DPRIVE WG
Internet-Draft
Intended status: Standards Track
Expires: September 4, 2020

T. Reddy
McAfee
D. Wing
Citrix
M. Richardson
Sandelman Software Works
M. Boucadair
Orange

March 3, 2020

DNS Server Privacy Statement and Filtering Policy with Assertion Token
draft-reddy-dprive-dprive-privacy-policy-03

Abstract

Users may want to control how their DNS queries are handled by DNS servers so they can configure their system to use DNS servers that comply with their privacy and DNS filtering expectations.

This document defines a mechanism for a DNS server to communicate its privacy statement URL and filtering policy to a DNS client. This communication is cryptographically signed to attest its authenticity. By evaluating the DNS privacy statement, filtering policy and the signatory, the user can choose a DNS server that best supports his/her desired privacy and filtering policy. This token is particularly useful for DNS-over-TLS and DNS-over-HTTPS servers that are either public resolvers or are discovered on a local network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC7626] discusses DNS privacy considerations in both "on the wire" (Section 2.4 of [RFC7626]) and "in the server" (Section 2.5 of [RFC7626] contexts. In recent years there has also been an increase in the availability of "public resolvers" [RFC8499] which DNS clients may be pre-configured to use instead of the default network resolver because they offer a specific feature (e.g., good reachability, encrypted transport, strong privacy policy, (lack of) filtering, etc.). The DNS Recursive Operator Privacy (DROP) statement explained in [I-D.ietf-dprive-bcp-op] outlines the recommended contents an DNS operator should publish, thereby providing a means for users to evaluate the privacy properties of a given DNS service. While a human can review the privacy statement of a DNS server operator, but the challenge is the user has to search to find the URL that points to the human readable privacy policy information of the DNS server. Also, a user does not know if a locally-discovered server performs DNS-based filtering.

For DNS servers operated on the local network, the DNS client can be securely bootstrapped to discover and authenticate DNS-over-HTTPS (DoH) [RFC8484] and DNS-over-TLS (DoT) [RFC7858] servers provided by a local network, for example using the technique proposed in [I-D.reddy-dprive-bootstrap-dns-server]. This document defines a retrievable DNS server policy permitting the user to consent to using a certain DNS server that meets their needs.

The cryptographically signed policy allows a DNS client to connect to multiple DNS servers and prompt the user to review the DNS privacy statements to select the DNS server that adheres to the privacy preserving data policy and DNS filtering expectations of the user. For example, a browser with pre-configured DNS-over-HTTPS server can discover the DNS-over-HTTPS server provided the local network, connects to both the DNS servers, gets the policy information from each of the DNS servers, validates the signatures and prompts the user to review the privacy policy statements of both the local and

public DNS server. If both servers meet the privacy preserving data policy and DNS filtering requirements of the user, the user can select to use the local DNS server. A quality implementation can avoid presenting this information to the user if the DNS server's policies have not changed.

2. Use Cases Overview

The mechanism for a DNS server to communicate its cryptographically signed policies to a DNS client contribute to solve the following problems in various deployments:

- Typically Enterprise networks do not assume that all devices in their network are managed by the IT team or Mobile Device Management (MDM) devices, especially in the quite common BYOD (Bring Your Own Device) scenario. The mechanism specified in this document can be used by users of the BYOD devices to determine if the DNS server on the local network complies with the user's privacy policy and DNS filtering expectations.

- The user selects specific well-known networks (e.g., organization for which a user works or a user works temporarily within another corporation) to learn the privacy policy statement and filtering policy of the local DNS server. Then, the user can choose to use the discovered DoT or DoH server. If that discovered DoT/DoH server does not meet the privacy preserving data policy and filtering requirements of the user, the user can instruct the DNS client to take appropriate actions. For example, the action can be to use the local DNS server only to access internal-only DNS names and use another DNS server (adhering with his/her expectations) for public domains.

- The policy information signals the presence of DNS-based content filtering in the attached network. If the network is well-known to the user and the local DNS server meets the privacy requirements of the user, the DNS client can continue to use encrypted connection with the local DoT/DoH server. If the error code returned by the DNS server indicates access to the domain is blocked because of internal security policy [I-D.ietf-dnsop-extended-error], the DNS client can securely identify access to the domain is censored by the network.

- The signed policy contains an URL that points to a human-readable privacy policy information of the DNS server for the user to review and can make an informed decision whether the DNS server is trustworthy to honor the privacy of the user. The DNS Push Notifications mechanism defined in [I-D.ietf-dnsop-extended-error] can be used by the DNS client to be asynchronously notified when

the policy change occurs. The client automatically learns updates to the policy of the DNS server, and whenever the privacy statement of the DNS server changes, the client can notify the user to re-evaluate the updated privacy statement.

3. Terminology

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

This document uses the terms defined in [RFC8499].

4. Policy Assertion Token (PAT): Overview

JSON Web Token (JWT) [RFC7519] and JSON Web Signature (JWS) [RFC7515] and related specifications define a standard token format that can be used as a way of encapsulating claimed or asserted information with an associated digital signature using X.509 based certificates. JWT provides a set of claims in JSON format that can accommodate asserted policy information of the DoT/DoH server. Additionally, JWS provides a path for updating methods and cryptographic algorithms used for the associated digital signatures.

JWS defines the use of JSON data structures in a specified canonical format for signing data corresponding to JOSE header, JWS Payload, and JWS Signature. The next sections define the header and claims that MUST be minimally used with JWT and JWS for privacy assertion token.

The Policy Assertion Token (PAT) specifically uses this token format and defines claims that convey the policy information of DoT/ DoH server. The client can retrieve the PAT object using the method discussed in [I-D.ietf-dnsop-resolver-information]. The signature of PAT object can be validated by the DNS client. If the signer and the contents of the PAT object comply with the user's requirements, the user's client software can use that DNS server.

The PAT object is signed by the DNS server's domain that is authoritative to assert the DNS server policy information. This authority is represented by the certificate credentials and the signature.

For example, the PAT object could be created by the domain hosting the DoT/DoH server and optionally by a third party who performed privacy and security audit of the DoT/DoH server. The DNS client

needs to have the capability to verify the digital signature and to parse the PAT object.

5. PAT Header

The JWS token header is a JOSE header (Section 4 of [RFC7515]) that defines the type and encryption algorithm used in the token.

PAT header MUST include, at a minimum, the header parameters defined in Sections 5.1, 5.2, and 5.3.

5.1. 'typ' (Type) Header Parameter

The 'typ' (Type) Header Parameter is defined Section 4.1.9 of [RFC7515] to declare the media type of the complete JWS.

For PAT Token the 'typ' header MUST be the string 'pat'. This represents that the encoded token is a JWT of type pat.

5.2. 'alg' (Algorithm) Header Parameter

The 'alg' (Algorithm) Header Parameter is defined in Section 4.1.1 of [RFC7515]. It specifies the JWS signature cryptographic algorithm. It also refers to a list of defined 'alg' values as part of a registry established by JSON Web Algorithms (JWA) [RFC7518] Section 3.1.

For the creation and verification of PAT tokens and their digital signatures, implementations MUST support ES256 as defined in Section 3.4 of [RFC7518]. Implementations MAY support other algorithms registered in the JSON Web Signature and Encryption Algorithms registry created by [RFC7518]. The content of that registry may be updated in the future depending on cryptographic strength requirements guided by current security best practice. The mandatory-to-support algorithm for PAT tokens may likewise be updated in the future.

Implementations of PAT digital signatures using ES256 as defined above SHOULD use deterministic ECDSA when supported for the reasons stated in [RFC6979].

5.3. 'x5u' (X.509 URL) Header Parameter

As defined in Section 4.1.5 of [RFC7515], the 'x5u' header parameter defines a URI [RFC3986] referring to the resource for the X.509 public key certificate or certificate chain [RFC5280] corresponding to the key used to digitally sign the JWS. Generally, as defined in

Section 4.1.5 of [RFC7515] this corresponds to an HTTPS or DNSSEC resource using integrity protection.

5.4. An Example of PAT Header

An example of the PAT header is shown in Figure 1. It includes the specified PAT type, ES256 algorithm, and an URI referencing the network location of the certificate needed to validate the PAT signature.

```
{
  "typ":"pat",
  "alg":"ES256",
  "x5u":"https://cert.example.com/pat.cer"
}
```

Figure 1: A PAT Header Example

6. PAT Payload

The token claims consists of the policy information of the DNS server which needs to be verified at the DNS client. These claims follow the definition of a JWT claim (Secion 4 of [RFC7519]) and are encoded as defined by the JWS Payload (Section 3 of [RFC7515]).

PAT defines the use of a standard JWT-defined claim as well as custom claims corresponding to the DoT or DoH servers.

Claim names MUST use the US-ASCII character set. Claim values MAY contain characters that are outside the ASCII range, however they MUST follow the default JSON serialization defined in Section 7 of [RFC7519].

6.1. JWT Defined Claims

6.1.1. 'iat' - Issued At Claim

The JSON claim MUST include the 'iat' (Section 4.1.6 of [RFC7519]) defined claim "Issued At". The 'iat' should be set to the date and time of issuance of the JWT. The time value should be of the format (NumericDate) defined in Section 2 of [RFC7519].

6.1.2. 'exp' - Expiration Time Claim

The JSON claim MUST include the 'exp' (Section 4.1.4 of [RFC7519]) defined "claim Expiration Time". The 'exp' should be set to specify the expiration time on or after which the JWT is not accepted for processing. The PAT object should expire after a reasonable

duration. A short expiration time for the PAT object periodically reaffirms the policy information of the DNS server to the DNS client and ensures the DNS client does not use outdated policy information. If the DNS client knows the PAT object has expired, it should make another request to get the new PAT object from the DNS server.

6.2. PAT Specific Claims

6.2.1. DNS Server Identity Claims

The DNS server identity is represented by a claim that is required for PAT: the 'server' claim. The 'server' MUST contain claim values that are identity claim JSON objects where the child claim name represents an identity type and the claim value is the identity string, both defined in subsequent subsections.

These identities can be represented as either authentication domain name (ADN) (defined in [RFC8310]) or Uniform Resource Indicators (URI).

6.2.1.1. 'adn' - Authentication Domain Name Identity

If the DNS server identity is an ADN, the claim name representing the identity MUST be 'adn'. The claim value for the 'adn' claim is the ADN.

6.2.1.2. 'uri' - URI Identity

If the DNS server identity is of the form URI, as defined in [RFC3986], the claim name representing the identity MUST be 'uri' and the claim value is the URI form of the DNS server identity.

As a reminder, if DoH is supported by the DNS server, the DNS client uses the https URI scheme (Section 3 of [RFC8484]).

6.2.2. 'policyinfo' (Policy Information) Claim

The 'policyinfo' claim MUST be formatted as a JSON object. The 'policyinfo' claim contains the policy information of the DNS server, it includes the following attributes:

filtering: If the DNS server changes some of the answers that it returns based on policy criteria, such as to prevent access to malware sites or objectionable content. This optional attribute has the following structure:

malwareblocking: The DNS server offers malware blocking service. If access to domains is blocked on threat data, the parameter value is set to 'true'.

policyblocking: If access to domains is blocked on a blacklist or objectionable content, the parameter value is set to 'true'.

qnameminimization: If the DNS server implements QNAME minimisation [RFC7816] to improve DNS privacy. If the parameter value is set to 'true', QNAME minimisation is supported by the DNS server. This is a mandatory attribute.

privacyurl: A URL that points to the privacy policy information of the DNS server. This is a mandatory attribute.

auditurl: A URL that points to the security assessment report of the DNS server by a third party auditor. This is an optional attribute.

6.2.3. Example

Figure 2 shows an example of policy information.


```
{
  "server":{
      "adn":["example.com"]
  },
  "iat":1443208345,
  "exp":1443640345,
  "policyinfo": {
     "filtering": {
         "malwareblocking": true,
         "policyblocking": false
     },
     "qnameminimization":false,
     "privacyurl": "https://example.com/commitment-to-privacy/"
  }
}
```


Figure 2: An Example of Policy Information

7. PAT Signature

The signature of the PAT is created as specified in Section 5.1 of [RFC7515] (Steps 1 through 6). PAT MUST use the JWS Protected Header.

For the JWS Payload and the JWS Protected Header, the lexicographic ordering and white space rules described in Section 5 and Section 6, and JSON serialization rules in Section 9 MUST be followed.

The PAT is cryptographically signed by the domain hosting the DNS server and optionally by a third party who performed privacy and security audit of the DNS server.

The policy information is attested using "Organization Validation" (OV) or "Extended Validation" (EV) certificates to avoid bad actors taking advantage of this mechanism to advertise DoH/DoT servers for illegitimate and fraudulent purposes meant to trick DNS clients into believing that they are using a legitimate DoT/DoH server hosted to provide privacy for DNS transactions.

Alternatively, a DNS client has to be configured to trust the leaf of the signer of the PAT object. That is, trust of the signer MUST NOT be determined by validating the signer via the OS or the browser trust chain because that would allow any arbitrary entity to operate a DNS server and assert any sort of policy.

Appendix A provides an example of how to follow the steps to create the JWS Signature.

JWS JSON serialization (Step 7 in Section 5.1 of [RFC7515]) is supported for PAT to enable multiple signatures to be applied to the PAT object. For example, the PAT object can be cryptographically signed by the domain hosting the DNS server and by a third party who performed privacy and security audit of the DNS server.

Appendix B includes an example of the full JWS JSON serialization representation with multiple signatures.

Section 5.1 of [RFC7515] (Step 8) describes the method to create the final JWS Compact Serialization form of the PAT Token.

8. Extending PAT

PAT includes the minimum set of claims needed to securely assert the policy information of the DNS server. JWT supports a mechanism to add additional asserted or signed information by simply adding new claims. PAT can be extended beyond the defined base set of claims to represent other DNS server information requiring assertion or validation. Specifying new claims follows the baseline JWT procedures (Section 10.1 of [RFC7519]). Understanding new claims on the DNS client is optional. The creator of a PAT object cannot assume that the DNS client will understand the new claims.

9. Deterministic JSON Serialization

JSON objects can include spaces and line breaks, and key value pairs can occur in any order. It is therefore a non-deterministic string format. In order to make the digital signature verification work deterministically, the JSON representation of the JWS Protected Header object and JWS Payload object MUST be computed as follows.

The JSON object MUST follow the following rules. These rules are based on the thumbprint of a JSON Web Key (JWK) as defined in Section 3 of [RFC7638] (Step 1).

1. The JSON object MUST contain no whitespace or line breaks before or after any syntactic elements.

2. JSON objects MUST have the keys ordered lexicographically by the Unicode [UNICODE] code points of the member names.

3. JSON value literals MUST be lowercase.

4. JSON numbers are to be encoded as integers unless the field is defined to be encoded otherwise.

5. Encoding rules MUST be applied recursively to member values and array values.

9.1. Example PAT Deterministic JSON Form

This section demonstrates the deterministic JSON serialization for the example PAT Payload shown in Section 6.2.3.

The initial JSON object is shown in Figure 3.


```
{
  "server":{
      "adn":["example.com"]
  },
  "iat":1443208345,
  "exp":1443640345,
  "policyinfo": {
     "qnameminimization":false,
     "privacyurl": "https://example.com/commitment-to-privacy/"
  }
}
```


Figure 3: Initial JSON Object

The parent members of the JSON object are as follows, in lexicographic order: "exp", "iat", "policyinfo", "server".

The final constructed deterministic JSON serialization representation, with whitespace and line breaks removed, (with line breaks used for display purposes only) is:


```
{"exp":1443640345,"iat":1443208345,
"policyinfo":{"privacyurl":"https://example.com/commitment-to-privacy/",
"qnameminimization":false},"server":{"adn":["example.com"]}}
```


Figure 4: Deterministic JSON Form

10. Privacy Considerations

Users are expected to indicate to their system in some way that they trust certain PAT signers (e.g., if working for Example, Inc., the user's system is configured to trust "example.com" signing the PAT). By doing so, the DNS client can automatically discover DoT/DoH server in specific networks, validate the PAT signature and the user can check if the human readable privacy policy information of the DNS server complies with user's privacy needs, prior to using that DoT/DoH server for DNS queries.

The DNS client MUST retrieve the human-readable privacy statement from the 'privacyurl' attribute to assist with that decision (e.g., display the privacy statement when it changes, show differences in previously-retrieved version, etc.). With the steps above, user consent is obtained prior to using a DoT/DoH server.

11. Security Considerations

The use of PAT object based on the validation of the digital signature and the associated certificate requires consideration of the authentication and authority or reputation of the signer to attest the policy information of the DNS server being asserted. Bad actors can host DNS-over-TLS and DNS-over-HTTPS servers, and claim the servers offer privacy but exactly do the opposite to invade the privacy of the user. Bad actor can get a domain name, host DNS-over-TLS and DNS-over-HTTPS servers, and get the DNS server certificate signed by a CA. The policy information will have to be attested using OV/EV certificates or a PAT object signer trusted by the DNS client to prevent the attack.

If the PAT object is asserted by a third party, it can do a "time of check" but the DNS server is susceptible of "time of use" attack. For example, changes to the policy of the DNS server can cause a disagreement between the auditor and the DNS server operation, hence

the PAT object needs to be also asserted by the domain hosting the DNS server. In addition, the PAT object needs to have a short expiration time (e.g., 7 days) to ensure the DNS server's domain re-asserts the policy information and limits the damage from change in policy and mis-issuance.

12. IANA Considerations

12.1. Media Type Registration

12.1.1. Media Type Registry Contents Additions Requested

This section registers the 'application/pat' media type [RFC2046] in the 'Media Types' registry in the manner described in [RFC6838], which can be used to indicate that the content is a PAT defined JWT.

- Type name: application
- Subtype name: pat
- Required parameters: n/a
- Optional parameters: n/a
- Encoding considerations: 8bit; application/pat values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters..
- Security considerations: See the Security Considerations Section of [RFC7515].
- Interoperability considerations: n/a
- Published specification: [TODO this document]
- Applications that use this media type: DNS
- Fragment identifier considerations: n/a
- Additional information:

  Magic number(s): n/a File extension(s): n/a Macintosh file type code(s): n/a
- Person & email address to contact for further information: Tirumaleswar Reddy, kondtir@gmail.com
- Intended usage: COMMON

- o Restrictions on usage: none
- o Author: Tirumaleswar Reddy, kondtir@gmail.com
- o Change Controller: IESG
- o Provisional registration? No

12.2. JSON Web Token Claims Registration

12.2.1. Registry Contents Additions Requested

- o Claim Name: 'server'
- o Claim Description: DNS server identity
- o Change Controller: IESG
- o Specification Document(s): Section 6.2.1 of [TODO this document]
- o Claim Name: 'policyinfo'
- o Claim Description: Policy information of DNS server.
- o Change Controller: IESG
- o Specification Document(s): Section 6.2.2 of [TODO this document]

12.3. DNS Resolver Information Registration

IANA will add the names filtering, qnameminimization, privacyurl and auditurl to the DNS Resolver Information registry defined in Section 5.2 of [I-D.ietf-dnsop-resolver-information].

13. Acknowledgments

This specification leverages some of the work that has been done in [RFC8225]. Thanks to Ted Lemon, Paul Wouters and Shashank Jain for the discussion and comments.

14. References

14.1. Normative References

[I-D.ietf-dnsop-resolver-information]
Sood, P., Arends, R., and P. Hoffman, "DNS Resolver Information Self-publication", draft-ietf-dnsop-resolver-information-01 (work in progress), February 2020.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[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>.

[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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

14.2. Informative References

[I-D.ietf-dnsop-extended-error]
Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", draft-ietf-dnsop-extended-error-14 (work in progress), January 2020.

[I-D.ietf-dprive-bcp-op]
Dickinson, S., Overeinder, B., Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", draft-ietf-dprive-bcp-op-08 (work in progress), January 2020.

[I-D.reddy-dprive-bootstrap-dns-server]
Reddy.K, T., Wing, D., Richardson, M., and M. Boucadair, "A Bootstrapping Procedure to Discover and Authenticate DNS-over-(D)TLS and DNS-over-HTTPS Servers", draft-reddy-dprive-bootstrap-dns-server-07 (work in progress), February 2020.

[RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <https://www.rfc-editor.org/info/rfc7626>.

[RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016, <https://www.rfc-editor.org/info/rfc7816>.

[RFC8225] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/info/rfc8225>.

[RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", June 2016, <http://www.unicode.org/versions/latest/>.

Appendix A. Example ES256 based PAT JWS Serialization and Signature

For PAT, there will always be a JWS with the following members:

- o 'protected', with the value BASE64URL(UTF8(JWS Protected Header))
- o 'payload', with the value BASE64URL (JWS Payload)
- o 'signature', with the value BASE64URL(JWS Signature)

This example will follow the steps in JWS [RFC7515] Section 5.1, steps 1-6 and 8 and incorporates the additional serialization steps required for PAT.

Step 1 for JWS references the JWS Payload, an example PAT Payload is as follows:


```
{
  "server":{
      "adn":["example.com"]
  },
  "iat":1443208345,
  "exp":1443640345,
  "policyinfo": {
     "filtering": {
         "malwareblocking": true,
         "policyblocking": false
     },
     "qnameminimization":false,
     "privacyurl": "https://example.com/commitment-to-privacy/"
  }
}
```


This would be serialized to the form (with line break used for display purposes only):


```
{"exp":1443640345,"iat":1443208345,"policyinfo":{
"filtering":{"malwareblocking": true,"policyblocking": false},
"privacyurl":"https://example.com/commitment-to-privacy/",
"qnameminimization":false},"server":{"adn":["example.com"]}}
```

Step 2 Computes the BASE64URL(JWS Payload) producing this value (with line break used for display purposes only):


```
eyJleHAiOjE0NDM2NDAzNDUsImlhdCI6MTQ0MzIwODM0NSwicG9saWN5aW5mbyI6e
yJmaWx0ZXJpbmciOnsibWFsd2FyZWJsb2NraW5nIjp0cnVlLCJwb2xpY3libG9ja2
luZyI6ZmFsc2V9LCJwcml2YWN5dXJsIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9jb21
taXRtZW50LXRvLXByaXZhY3kvIiwicW5hbWVtaW5pbWl6YXRpb24iOmZhbHNlfSwi
c2VydmVyIjp7ImFkbiI6WyJleGFtcGxlLmNvbSJdfX0
```


For Step 3, an example PAT Protected Header comprising the JOSE Header is as follows:


```
{
  "alg":"ES256",
  "typ":"pat",
  "x5u":"https://cert.example.com/pat.cer"
}
```


This would be serialized to the form (with line break used for display purposes only):


```
{"alg":"ES256","typ":"pat","x5u":"https://cert.example.com
/pat.cer"}
```


Step 4 Performs the BASE64URL(UTF8(JWS Protected Header)) operation and encoding produces this value (with line break used for display purposes only):


```
eyJhbGciOiJFUzI1NiIsInR5cCI6InBhdCIsIng1dSI6Imh0dHBzOi8vY2VydC5l
eGFtcGxlLmNvbS9wYXQuY2VyIn0
```


Step 5 and Step 6 performs the computation of the digital signature of the PAT Signing Input ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload)) using ES256 as the algorithm and the BASE64URL(JWS Signature).


```
4vQEAF_Vlp1Tr6sJmS4pnIKDRmIjH8EZzY5BMT2qJCHD8PmjBktWVnlmbmyHs05G
KauRBdIFnfp3oDPbE0Jq4w
```


Step 8 describes how to create the final PAT token, concatenating the values in the order Header.Payload.Signature with period ('.') characters. For the above example values this would produce the

following (with line breaks between period used for readability purposes only):


```
eyJhbGciOiJFUzI1NiIsInR5cCI6InBhdCIsIng1dSI6Imh0dHBzOi8vY2VydC5l
eGFtcGxlLmNvbS9wYXQuY2VyIn0
.
eyJleHAiOjE0NDM2NDAzNDUsImlhdCI6MTQ0MzIwODM0NSwicG9saWN5aW5mbyI6e
yJmaWx0ZXJpbmciOnsibWFsd2FyZWJsb2NraW5nIjp0cnVlLCJwb2xpY3libG9ja2
luZyI6ZmFsc2V9LCJwcml2YWN5dXJsIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9jb21
taXRtZW50LXRvLXByaXZhY3kvIiwicW5hbWVtaW5pbWl6YXRpb24iOmZhbHNlfSwi
c2VydmVyIjp7ImFkbiI6WyJleGFtcGxlLmNvbSJdfX0
.
4vQEAF_Vlp1Tr6sJmS4pnIKDRmIjH8EZzY5BMT2qJCHD8PmjBktWVnlmbmyHs05G
KauRBdIFnfp3oDPbE0Jq4w
```


A.1. X.509 Private Key in PKCS#8 Format for ES256 Example**


```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgevZzL1gdAFr88hb2
OF/2NxApJCzGCEDdfSp6VQO30hyhRANCAAQRWz+jn65BtOMvdyHKcvjBeBSDZH2r
1RTwjmYSi9R/zpBnuQ4EiMnCqfMPWiZqB4QdbAd0E7oH50VpuZ1P087G
-----END PRIVATE KEY-----
```


A.2. X.509 Public Key for ES256 Example**


```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEEVs/o5+uQbTjL3chynL4wXgUg2R9
q9UU8I5mEovUf86QZ7kOBIjJwqnzD1omageEHWwHdBO6B+dFabmdT9POxg==
-----END PUBLIC KEY-----
```


Appendix B. Complete JWS JSON Serialization Representation with multiple Signatures

The JWS payload used in this example as follows.

```
{
  "server":{
      "adn":["example.com"]
  },
  "iat":1443208345,
  "exp":1443640345,
  "policyinfo": {
     "filtering": {
         "malwareblocking": true,
         "policyblocking": false
     },
     "qnameminimization":false,
     "privacyurl": "https://example.com/commitment-to-privacy/"
  }
}
```


This would be serialized to the form (with line break used for display purposes only):


```
{"exp":1443640345,"iat":1443208345,"policyinfo":{
"filtering":{"malwareblocking": true,"policyblocking": false},
"privacyurl":"https://example.com/commitment-to-privacy/",
"qnameminimization":false},"server":{"adn":["example.com"]}}
```


The JWS protected Header value used for the first signature is same as that used in the example in Appendix A. The X.509 private key used for generating the first signature is same as that used in the example in Appendix A.1.

The JWS Protected Header value used for the second signature is:


```
{
  "alg":"ES384",
  "typ":"pat",
  "x5u":"https://cert.audit-example.com/pat.cer"
}
```


The complete JWS JSON Serialization for these values is as follows (with line breaks within values for display purposes only):

```
{
  "payload":
       "eyJleHAiOjE0NDM2NDAzNDUsImlhdCI6MTQ0MzIwODM0NSwicG9saWN5aW5mbyI6
        eyJmaWx0ZXJpbmciOnsibWFsd2FyZWJsb2NraW5nIjp0cnVlLCJwb2xpY3libG9j
        a2luZyI6ZmFsc2V9LCJwcml2YWN5dXJsIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9j
        b21taXRtZW50LXRvLXByaXZhY3kvIiwicW5hbWVtaW5pbWl6YXRpb24iOmZhbHNl
        fSwic2VydmVyIjp7ImFkbiI6WyJleGFtcGxlLmNvbSJdfX0",
  "signatures":[
       {"protected":"eyJhbGciOiJFUzI1NiIsInR5cCI6InBhdCIsIng1dSI6Imh0dHB
        zOi8vY2VydC5leGFtcGxlLmNvbS9wYXQuY2VyIn0",
        "signature": "4vQEAF_Vlp1Tr6sJmS4pnIKDRmIjH8EZzY5BMT2qJCHD8PmjBk
        tWVnlmbmyHs05GKauRBdIFnfp3oDPbE0Jq4w"},
       {"protected":"eyJhbGciOiJFUzM4NCIsInR5cCI6InBhdCIsIng1dSI6Imh0dHB
        zOi8vY2VydC5hdWRpdC1leGFtcGxlLmNvbS9wYXQuY2VyIn0",
        "signature":666ag_mAqDa3Oyxo1DGXUocr0MmRjpXwq8kWp1S21mvs2-kPCIq3
        0xsBJt4apy-sq3VyJgIqzjijoFYURhHvupF0obo-IFUGSZ1YHBCX_MiyBwJQJjtp
        S91ujDatRTtZ"}]
}
```


B.1. X.509 Private Key in PKCS#8 format for E384 Example**


```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgevZzL1gdAFr88hb2
OF/2NxApJCzGCEDdfSp6VQO30hyhRANCAAQRWz+jn65BtOMvdyHKcvjBeBSDZH2r
1RTwjmYSi9R/zpBnuQ4EiMnCqfMPWiZqB4QdbAd0E7oH50VpuZ1P087G
-----END PRIVATE KEY-----
```


B.2. X.509 Public Key for ES384 Example**


```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEEVs/o5+uQbTjL3chynL4wXgUg2R9
q9UU8I5mEovUf86QZ7kOBIjJwqnzD1omageEHWwHdBO6B+dFabmdT9POxg==
-----END PUBLIC KEY-----
```


Authors' Addresses

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com

Michael C. Richardson
Sandelman Software Works
USA

Email: mcr+ietf@sandelman.ca

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com