```

```
  rsaSha256 : {
    modulus 'E1EF8B24 D6F76B09 C81ED775 2AA262F0 44F04A87 4D43809D 31CEA612 F9
9B0C97 A8B43741
          53E3EEF3 D6661684 3E0E41C2 93264B71 B6173DB1 CF0D6CD5 58C58657 70
6FCF09 7F704C48
          3E59CBFD FD5B3EE7 BC80D740 C5E0F047 F3E85FC0 D7581577 6A6F3F23 C5
DC5E79 7139A688
          2E38336A 4A5FB361 37620FF3 663DBAE3 28472801 862F72F2 F87B202B 9C
89ADD7 CD5B0A07
          6F7C53E3 5039F67E D17EC815 E5B4305C C6319706 8D5E6E57 9BA6DE5F 4E
3E57DF 5E4E072F
          F2CE4C66 EB452339 73875275 9639F025 7BF57DBD 5C443FB5 158CCE0A 3D
36ADC7 BA01F33A
          0BB6DBB2 BF989D60 7112F234 4D993E77 E563C1D3 61DEDF57 DA96EF2C FC
685F00 2B638246
          A5B309B9'H,
    signature '48E8945E FE007556 D5BF4D5F 249E4808 F7307E29 511D3262 DAEF61D8 80
98F9AA 4A8BC062
          3A8C9757 38F65D6B F459D543 F289D73C BC7AF4EA 3A33FBBF3 EC444044 79
11D722 94091E56
          1833628E 49A772ED 608DE6C4 4595A91E 3E17D6CF 5EC3B252 8D63D2AD D6
463989 B12EEC57
          7DF64709 60DF6832 A9D84C36 0D1C217A D64C8625 BDB594FB 0ADA086C DE
CBBDE5 80D424BF
          9746D2F0 C312826D BBB00AD6 8B52C4CB 7D47156B A35E3A98 1C973863 79
2CC80D 04A18021
          0A524158 65B64B3A 61774B1D 3975D78A 98B0821E E55CA0F8 6305D425 29
E10EB0 15CEFD40
          2FB59B2A BB8DEEE5 2A6F2447 D2284603 D219CD4E 8CF9CFFD D5498889 C3
780B59 DD6A57EF
          7D732620'H
  }
```

```
exampleRsaFingerprintContents RsaFingerprintContents ::= {
```

```
  modulus 'E1EF8B24 D6F76B09 C81ED775 2AA262F0 44F04A87 4D43809D 31CEA612 F9
9B0C97 A8B43741
          53E3EEF3 D6661684 3E0E41C2 93264B71 B6173DB1 CF0D6CD5 58C58657 70
6FCF09 7F704C48
          3E59CBFD FD5B3EE7 BC80D740 C5E0F047 F3E85FC0 D7581577 6A6F3F23 C5
DC5E79 7139A688
          2E38336A 4A5FB361 37620FF3 663DBAE3 28472801 862F72F2 F87B202B 9C
89ADD7 CD5B0A07
          6F7C53E3 5039F67E D17EC815 E5B4305C C6319706 8D5E6E57 9BA6DE5F 4E
3E57DF 5E4E072F
          F2CE4C66 EB452339 73875275 9639F025 7BF57DBD 5C443FB5 158CCE0A 3D
36ADC7 BA01F33A
          0BB6DBB2 BF989D60 7112F234 4D993E77 E563C1D3 61DEDF57 DA96EF2C FC
685F00 2B638246
          A5B309B9'H
}
```

8.5. ED25519-SHA256

ED25519-SHA-256 is assigned the type ID 4. It relies on the SHA-256 and SHA-512 digest algorithms and the ED25519 signature scheme.

The exact algorithm and encodings used for the public key and signature are defined in [I-D.irtf-cfrg-eddsa] as Ed25519. SHA-512 is used as the hashing function for this signature scheme.

8.5.1. Cost

The public key and signature are a fixed size therefore the cost for an ED25519 crypto-condition is fixed at 131072.

cost = 131072

8.5.2. ASN.1

```
-- Condition Fingerprint
Ed25519FingerprintContents ::= SEQUENCE {
    publicKey          OCTET STRING (SIZE(32))
}

-- Fulfillment
Ed25519Sha512Fulfillment ::= SEQUENCE {
    publicKey          OCTET STRING (SIZE(32)),
    signature          OCTET STRING (SIZE(64))
}
```

8.5.3. Condition Format

The fingerprint of an ED25519-SHA-256 condition is the SHA-256 digest of the DER encoded Ed25519 public key included as the only value within a SEQUENCE. While the public key is already very small and constant size, we wrap it in a SEQUENCE type and hash it for consistency with the other types.

8.5.4. Fulfillment

The fulfillment of an ED25519-SHA-256 crypto-condition is an Ed25519Sha512Fulfillment which is a SEQUENCE of:

publicKey An octet string containing the Ed25519 public key.

signature An octet string containing the Ed25519 signature.

8.5.5. Validating

An ED25519-SHA-256 fulfillment is valid iff :

1. F.signature is valid for the message M, given the ED25519 public key F.publicKey.
2. D is equal to C.

8.5.6. Example

```
exampleEd25519Condition Condition ::=
```

```
  ed25519Sha256 : {
    fingerprint '799239AB A8FC4FF7 EABFBC4C 44E69E8B DFED9933 24E12ED6 4792ABE2
89CF1D5F'H,
    cost 131072
  }
```

```
exampleEd25519Fulfillment Fulfillment ::=
```

```
  ed25519Sha256 : {
    publicKey 'D75A9801 82B10AB7 D54BFED3 C964073A 0EE172F3 DAA62325 AF021A68 F
707511A'H,
    signature 'E5564300 C360AC72 9086E2CC 806E828A 84877F1E B8E5D974 D873E065 2
2490155
                    5FB88215 90A33BAC C61E3970 1CF9B46B D25BF5F0 595BBE24 65514143 8
E7A100B'H
  }
```

```
exampleEd25519FingerprintContents Ed25519FingerprintContents ::= {
```

```
  publicKey 'D75A9801 82B10AB7 D54BFED3 C964073A 0EE172F3 DAA62325 AF021A68 F
707511A'H
}
```

9. URI Encoding Rules

Conditions can be encoded as URIs per the rules defined in the Named Information specification, [RFC6920]. There are no URI encoding rules for fulfillments.

Applications that require a string encoding for fulfillments MUST use an appropriate string encoding of the DER encoded binary representation of the fulfillment. No string encoding is defined in this specification. For consistency with the URI encoding of conditions, BASE64URL is recommended as described in [RFC4648], Section 5.

The URI encoding is only used to encode top-level conditions and never for sub-conditions. The binary encoding is considered the canonical encoding.

9.1. Condition URI Format

Conditions are represented as URIs using the rules defined in [RFC6920] where the object being hashed is the DER encoded fingerprint content of the condition as described for the specific condition type.

While [RFC6920] allows for truncated hashes, implementations using the Named Information URI schemes for crypto-conditions MUST only use untruncated SHA-256 hashes (Hash Name: sha-256, ID: 1 from the "Named Information Hash Algorithm Registry" defined in [RFC6920]).

9.2. New URI Parameter Definitions

[RFC6920] established the IANA registry of "Named Information URI Parameter Definitions". This specification defines three new definitions that are added to that registry and passed in URI encoded conditions as query string parameters.

9.2.1. Parameter: Fingerprint Type (fpt)

The "type" parameter indicates the type of condition that is represented by the URI. The value MUST be one of the names from the Crypto-Condition Type Registry (Appendix E.1).

9.2.2. Parameter: Cost (cost)

The cost parameter is the cost of the condition that is represented by the URI.

9.2.3. Parameter: Subtypes (subtypes)

The subtypes parameter indicates the types of conditions that are subtypes of the condition represented by the URI. The value MUST be a comma-separated list of names from the Crypto-Condition Type Registry (Appendix E.1).

The subtypes list MUST exclude the type of the root crypto-condition. Specifically, the value of the "fpt" parameter should not appear in the list of subtypes.

For example, if a threshold condition contains another threshold condition as well as a prefix condition, then its URI query parameters would appear like this:

```
ni:///...?cost=30&fpt=threshold-sha-256&subtypes=prefix-sha-256
```

Notice that the "subtypes" parameter does not contain "threshold-sha-256" because that type is already indicated in the "fpt" parameter.

The commas in the list should be treated as reserved characters per [RFC3986] and MUST not be percent encoded when used as list delimiters in the subtypes parameter.

9.2.3.1. Subtype Parameter Value Ordering

The subtypes list MUST be ordered by the type id value of each type, in ascending lexicographical order. That is, "preimage-sha-256" MUST appear before "prefix-sha-256", which MUST appear before "threshold-sha-256", and so on.

9.3. Condition URI Parameter Ordering

The parameters of a condition URI MUST appear in ascending lexicographical order based upon the name of each parameter. For example, the "cost" parameter must appear before the "fpt" parameter, which must appear before the "subtypes" parameter.

10. Example Condition

An example condition (PREIMAGE-SHA-256):

```
0x00000000 A0 25 80 20 7F 83 B1 65 7F F1 FC 53 B9 2D C1 81 .%.e...S.-...
0x00000010 48 A1 D6 5D FC 2D 4B 1F A3 D6 77 28 4A DD D2 00 H..]-K...w(J...
0x00000020 12 6D 90 69 81 01 0C .m.i...
```

```
ni:///sha-256;f4OxZX_x_F05LcGBSKHWXfwtSx-jlncost3SABJtkGk?fpt=preimage-sha-256&c
ost=12
```

The example has the following attributes:

| Field | Value | Description |
|--------------------|---|---|
| scheme | "ni:///" | The named information scheme. |
| hash function name | "sha-256" | The fingerprint is hashed with the SHA-256 digest function |
| fingerprint | "f40xZX_x_F05LcGBSKHWXfwtSx-jlncoSt3SABJtkGk" | The fingerprint for this condition. |
| type | "preimage-sha-256" | This is a PREIMAGE-SHA-256 (Section 8.1) condition. |
| cost | "12" | The fulfillment payload is 12 bytes long, therefore the cost is 12. |

11. References

11.1. Normative References

[I-D.irtf-cfrg-eddsa]

Josefsson, S. and I. Liusvaara, "Edwards-curve Digital Signature Algorithm (EdDSA)", draft-irtf-cfrg-eddsa-08 (work in progress), August 2016.

[itu.X680.2015]

International Telecommunications Union, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", August 2015, <<http://handle.itu.int/11.1002/1000/12479>>.

[itu.X690.2015]

International Telecommunications Union, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", August 2015, <<http://handle.itu.int/11.1002/1000/12483>>.

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, DOI 10.17487/RFC3280, April 2002, <<https://www.rfc-editor.org/info/rfc3280>>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, DOI 10.17487/RFC4055, June 2005, <<https://www.rfc-editor.org/info/rfc4055>>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.

[RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013, <<https://www.rfc-editor.org/info/rfc6920>>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.

11.2. Informative References

[KEYLENGTH-RECOMMENDATION]

"BlueKrypt - Cryptographic Key Length Recommendation", September 2015, <<https://www.keylength.com/en/compare/>>.

[LARGE-RSA-EXPONENTS]

"Imperial Violet - Very large RSA public exponents (17 Mar 2012)", March 2012,
<<https://www.imperialviolet.org/2012/03/17/rsados.html>>.

[NIST-KEYMANAGEMENT]

Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid,
"NIST - Recommendation for Key Management - Part 1 -
General (Revision 3)", July 2012,
<[http://csrc.nist.gov/publications/nistpubs/800-57/
sp800-57_part1_rev3_general.pdf](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf)>.

[OPENSSL-X509-CERT-EXAMPLES]

"OpenSSL - X509 certificate examples for testing and
verification", July 2012,
<<http://fm4dd.com/openssl/certexamples.htm>>.

[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>>.

[RFC3110] Eastlake 3rd, D., "RSA/SHA-1 SIGs and RSA KEYS in the
Domain Name System (DNS)", RFC 3110, DOI 10.17487/RFC3110,
May 2001, <<https://www.rfc-editor.org/info/rfc3110>>.

[RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton,
J., and M. Thomas, "DomainKeys Identified Mail (DKIM)
Signatures", RFC 4871, DOI 10.17487/RFC4871, May 2007,
<<https://www.rfc-editor.org/info/rfc4871>>.

[USING-RSA-EXPONENT-OF-65537]

"Cryptography - StackExchange - Impacts of not using RSA
exponent of 65537", November 2014,
<[https://crypto.stackexchange.com/questions/3110/
impacts-of-not-using-rsa-exponent-of-65537](https://crypto.stackexchange.com/questions/3110/impacts-of-not-using-rsa-exponent-of-65537)>.

11.3. URIs

[1] <https://interledger.org/>

[2] <mailto:ledger@ietf.org>

[3] <https://github.com/rfcs/crypto-conditions#test-vectors>

[4] <https://github.com/interledgerjs/five-bells-condition>

- [5] <https://github.com/hyperledger/quilt/tree/master/crypto-conditions>
- [6] <https://github.com/bigchaindb/cryptoconditions>
- [7] <https://github.com/go-interledger/cryptoconditions>
- [8] <https://github.com/jtremback/crypto-conditions>
- [9] <https://github.com/libscott/cryptoconditions-hs>
- [10] <http://www.iana.org/assignments/crypto-condition-types>

Appendix A. Security Considerations

This specification has a normative dependency on a number of other specifications with extensive security considerations therefore the considerations defined for SHA-256 hashing and RSA signatures in [RFC8017] and [RFC4055] and for ED25519 signatures in [I-D.irtf-cfrg-eddsa] must be considered.

The cost and subtypes values of conditions are provided to allow implementations to evaluate their ability to validate a fulfillment for the given condition later.

Appendix B. Test Values

Test vectors have been prepared at: <https://github.com/rfcs/crypto-conditions#test-vectors> [3]

Appendix C. Implementations

Implementations of this specification are known to be available in the following languages:

- o JavaScript: <https://github.com/interledgerjs/five-bells-condition> [4]
- o Java: <https://github.com/hyperledger/quilt/tree/master/crypto-conditions> [5]
- o Python: <https://github.com/bigchaindb/cryptoconditions> [6]
- o Go: <https://github.com/go-interledger/cryptoconditions> [7]
- o Go: <https://github.com/jtremback/crypto-conditions> [8]
- o Haskell: <https://github.com/libscott/cryptoconditions-hs> [9]

Appendix D. ASN.1 Module

```
--<ASN1.PDU CryptoConditions.Condition, CryptoConditions.Fulfillment>--

Crypto-Conditions DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- Conditions

Condition ::= CHOICE {
    preimageSha256      [0] SimpleSha256Condition,
    prefixSha256        [1] CompoundSha256Condition,
    thresholdSha256     [2] CompoundSha256Condition,
    rsaSha256           [3] SimpleSha256Condition,
    ed25519Sha256       [4] SimpleSha256Condition
}

SimpleSha256Condition ::= SEQUENCE {
    fingerprint          OCTET STRING (SIZE(32)),
    cost                 INTEGER (0..4294967295)
}

CompoundSha256Condition ::= SEQUENCE {
    fingerprint          OCTET STRING (SIZE(32)),
    cost                 INTEGER (0..4294967295),
    subtypes             ConditionTypes
}

ConditionTypes ::= BIT STRING {
    preImageSha256      (0),
    prefixSha256        (1),
    thresholdSha256     (2),
    rsaSha256           (3),
    ed25519Sha256       (4)
}

-- Fulfillments

Fulfillment ::= CHOICE {
    preimageSha256      [0] PreimageFulfillment ,
    prefixSha256        [1] PrefixFulfillment,
    thresholdSha256     [2] ThresholdFulfillment,
    rsaSha256           [3] RsaSha256Fulfillment,
    ed25519Sha256       [4] Ed25519Sha512Fulfillment
}

PreimageFulfillment ::= SEQUENCE {
    preimage             OCTET STRING
}
```

```
PrefixFulfillment ::= SEQUENCE {
    prefix          OCTET STRING,
    maxMessageLength INTEGER (0..4294967295),
    subfulfillment  Fulfillment
}
```

```
ThresholdFulfillment ::= SEQUENCE {
    subfulfillments SET OF Fulfillment,
    subconditions   SET OF Condition
}
```

```
RsaSha256Fulfillment ::= SEQUENCE {
    modulus          OCTET STRING,
    signature        OCTET STRING
}
```

```
Ed25519Sha512Fulfillment ::= SEQUENCE {
    publicKey        OCTET STRING (SIZE(32)),
    signature        OCTET STRING (SIZE(64))
}
```

-- Fingerprint Content

-- The PREIMAGE-SHA-256 condition fingerprint content is not DER encoded
-- The fingerprint content is the preimage

```
PrefixFingerprintContents ::= SEQUENCE {
    prefix          OCTET STRING,
    maxMessageLength INTEGER (0..4294967295),
    subcondition    Condition
}
```

```
ThresholdFingerprintContents ::= SEQUENCE {
    threshold       INTEGER (1..65535),
    subconditions   SET OF Condition
}
```

```
RsaFingerprintContents ::= SEQUENCE {
    modulus          OCTET STRING
}
```

```
Ed25519FingerprintContents ::= SEQUENCE {
    publicKey        OCTET STRING (SIZE(32))
}
```

END

Appendix E. IANA Considerations

E.1. Crypto-Condition Type Registry

The following initial entries should be added to the Crypto-Condition Type registry to be created and maintained at (the suggested URI) <http://www.iana.org/assignments/crypto-condition-types> [10]:

The following types are registered:

| Type ID | Type Name |
|---------|-------------------|
| 0 | PREIMAGE-SHA-256 |
| 1 | PREFIX-SHA-256 |
| 2 | THRESHOLD-SHA-256 |
| 3 | RSA-SHA-256 |
| 4 | ED25519 |

Table 1: Crypto-Condition Types

Authors' Addresses

Stefan Thomas
Ripple
300 Montgomery Street
San Francisco, CA 94104
US

Phone: -----
Email: stefan@ripple.com
URI: <https://www.ripple.com>

Rome Reginelli
Ripple
300 Montgomery Street
San Francisco, CA 94104
US

Phone: -----
Email: rome@ripple.com
URI: <https://www.ripple.com>

Adrian Hope-Bailie
Ripple
300 Montgomery Street
San Francisco, CA 94104
US

Phone: -----
Email: adrian@ripple.com
URI: <https://www.ripple.com>