Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 09, 2016

P. Wouters
Red Hat

March 08, 2016

Using DANE to Associate OpenPGP public keys with email addresses
draft-ietf-dane-openpgpkey-08

Abstract

OpenPGP is a message format for email (and file) encryption that lacks a standardized lookup mechanism to securely obtain OpenPGP public keys. DNS-Based Authentication of Named Entities ("DANE") is a method for publishing public keys in DNS. This document specifies a DANE method for publishing and locating OpenPGP public keys in DNS for a specific email address using a new OPENPGPKEY DNS Resource Record. Security is provided via Secure DNS, however the OPENPGPKEY record is not a replacement for verification of authenticity via the "Web Of Trust" or manual verification. The OPENPGPKEY record can be used to encrypt an email that would otherwise have to be sent unencrypted.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

OpenPGP [RFC4880] public keys are used to encrypt or sign email messages and files. To encrypt an email message, or verify a sender's OpenPGP signature, the email client or MTA needs to locate the recipient's OpenPGP public key.

OpenPGP clients have relied on centralized "well-known" key servers that are accessed using the HTTP Keyserver Protocol [HKP]. Alternatively, users need to manually browse a variety of different front-end websites. These key servers do not require a confirmation of the email address used in the User ID of the uploaded OpenPGP public key. Attackers can - and have - uploaded rogue public keys with other people's email addresses to these key servers.

Once uploaded, public keys cannot be deleted. People who did not pre-sign a key revocation can never remove their OpenPGP public key from these key servers once they have lost access to their private key. This results in receiving encrypted email that cannot be decrypted.

Therefore, these keyservers are not well suited to support email clients and MTA's to automatically encrypt email - especially in the absence of an interactive user.

This document describes a mechanism to associate a user's OpenPGP public key with their email address, using the OPENPGPKEY DNS RRtype. These records are published in the DNS zone of the user's email address. If the user loses their private key, the OPENPGPKEY DNS record can simply be updated or removed from the zone.

The OPENPGPKEY data is secured using Secure DNS [RFC4035]

The main goal of the OPENPGPKEY resource record is to stop passive attacks against plaintext emails. While it can also thwart some active attacks (such as people uploading rogue keys to keyservers in the hopes that others will encrypt to these rogue keys), this resource record is not a replacement for verifying OpenPGP public keys via the web of trust signatures, or manually via a fingerprint verification.

1.1. Experiment goal

This specification is one experiment in improving access to public keys for end-to-end email security. There are a range of ways in which this can reasonably be done, for OpenPGP or S/MIME, for example using the DNS, or SMTP, or HTTP. Proposals for each of these have been made with various levels of support in terms of implementation

and deployment. For each such experiment, specifications such as this will enable experiments to be carried out that may succeed or that may uncover technical or other impediments to large- or small-scale deployments. The IETF encourages those implementing and deploying such experiments to publicly document their experiences so that future specifications in this space can benefit.

This document defines an RRtype whose use is Experimental. The goal of the experiment is to see whether encrypted email usage will increase if an automated discovery method is available to MTA's and MUA's to help the enduser with email encryption key management.

It is unclear if this RRtype will scale to some of the larger email service deployments. Concerns have been raised about the size of the OPENPGPKEY record and the size of the resulting DNS zone files. This experiment hopefully will give the working group some insight into whether this is a problem or not.

If the experiment is successful, it is expected that the findings of the experiment will result in an updated document for standards track approval.

The OPENPGPKEY RRtype somewhat resembles the generic CERT record defined in [RFC4398]. However, the CERT record uses sub-typing with many different types of keys and certificates. It is suspected that its general application of very different protocols (PKIX versus OpenPGP) has been the cause for lack of implementation and deployment. Furthermore, the CERT record uses sub-typing, which is now considered to be a bad idea for DNS.

1.2. 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 RFC 2119 [RFC2119].

This document also makes use of standard DNSSEC and DANE terminology. See DNSSEC [RFC4033], [RFC4034], [RFC4035], and DANE [RFC6698] for these terms.

2. The OPENPGPKEY Resource Record

The OPENPGPKEY DNS resource record (RR) is used to associate an end entity OpenPGP Transferable Public Key (see Section 11.1 of [RFC4880] with an email address, thus forming a "OpenPGP public key association". A user that wishes to specify more than one OpenPGP key, for example because they are transitioning to a newer stronger key, can do so by adding multiple OPENPGPKEY records. A single OPENPGPKEY DNS record MUST only contain one OpenPGP key.

The type value allocated for the OPENPGPKEY RR type is 61. The OPENPGPKEY RR is class independent.

2.1. The OPENPGPKEY RDATA component

The RDATA portion of an OPENPGPKEY Resource Record contains a single value consisting of a [RFC4880] formatted Transferable Public Key.

2.1.1. The OPENPGPKEY RDATA content

An OpenPGP Transferable Public Key can be arbitrarily large. DNS records are limited in size. When creating OPENPGPKEY DNS records, the OpenPGP Transferable Public Key should be filtered to only contain appropriate and useful data. At a minimum, an OPENPGPKEY Transferable Public Key for the user hugh@example.com should contain:

```
o The primary key X
  o One User ID Y, which SHOULD match 'hugh@example.com'
    o self-signature from X, binding X to Y
```

If the primary key is not encryption-capable, a relevant subkey should be included resulting in an OPENPGPKEY Transferable Public Key containing:

```
o The primary key X
  o One User ID Y, which SHOULD match 'hugh@example.com'
    o self-signature from X, binding X to Y
  o encryption-capable subkey Z
    o self-signature from X, binding Z to X
  o [ other subkeys if relevant ... ]
```

The user can also elect to add a few third-party certifications which they believe would be helpful for validation in the traditional Web Of Trust. The resulting OPENPGPKEY Transferable Public Key would then look like:

```
o The primary key X
  o One User ID Y, which SHOULD match 'hugh@example.com'
```

```
            o self-signature from X, binding X to Y
            o third-party certification from V, binding Y to X
            o [ other third-party certifications if relevant ... ]
          o encryption-capable subkey Z
            o self-signature from X, binding Z to X
          o [ other subkeys if relevant ... ]
```

2.1.2. Reducing the Transferable Public Key size

When preparing a Transferable Public Key for a specific OPENPGPKEY RDATA format with the goal of minimizing certificate size, a user would typically want to:

- Where one User ID from the certifications matches the looked-up address, strip away non-matching User IDs and any associated certifications (self-signatures or third-party certifications).

- Strip away all User Attribute packets and associated certifications.

- Strip away all expired subkeys. The user may want to keep revoked subkeys if these were revoked prior to their preferred expiration time to ensure that correspondents know about these earlier than expected revocations.

- Strip away all but the most recent self-signature for the remaining user IDs and subkeys.

- Optionally strip away any uninteresting or unimportant third-party User ID certifications. This is a value judgment by the user that is difficult to automate. At the very least, expired and superseded third-party certifcations should be stripped out. The user should attempt to keep the most recent and most well connected certifications in the Web Of Trust in their Transferable Public Key.

2.2. The OPENPGPKEY RDATA wire format

The RDATA Wire Format consists of a single OpenPGP Transferable Public Key as defined in Section 11.1 of [RFC4880]. Note that this format is without ASCII armor or base64 encoding.

2.3. The OPENPGPKEY RDATA presentation format

The RDATA Presentation Format, as visible in master files [RFC1035], consists of a single OpenPGP Transferable Public Key as defined in Section 11.1 of [RFC4880] encoded in base64 as defined in Section 4 of [RFC4648].

3. Location of the OPENPGPKEY record

The DNS does not allow the use of all characters that are supported in the "local-part" of email addresses as defined in [RFC5322] and [RFC6530]. Therefore, email addresses are mapped into DNS using the following method:

- o The user name (the "left-hand side" of the email address, called the "local-part" in the mail message format definition [RFC5322] and the local-part in the specification for internationalized email [RFC6530]) is encoded in UTF-8 (or its subset ASCII). If the local-part is written in another encoding it MUST be converted to UTF-8.

- o The local-part is hashed using the SHA2-256 [RFC5754] algorithm, with the hash truncated to 28 octets and represented in its hexadecimal representation, to become the left-most label in the prepared domain name.

- o The string "_openpgpkey" becomes the second left-most label in the prepared domain name.

- o The domain name (the "right-hand side" of the email address, called the "domain" in [RFC5322]) is appended to the result of step 2 to complete the prepared domain name.

For example, to request an OPENPGPKEY resource record for a user whose email address is "hugh@example.com", an OPENPGPKEY query would be placed for the following QNAME: "c93f1e400f26708f98cb19d936620da35eec8f72e57f9eec01c1afd6._openpgpkey.example.com". The corresponding RR in the example.com zone might look like (key shortened for formatting):

```
c9[..]d6._openpgpkey.example.com. IN OPENPGPKEY <base64 public key>
```

4. Email address variants and internationalization considerations

Mail systems usually handle variant forms of local-parts. The most common variants are upper and lower case, often automatically corrected when a name is recognized as such. Other variants include systems that ignore "noise" characters such as dots, so that local parts johnsmith and John.Smith would be equivalent. Many systems allow "extensions" such as john-ext or mary+ext where john or mary is treated as the effective local-part, and the ext is passed to the recipient for further handling. This can complicate finding the OPENPGPKEY record associated with the dynamically created email address.

[RFC5321] and its predecessors have always made it clear that only the recipient MTA is allowed to interpret the local-part of an address. A client supporting OPENPGPKEY therefore MUST NOT perform any kind of mapping rules based on the email address.

Section 3 above defines how the local-part is used to determine the location in which one looks for an OPENPGPKEY record. Given the variety of local-parts seen in email, designing a good experiment for this is difficult as: a) some current implementations are known to lowercase at least US-ASCII local-parts, b) we know from (many) other situations that any strategy based on guessing and making multiple DNS queries is not going to achieve consensus for good reasons, and c) the underlying issues are just hard - see Section 10.1 of [RFC6530] for discussion of just some of the issues that would need to be tackled to fully address this problem.

However, while this specification is not the place to try to address these issues with local-parts, doing so is also not required to determine the outcome of this experiment. If this experiment succeeds then further work on email addresses with non-ASCII local-parts will be needed and that would be better based on the findings from this experiment, rather than doing nothing or starting this experiment based on a speculative approach to what is a very complex topic.

5. Application use of OPENPGPKEY

The OPENPGPKEY record allows an application or service to obtain an OpenPGP public key and use it for verifying a digital signature or encrypting a message to the public key. The DNS answer MUST pass DNSSEC validation; if DNSSEC validation reaches any state other than "Secure" (as specified in [RFC4035]), the DNSSEC validation MUST be treated as a failure.

5.1. Obtaining an OpenPGP key for a specific email address

If no OpenPGP public keys are known for an email address, an OPENPGPKEY DNS lookup MAY be performed to seek the OpenPGP public key that corresponds to that email address. This public key can then be used to verify a received signed message or can be used to send out an encrypted email message. An application whose attempt fails to retrieve a DNSSEC verified OPENPGPKEY RR from the DNS should remember that failure for some time to avoid sending out a DNS request for each email message the application is sending out; such DNS requests constitute a privacy leak

5.2. Confirming that an OpenPGP key is current

Locally stored OpenPGP public keys are not automatically refreshed. If the owner of that key creates a new OpenPGP public key, that owner is unable to securely notify all users and applications that have its old OpenPGP public key. Applications and users can perform an OPENPGPKEY lookup to confirm the locally stored OpenPGP public key is still the correct key to use. If the locally stored OpenPGP public key is different from the DNSSEC validated OpenPGP public key currently published in DNS, the confirmation MUST be treated as a failure unless the locally stored OpenPGP key signed the newly published OpenPGP public key found in DNS. An application that can interact with the user MAY ask the user for guidance, otherwise the application will have to apply local policy. For privacy reasons, an application MUST NOT attempt to lookup an OpenPGP key from DNSSEC at every use of that key.

5.3. Public Key UIDs and query names

An OpenPGP public key can be associated with multiple email addresses by specifying multiple key uids. The OpenPGP public key obtained from a OPENPGPKEY RR can be used as long as the query and resulting data form a proper email to uid identity association.

CNAME's (see [RFC2181]) and DNAME's (see [RFC6672]) can be followed to obtain an OPENPGPKEY RR, as long as the original recipient's email address appears as one of the OpenPGP public key uids. For example, if the OPENPGPKEY RR query for hugh@example.com (8d57[...]b7._openpgpkey.example.com) yields a CNAME to 8d57[...]b7._openpgpkey.example.net, and an OPENPGPKEY RR for 8d57[...]b7._openpgpkey.example.net exists, then this OpenPGP public key can be used, provided one of the key uids contains "hugh@example.com". This public key cannot be used if it would only contain the key uid "hugh@example.net".

If one of the OpenPGP key uids contains only a single wildcard as the LHS of the email address, such as "*@example.com", the OpenPGP public key may be used for any email address within that domain. Wildcards at other locations (eg hugh@*.com) or regular expressions in key uids are not allowed, and any OPENPGPKEY RR containing these MUST be ignored.

6. OpenPGP Key size and DNS

Due to the expected size of the OPENPGPKEY record, applications SHOULD use TCP - not UDP - to perform queries for the OPENPGPKEY Resource Record.

Although the reliability of the transport of large DNS Resource Records has improved in the last years, it is still recommended to keep the DNS records as small as possible without sacrificing the security properties of the public key. The algorithm type and key size of OpenPGP keys should not be modified to accommodate this section.

OpenPGP supports various attributes that do not contribute to the security of a key, such as an embedded image file. It is recommended that these properties not be exported to OpenPGP public keyrings that are used to create OPENPGPKEY Resource Records. Some OpenPGP software, for example GnuPG, support a "minimal key export" that is well suited to use as OPENPGPKEY RDATA. See Appendix A.

7. Security Considerations

DNSSEC is not an alternative for the "web of trust" or for manual fingerprint verification by users. DANE for OpenPGP as specified in this document is a solution aimed to ease obtaining someone's public key. Without manual verification of the OpenPGP key obtained via DANE, this retrieved key should only be used for encryption if the only other alternative is sending the message in plaintext. While this thwarts all passive attacks that simply capture and log all plaintext email content, it is not a security measure against active attacks. A user who publishes an OPENPGPKEY record in DNS still expects senders to perform their due diligence by additional (non-DNSSEC) verification of their public key via other out-of-band methods before sending any confidential or sensitive information.

In other words, the OPENPGPKEY record MUST NOT be used to send sensitive information without additional verification or confirmation that the OpenPGP key actually belongs to the target recipient.

Various components could be responsible for encrypting an email message to a target recipient. It could be done by the sender's

email client or software plugin, the sender's Mail User Agent (MUA) or the sender's Mail Transfer Agent (MTA). Each of these have their own characteristics. An email client can ask the user to make a decision before continuing. The MUA can either accept or refuse a message. The MTA must deliver the message as-is, or encrypt the message before delivering. Each of these programs should attempt to encrypt an unencrypted received message whenever possible.

In theory, two different local-parts could hash to the same value. This document assumes that such a hash collision has a negliable chance of happening.

Organisations that are required to be able to read everyone's encrypted email should publish the escrow key as the OPENPGPKEY record. Mail servers of such organizations MAY optionally re-encrypt the message to the individual's OpenPGP key.

7.1. MTA behaviour

An MTA could be operating in a stand-alone mode, without access to the sender's OpenPGP public keyring, or in a way where it can access the user's OpenPGP public keyring. Regardless, the MTA MUST NOT modify the user's OpenPGP keyring.

An MTA sending an email MUST NOT add the public key obtained from an OPENPGPKEY resource record to a permanent public keyring for future use beyond the TTL.

If the obtained public key is revoked, the MTA MUST NOT use the key for encryption, even if that would result in sending the message in plaintext.

If a message is already encrypted, the MTA SHOULD NOT re-encrypt the message, even if different encryption schemes or different encryption keys would be used.

If the DNS request for an OPENPGPKEY record returned an Indeterminate or Bogus answer as specified in [RFC4035], the MTA MUST NOT send the message and queue the plaintext message for encrypted delivery at a later time. If the problem persists, the email should be returned via the regular bounce methods.

If multiple non-revoked OPENPGPKEY resource records are found, the MTA SHOULD pick the most secure RR based on its local policy.

7.2. MUA behaviour

If the public key for a recipient obtained from the locally stored sender's public keyring differs from the recipient's OPENPGPKEY RR, the MUA MUST NOT accept the message for delivery.

If the public key for a recipient obtained from the locally stored sender's public keyring contains contradicting properties for the same key obtained from an OPENPGPKEY RR, the MUA SHOULD NOT accept the message for delivery.

If multiple non-revoked OPENPGPKEY resource records are found, the MUA SHOULD pick the most secure OpenPGP public key based on its local policy.

7.3. Email client behaviour

Email clients should adhere to the above listed MUA behaviour. Additionally, an email client MAY interact with the user to resolve any conflicts between locally stored keyrings and OPENPGPKEY RRdata.

An email client that is encrypting a message SHOULD clearly indicate to the user the difference between encrypting to a locally stored and user verified public key and encrypting to an unverified public key obtained via an OPENPGPKEY resource record.

7.4. Response size

To prevent amplification attacks, an Authoritative DNS server MAY wish to prevent returning OPENPGPKEY records over UDP unless the source IP address has been confirmed with [EDNS-COOKIE]. Such servers MUST NOT return REFUSED, but answer the query with an empty Answer Section and the truncation flag set ("TC=1").

7.5. Email address information leak

The hashing of the user name in this document is not a security feature. Publishing OPENPGPKEY records however, will create a list of hashes of valid email addresses, which could simplify obtaining a list of valid email addresses for a particular domain. It is desirable to not ease the harvesting of email addresses where possible.

The domain name part of the email address is not used as part of the hash so that hashes can be used in multiple zones deployed using DNAME [RFC6672]. This does makes it slightly easier and cheaper to brute-force the SHA2-256 hashes into common and short user names, as single rainbow tables can be re-used across domains. This can be somewhat countered by using NSEC3.

DNS zones that are signed with DNSSEC using NSEC for denial of existence are susceptible to zone-walking, a mechanism that allows someone to enumerate all the OPENPGPKEY hashes in a zone. This can be used in combination with previously hashed common or short user names (in rainbow tables) to deduce valid email addresses. DNSSEC-signed zones using NSEC3 for denial of existence instead of NSEC are significantly harder to brute-force after performing a zone-walk.

7.6. Storage of OPENPGPKEY data

Users may have a local key store with OpenPGP public keys. An application supporting the use of OPENPGPKEY DNS records MUST NOT modify the local key store without explicit confirmation of the user, as the application is unaware of the user's personal policy for adding, removing or updating their local key store. An application MAY warn the user if an OPENPGPKEY record does not match the OpenPGP public key in the local key store.

Applications that cannot interact with users, such as daemon processes, SHOULD store OpenPGP public keys obtained via OPENPGPKEY up to their DNS TTL value. This avoids repeated DNS lookups that third parties could monitor to determine when an email is being sent to a particular user.

7.7. Security of OpenPGP versus DNSSEC

Anyone who can obtain a DNSSEC private key of a domain name via coercion, theft or brute force calculations, can replace any OPENPGPKEY record in that zone and all of the delegated child zones. Any future messages encrypted with the malicious OpenPGP key could then be read.

Therefore, an OpenPGP key obtained via a DNSSEC validated OPENPGPKEY record can only be trusted as much as the DNS domain can be trusted, and is no substitute for in-person OpenPGP key verification or additional Openpgp verification via "Web Of Trust" signatures present on the OpenPGP in question.

8. Implementation Status

[RFC Editor Note: Please remove this entire seciton prior to publication as an RFC.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to

RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist. According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit."

8.1. The GNU Privacy Guard (GNUpg)

Implementation Name and Details: The GNUpg software, more commonly known as "gpg", is is available at https://gnupg.org/

Brief Description: Support has been added to gnupg in their git repository. This code is expected to be part of the next official release.

Level of Maturity: The implementation has just been added and has not seen widespread deployment.

Coverage: The implementation follows the latest draft with the exception that it first performs a lowercase of the local-part before hashing. This is done because other parts in the code that perform a lookup of uid already performed a localcasing to ensure case insensitivity. The implementors are tracking the development of this draft in particular with respect to the lowercase issue.

Licensing: All code is covered under the GNU Public License version 3 or later.

Implementation Experience: Currrent experience limited to small test networks only

Contact Information: https://gnupg.org/

Interoperability: No report.

8.2. hash-slinger

Implementation Name and Details: The hash-slinger software is a collection of tools to generate, download and verify application public keys and application fingerprints. It uses DNSSEC validation. The tool is written by the author of this document. It is available at http://people.redhat.com/pwouters/

Brief Description: Support has been added in the form of an "openpgpkey" command that can generate, fetch, validate the DNSSEC authentication and verify OPENPGPKEY records.

Level of Maturity: The implementation has been around for a few months but has not seen widespread deployment.

Coverage: The implementation follows the latest draft with the exception that it first performs a lowercase of the local-part before hashing.

Licensing: All code is covered under the GNU Public License version 3 or later.

Implementation Experience: Currrent experience limited to small test networks only

Contact Information: pwouters@redhat.com

Interoperability: No report.

8.3. openpgpkey-milter

Implementation Name and Details: The openpgpkey-milter is a Postfix and Sendmail Mail server plugin (milter) that automatically encrypts email before sending further to other SMTP servers. It is written by the author of this document. It is available at http://github.com/letoams/openpgpkey-milter/

Brief Description: Before forwarding an unencrypted email, the plugin looks for the presence of an OPENPGPKEY record. When available, it will encrypt the email message and send out the encrypted email.

Level of Maturity: The implementation has been around for a few months but has not seen widespread deployment.

Coverage: The implementation follows the latest draft with the exception that it first performs a lowercase of the local-part before hashing.

Licensing: All code is covered under the GNU Public License version 3 or later.

Implementation Experience: Currrent experience limited to small test networks only

Contact Information: pwouters@redhat.com

Interoperability: No report.

9. IANA Considerations

9.1. OPENPGPKEY RRtype

This document uses a new DNS RR type, OPENPGPKEY, whose value 61 has been allocated by IANA from the Resource Record (RR) TYPEs subregistry of the Domain Name System (DNS) Parameters registry.

10. Acknowledgments

This document is based on RFC-4255 and draft-ietf-dane-smime whose authors are Paul Hoffman, Jacob Schlyter and W. Griffin. Olafur Gudmundsson provided feedback and suggested various improvements. Willem Toorop contributed the gpg and hexdump command options. Daniel Kahn Gillmor provided the text describing the OpenPGP packet formats and filtering options. Edwin Taylor contributed language improvements for various iterations of this document. Text regarding email mappings was taken from draft-levine-dns-mailbox whose author is John Levine.

11. References

11.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <http://www.rfc-editor.org/info/rfc2181>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

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

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, DOI 10.17487/RFC5754, January 2010, <http://www.rfc-editor.org/info/rfc5754>.

11.2. Informative References

[EDNS-COOKIE] Eastlake, Donald., "Domain Name System (DNS) Cookies", draft-ietf-dnsop-cookies (work in progress), August 2015.

[HKP] Shaw, D., "The OpenPGP HTTP Keyserver Protocol (HKP)", draft-shaw-openpgp-hkp (work in progress), March 2013.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <http://www.rfc-editor.org/info/rfc3597>.

[RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, DOI 10.17487/RFC4398, March 2006, <http://www.rfc-editor.org/info/rfc4398>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <http://www.rfc-editor.org/info/rfc5321>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <http://www.rfc-editor.org/info/rfc6530>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <http://www.rfc-editor.org/info/rfc6672>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <http://www.rfc-editor.org/info/rfc6982>.

Appendix A. Generating OPENPGPKEY records

The commonly available GnuPG software can be used to generate a minimum Transferable Public Key for the RRdata portion of an OPENPGPKEY record:

```
gpg --export --export-options export-minimal,no-export-attributes \
    hugh@example.com | base64
```

The --armor or -a option of the gpg command should NOT be used, as it adds additional markers around the armored key.

When DNS software reading or signing the zone file does not yet support the OPENPGPKEY RRtype, the Generic Record Syntax of [RFC3597] can be used to generate the RDATA. One needs to calculate the number of octets and the actual data in hexadecimal:

```
gpg --export --export-options export-minimal,no-export-attributes \
    hugh@example.com | wc -c

gpg --export --export-options export-minimal,no-export-attributes \
    hugh@example.com | hexdump -e \
```

```
          '"\t" /1 "%.2x"' -e '/32 "\n"'
```

These values can then be used to generate a generic record (line break has been added for formatting):

```
<SHA2-256-trunc(hugh)>._openpgpkey.example.com. IN TYPE61 \# \
    <numOctets> <keydata in hex>
```

The openpgpkey command in the hash-slinger software can be used to generate complete OPENPGPKEY records

```
~> openpgpkey --output rfc hugh@example.com
c9[..]d6._openpgpkey.example.com. IN OPENPGPKEY mQCNAzIG[...]

~> openpgpkey --output generic hugh@example.com
c9[..]d6._openpgpkey.example.com. IN TYPE61 \# 2313 99008d03[...]
```

Author's Address

Paul Wouters
Red Hat

Email: pwouters@redhat.com