Workgroup: Network Working Group
Internet-Draft:
draft-koch-openpgp-webkey-service-14
Published: 13 May 2022
Intended Status: Informational
Expires: 14 November 2022
Authors: W. Koch
GnuPG e.V.

# OpenPGP Web Key Directory

## Abstract

This specification describes a service to locate OpenPGP keys by mail address using a Web service and the HTTPS protocol. It also provides a method for secure communication between the key owner and the mail provider to publish and revoke the public key.

## Status of This Memo

## Copyright Notice

## Table of Contents

## 1. Introduction

This memo describes a method to associate OpenPGP keys with a mail address and how to look them up using a web service with a well-known URI. In addition a mail based protocol is given to allow a client to setup such an association and to maintain it.

## 2. Notational Conventions

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. Web Key Directory

A major use case for OpenPGP is the encryption of mail. A common difficulty of sending encrypted mails to a new communication partner is to find the appropriate public key of the recipient. Unless an off-channel key exchange has been done, there are no easy ways to discover the required key. The common practice is to search the network of public key servers for a key matching the recipient's mail address. This practise bears the problem that the keyservers are not able to give a positive confirmation that a key actually belongs to the mail addresses given in the key. Further, there are often several keys matching a mail address and thus one needs to pick a key on good luck. This is clearly not a secure way to setup an end-to-end encryption. Even if the need for a trusted key for an initial mail message is relinquished, a non-authenticated key may be a wrong one and the actual recipient would receive a mail which she can't decrypt, due to the use of a wrong key.

Methods to overcome this problem are

* sending an initial unencrypted message with the public key attached,

* using the OpenPGP DANE protocol to lookup the recipients key via the DNS.

The first method has the obvious problems of not even trying to encrypt the initial mail, an extra mail round-trip, and problems with unattended key discovery.

The latter method works fine but requires that mail providers need to set up a separate DNS resolver to provide the key. The administration of a DNS zone is often not in the hands of small mail installations. Thus an update of the DNS resource records needs to be delegated to the ISP running the DNS service. Further, DNS lookups are not encrypted and missing all confidentially. Even if the participating MUAs are using STARTTLS to encrypt the mail exchange, a DNS lookup for the key unnecessarily identifies the local-part of the recipients mail address to any passive eavesdroppers.

This memo specified a new method for key discovery using an encrypted https connection.

## 3.1. Key Discovery

Although URIs are able to encode all kind of characters, straightforward implementations of a key directory may want to store the local-part of a mail address directly in the file system. This forbids the use of certain characters in the local-part. To allow for such an implementation method the URI uses an encoded form of the local-part which can be directly mapped to a file name.

OpenPGP defines its User IDs, and thus the mail address, as UTF-8 strings. To help with the common pattern of using capitalized names (e.g. "Joe.Doe@example.org") for mail addresses, and under the premise that almost all MTAs treat the local-part case-insensitive and that the domain-part is required to be compared case-insensitive anyway, all upper-case ASCII characters in a User ID are mapped to lowercase. Non-ASCII characters are not changed.

The so mapped local-part is hashed using the SHA-1 algorithm. The resulting 160 bit digest is encoded using the Z-Base-32 method as described in [RFC6189], section 5.1.6. The resulting string has a fixed length of 32 octets.

There are two variants on how to form the request URI: The advanced and the direct method. Implementations MUST first try the advanced method. Only if an address for the required sub-domain does not exist, they SHOULD fall back to the direct method. A non-responding server does not mean that the fall back should be carried out.

The advanced method requires that a sub-domain with the fixed name openpgpkey is created and queried. The URI is constructed by concatenating these items:

- *The scheme https://,
- *the string openpgpkey,
- *the domain-part,
- *the string /.well-known/openpgpkey/,
- *the domain-part in lowercase,
- *the string /hu/,
- *the above constructed 32 octet string,
- *the unchanged local-part as a parameter with name l using proper percent escaping.

An example for such an advanced method URI to lookup the key for Joe.Doe@Example.ORG is:

```
https://openpgpkey.example.org/.well-known/openpgpkey/
example.org/hu/iy9q119eutrkn8s1mk4r39qejnbu3n5q?l=Joe.Doe
```

(line has been wrapped for rendering purposes)

The direct method requires no additional DNS entries and constructs the URI by concatenating these items:

- *The scheme https://,
- *the domain-part,
- *the string /.well-known/openpgpkey/hu/,
- *the above constructed 32 octet string,

* the unchanged local-part as a parameter with name l using proper percent escaping.

Example for a direct method URI:

```
https://example.org/.well-known/openpgpkey/
hu/iy9q119eutrkn8s1mk4r39qejnbu3n5q?l=Joe.Doe
```

(line has been wrapped for rendering purposes)

Sites which do not use the advanced method but employ wildcard DNS for their sub-domains MUST make sure that the openpgpkey sub-domain is not subject to the wildcarding. This can be done by inserting an empty TXT RR for this sub-domain.

The HTTP GET method MUST return the binary representation of the OpenPGP key for the given mail address. The key needs to carry a User ID packet ([RFC4880]) with that mail address. Note that the key may be revoked or expired - it is up to the client to handle such conditions. To ease distribution of revoked keys, a server may return revoked keys in addition to a new key. The keys are returned by a single request as concatenated key blocks.

The server MUST accept the HTTP HEAD method to allow a client to check for the existence of a key.

The server SHOULD use "application/octet-stream" as the Content-Type for the data but clients SHOULD also accept any other Content-Type. The server MUST NOT return an ASCII armored version of the key.

The server MUST serve a Policy Flags file as specified below. That file is even required if the Web Key Directory Update Protocol is not supported.

The benefit of the advanced method is its greater flexibility in setting up the Web Key Directory in environments where more than one mail domain is hosted. DNS SRV resource records, as used in earlier specifications of this protocol, posed a problem for implementations which have only limited access to DNS resolvers. The direct method is kept for backward compatibility and to allow providing a Web Key Directory even with without DNS change requirements.

## 4. Web Key Directory Update Protocol

To put keys into the key directory a protocol to automate the task is desirable. The protocol defined here is entirely based on mail and the assumption that a mail provider can securely deliver mail to the INBOX of a user (e.g. an IMAP folder). Note that the same protocol may also be used for submitting keys for use with OpenPGP DANE.

In the following sections the term "target key" denotes the to be published key, the term "submission key" the key associated with the submission-address of the mail provider. The string "WELLKNOWN" denotes the first part of an URI specific for a domain. In the examples the domain "example.org" is assumed, thus:

```
WELLKNOWN := https://openpgpkey.example.org/.well-known/
             openpgpkey/example.org
```

(line has been wrapped for rendering purposes)

or if the sub-domain openpgpkey does not exist (direct method):

```
WELLKNOWN := https://example.org/.well-known/openpgpkey
```

We assume that the user already created a key for her mail account alice@example.org. To install the key at her provider's Web Key Directory, she performs the following steps:

1. She retrieves a file which contains one line with the mail address used to submit the key to the mail provider. See below for the syntax of that file. For a mail address at the domain "example.org" the URI of the file is

   ```
   WELLKNOWN/submission-address
   ```

2. She sends her key using SMTP (or any other transport mechanism) to the provider using the submission address and key format as specified by PGP/MIME.

3. The provider checks that the received key has a User ID which matches an account name of the provider.

4. The provider sends an encrypted message containing a nonce and the fingerprint of the key to the mail account of the user. Note that a similar scheme is used by the well known caff(1) tool to help with key signing parties.

5. A legitimate user will be able to decrypt the message because she created the key and is in charge of the private key. This step verifies that the submitted key has actually been created by the owner of the account.

6. The user sends the decrypted nonce back to the submission address as a confirmation that the private key is owned by her and that the provider may now publish the key. The confirmation mail to the provider MUST be encrypted using the provider's public key as retrieved using the key lookup protocol described above.

7. The provider receives the nonce, matches it with its database of pending confirmations and then publishes the key. Finally the provider sends a mail back to the user to notify her of the publication of her key.

The message data structures used for the above protocol are specified in detail below.

## 4.1. The Submission Address

The address of the submission file is

```
WELLKNOWN/submission-address
```

The file consists of exactly one line, terminated by a LF, or the sequence of CR and LF, with the full mail address to be used for submission of a key to the mail provider. For example the content of the file may be

```
key-submission-example.org@directory.example.org
```

## 4.2. The Submission Mail

The mail used to submit a key to the mail provider MUST comply to the PGP/MIME specification ([RFC3156], section 7), which states that the Content-Type must be "application/pgp-keys", there are no required or optional parameters, and the body part contains the ASCII-armored transferable Public Key Packets as defined in [RFC4880], section 11.1.

The mail provider MUST publish a key capable of signing and encryption for the submission-address in the Web Key Directory or via DANE. The key to be published MUST be submitted using a PGP/MIME encrypted message ([RFC3156], section 4). The message MUST NOT be signed (because the authenticity of the signing key has not yet been confirmed). After decryption of the message at the mail provider a single "application/pgp-keys" part, as specified above, is expected.

## 4.3. The Confirmation Request

The mail provider sends a confirmation mail in response to a received key publication request. The message MUST be sent from the submission-address of the mail provider to the mail address extracted from the target key. The message needs to be a PGP/MIME signed message using the submission key of the provider for the signature. The signed message MUST have two parts:

The first part MUST have "text" as its Content-Type and can be used to explain the purpose of the mail. For example it may point to this specification and explain on how to manually perform the protocol.

The second part MUST have a Content-Type of "application/vnd.gnupg.wkd" and carry an OpenPGP encrypted message in ASCII Armor format. If the protocol version is unknown or less than 5 the Content-Type "application/vnd.gnupg.wks" MUST be used for backward compatibility. The message MUST be encrypted to the target key and MUST NOT be signed. After decryption a text file in the Web Key data format must be yielded.

That data format consists of name-value pairs with one name-value pair per LF or CR+LF terminated line. Empty lines are allowed and will be ignored by the receiver. A colon is used to terminate a name.

In a confirmation request the following names MUST be send in the specified order:

* "type": The value must be "confirmation-request".

* "sender": This is the mailbox the user is expected to sent the confirmation response to. The value must match the mailbox part of the "From:" address of this request. Exactly one address MUST be given.

* "address": The value is the addr-spec part of the target key's mail address. The value SHOULD match the addr-spec part of the recipient's address. The value MUST be UTF-8 encoded as required for an OpenPGP User ID.

* "fingerprint": The value is the fingerprint of the target key. The fingerprint is given in uppercase hex encoding without any interleaving spaces.

* "nonce": The value is a string with a minimum length of 16 octets and a maximum length of 64 octets. The string must entirely be made up of random ASCII letters or digits. This nonce will be sent back to the mail provider as proof that the recipient is the legitimate owner of the target-key.

The receiver of that message is expected to verify the outer signature and disregard the entire message if it can't be verified or has not been signed by the key associated with the submission address.

After the message has been verified the receiver decrypts the second part of the signed message, checks that the "fingerprint" matches the target key, checks that the "address" matches a User ID of the target key, and checks the other constrains of the request format. If any constraint is not asserted, or the fingerprint or User ID do not match the target key, or there is no pending publication requests (i.e. a mail recently sent to the submission address), the user MAY be notified about this fake confirmation attempt.

In other cases the confirmation request is legitimate and the MUA shall silently send a response as described in the next section.

The rationale for the outer signature used with this request is to allow early detection of spam mails. This can be done prior to the decryption step and avoids asking the user to enter a passphrase to perform the decryption for a non-legitimate message. The use of a simple encrypted attachment, instead of using PGP/MIME encryption, is to convey the Content-Type of that attachment in the clear and also to prevent automatic decryption of that attachment by PGP/MIME aware clients. The MUA may in fact detect this confirmation request and present a customized dialog for confirming that request.

## 4.4. The Confirmation Response

A response to a confirmation request MUST only be send in the positive case; there is no negative confirmation response. A mail service provider is expected to cancel a pending key submission after a suitable time without a confirmation. The mail service provider SHOULD NOT retry the sending of a confirmation request after the first request has been send successfully.

The user MUST send the confirmation response from her target mail address to the "from" address of the confirmation request. The message MUST be signed and encrypted using the PGP/MIME Combined format ([RFC3156], section 6.2). The signing key is the target key and the encryption key is the key associated with the provider's submission address.

The Content-Type used for the plaintext message MUST match the Content-Type of the request. The format is the same as described above for the Confirmation Request. The body must contain four name-value pairs in this order:

* "type": The value must be "confirmation-response".

* "sender": The value is the value of the "sender" parameter from the confirmation request.

* "address": The value is the value of the "address" parameter from the confirmation request.

* "nonce": The value is the value of the "nonce" parameter from the confirmation request.

## 4.5. Policy Flags

For key generation and submission it is useful to tell the client about certain properties of the mail provider in advance. This can be done with a file at the URL

```
WELLKNOWN/policy
```

A site supporting the Web Key Directory MUST serve this file; it is sufficient if that file has a zero length. Clients may use this file to check for Web Key Directory support.

The file contains keywords and optionally values, one per line with each line terminated by a LF or the sequence of CR and LF. Empty lines and lines starting with a "#" character are considered comment lines. A keyword is made up of lowercase letters, digits, hyphens, or dots. An underscore is allowed as a name space delimiters; see below. The first character must be a letter. Keywords which are defined to require a value are directly followed by a colon and then after optional white space the value. Clients MUST use case-insensitive matching for the keyword.

Currently defined keywords are:

* "mailbox-only": The mail server provider does only accept keys with only a mailbox in the User ID. In particular User IDs with a real name in addition to the mailbox will be rejected as invalid.

* "dane-only": The mail server provider does not run a Web Key Directory but only an OpenPGP DANE service. The Web Key Directory Update protocol is used to update the keys for the DANE service.

* "auth-submit": The submission of the mail to the server is done using an authenticated connection. Thus the submitted key will be published immediately without any confirmation request.

* "protocol-version": This keyword can be used to explicitly claim the support of a specific version of the Web Key Directory update protocol. This is in general not needed but implementations may have workarounds for providers which only support an old protocol version. If these providers update to a newer version they should add this keyword so that the implementation can disable the workaround. The value is an integer corresponding to the respective draft revision number.

* "submission-address": An alternative way to specify the submission address. The value is the addr-spec part of the address to send requests to this server. If this keyword is used in addition to the submission-address file, both MUST have the same value.

More keywords will be defined in updates to this I-D. There is no registry except for this document. For experimental use of new features or for provider specific settings, keywords MUST be prefixed with a domain name and an underscore.

## 5. Security Considerations

The use of SHA-1 for the mapping of the local-part to a fixed string is not a security feature but merely used to map the local-part to a fixed-sized string made from a well defined set of characters. It is not intended to conceal information about a mail address.

The domain name part of the mail address is not part of the hash to avoid problems with internationalized domain names. Instead a separate URL is required for each domain name.

To make it a bit harder to test for published keys, the server responsible to serve the WELLKNOWN directory SHOULD NOT create an index file for that directory or any sub-directory.

The mail provider MUST make sure to publish a key in a way that only the mail address belonging to the requested user is part of the User ID packets included in the returned key. Other User ID packets and their associated binding signatures MUST be removed before publication. Confirmation requests MUST only be send for such to be published User ID. It is further recommended that a client filters a received key or a key send for a publication requests so that only the specific User ID with the mail address of the provider is imported or send.

A client MUST NOT accept a HTTP authentication challenge (HTTP code 401) because the information in the Web Key Directory is public and needs no authentication. Allowing an authentication challenge has the problem to easily confuse a user with a password prompt and tricking him into falsely entering the passphrase used to protect his private key or to login to his mail provider.

The use of DNS SRV records as specified in former revisions of this document reduces the certainty that a mail address belongs to a domain. For example an attacker may change the target to a host in a sub-domain under their control and thus gain full control over all keys.

## 6. IANA Considerations

### 6.1. Well-Known URI

IANA is requested to assign a well-known URI in the "Well-Known URIs" registry as defined by [RFC8615]:

URI suffix: openpgpkey

Change controller: IETF

Specification document: This

## 7. Acknowledgments

The author would like to acknowledge the help of the individuals who kindly voiced their opinions on the GnuPG mailing lists, in particular, the help of Bernhard Reiter and Guilhem Moulin.

## 8. Normative References

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

**[RFC3156]** Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, August 2001, <http://www.rfc-editor.org/rfc/rfc3156.txt>.

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

**[RFC6189]** Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, DOI 10.17487/RFC6189, April 2011, <http://www.rfc-editor.org/info/rfc6189>.

**[RFC8615]** Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

## Appendix A. Sample Protocol Run

The following non-normative example can be used by implementors as guidance.

Note that GnuPG version 2.1.12 supports the key discovery described in version -00 of this document (auto-key-locate method "wkd"). Version 2.1.16 can run the protocol described in this document but is also able to run the protocol version specified by -01. For backward compatibility this example uses the Content-Type as required for versions of this protocol prior to -04; if the client knows that the server support -04 "vnd.gnupg.wkd" should be used.

## A.1. Sample Keys

This is the provider's submission key:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lFgEV/TAohYJKwYBBAHaRw8BAQdAB/k9YQfSTI8qQqqK1KimH/BsvzsowWItSQPT
FP+fOC4AAP46uJ3Snno3Vy+kORye3rf0VvWvuz82voEQLxG6WpfHhREEtBprZXkt
c3VibWlzc2lvbkBleGFtcGxlLm5ldIh5BBMWCAAhBQJX9MCiAhsDBQsJCAcCBhUI
CQoLAgQWAgMBAh4BAheAAAoJEKhtNooW0cqEWMUA/0e9XaeptszWC9ZvPg8INL6a
BvRqPBYGU7PGmuXsxBovAQDyckOykG0UAfHVyN1w4gSK/biMcnqVr857i8/HuvjW
C5xdBFf0wKISCisGAQQBl1UBBQEBB0Apvaoe4MtSEJ1fpds/4DFl2kXXBpnVji/s
Wg9btdthNQMBCAcAAP9FJX99T1LEJzBnvBBnc6bimnT6/1OKM9RdO4R0/uVP6BFL
iGEEGBYIAAkFAlf0wKICGwwACgkQqG02ihbRyoTlGwD9FBr92osjL7HkhhZZ7Z2D
My3b9zpoZeMjvPg5YPqpdKMA/jhZoHuZCRMBYf7YRFb8aXtuyetDFZYrkjnum+OG
HFAD
=Hnwd
-----END PGP PRIVATE KEY BLOCK-----
```

This is the target key to be published:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lFgEV2o9XRYJKwYBBAHaRw8BAQdAZ8zkuQDL9x7rcvvoo6s3iEF1j88Dknd9nZhL
nTEoBRkAAP94nCZMM4WY2IORXfM6phLGSz3RsHvs/vA1Opaus4+R3BKJtBtwYXRy
aWNlLmx1bXVtYmFAZXhhbXBsZS5uZXSIeQQTFggAIQUCV2o9XQIbAwULCQgHAgYV
CAkKCwIEFgIDAQIeAQIXgAAKCRATlWNoKgINCpkNAQDFDcwJUzsxu7aJUiPdpYXj
4uVarrXakxEE8mGFotWhLAD9GH4rqLDYIE3NKEU0s+Okt4tEIwJaV8H1NNPPPMiK
3g2cXQRXaj2NEgorBgEEAZdVAQUBAQdAFnnmZc99TuKk5iCq9wmYZUVF2RcXN2Cs
qAl8iGQQUWsDAQgHAAD/VN/VGmlcwGBPcLTya2hfU4t37nMcFCKdNSXjJ5DFA0AP
PohhBBgWCAAJBQJXaj2NAhsMAAoJEBOVY2gqAg0Ky4UA/0GmVaXzXemLvv1Xw4yx
Eaz/KfKKGc4RJ+38fyqUzw8NAQCohQ+ki3I5f84EXLZEiUiLsnVtOn1HNxvND/gW
TiFZBA==
=GHi7
-----END PGP PRIVATE KEY BLOCK-----
```

## A.2. Sample Messages

The first message triggers the publication requests.

```
From: patrice.lumumba@example.net
To: key-submission@example.net
Subject: Key publishing request
MIME-Version: 1.0
Content-Type: multipart/encrypted;
        protocol="application/pgp-encrypted";
  boundary="=-=01-e8k41e11ob31eefa36wo=-="
Date: Wed, 05 Oct 2016 10:15:51 +0000


--=-=01-e8k41e11ob31eefa36wo=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-e8k41e11ob31eefa36wo=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
```

```
hF4DUgLY5tvmW2sSAQdAR1AcqvFpQe/fHRZbf0xcnl9Tb+AtwaX2yZnZXGELGHsw
1/e3E0JptwM5tpRAVe71ooF8Zq4jl76ZgQKfj/SyjpLJxyoEDy2N5wTQaqW4JtML
0ukB1vh7dIRDxBJX/LQIJC0wz8o1Q3vjcLJKFFvDb7YrerABpPIzwOAupcgIbQHj
5m1+2WU5CL8ffyJy2h1jV2X4OnvWF1Sn6J6SVD6DfZpOPRt9TxSemJrN1LJ3lG0N
ts8AuYmCOeC1H2r5TYyxqkC98JF8+Nvyxd/fwne8IOjK9uixkNMC5H9/ZOH0YWCb
wBnNB4iXuym4OIPxiLkDymsVF0ww/XrODE9Y259EGmO45VFNrJAX3HFs9/PcMCVk
n2qMyEkr8LHiXeEPun6Z54RHUPYv2cUkEZ0hhSJ+rtBxkc/5D/cAScCEXRKFSKEF
jLJAvLK/u/ga5DAzVai+vh6b6Bq+YVPaD9GWMhWj4CgR90p9LULi6S/Hzwhv9Wzf
8fJoJOaDjyvRDgr09jYLWamxkS9NWxqwy6MXJvxwbNdd5XtqiW4Y4o0Ll1hDJhxR
ljn/XvotXKwhKN+4QGhIXDVt4Dl4XxS5ptWfVTau8W8DYqDsU2obEcfsirZv53M1
Q9FCD8CD9+dkBt8VAJekCWVhEltcRHxlrznbk2jxm93xSD2o6gZ5X0VSaSUXyEhm
J+8F3gyTHGgbq/TgyjFoockWh5EtGgAFuWvmPJCF5PO/UaNeoKwgwSJBu6oTXkHx
R4nvvMRcj5UgTsKpZ79NiDQukbjG5ScNT5TCUiiZsBXBqBx3fD61EH6cAuh4P3Kr
iM7PY4fwAHo890Dx+Qlt
=WIhx
```

```
-----END PGP MESSAGE-----

--=-=01-e8k41e11ob31eefa36wo=-=--
```

The server decrypts this message to

```
Content-Type: application/pgp-keys

-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEV2o9XRYJKwYBBAHaRw8BAQdAZ8zkuQDL9x7rcvvoo6s3iEF1j88Dknd9nZhL
nTEoBRm0G3BhdHJpY2UubHVtdW1iYUBleGFtcGxlLm5ldIh5BBMWCAAhBQJXaj1d
AhsDBQsJCAcCBhUICQoLAgQWAgMBAh4BAheAAAoJEBOVY2gqAg0KmQ0BAMUNzAlT
OzG7tolSI92lhePi5VqutdqTEQTyYYWi1aEsAP0YfiuosNggTc0oRTSz46S3i0Qj
AlpXwfU00888yIreDbg4BFdqPY0SCisGAQQBl1UBBQEBB0AWeeZlz31O4qTmIKr3
CZhlRUXZFxc3YKyoCXyIZBBRawMBCAeIYQQYFggACQUCV2o9jQIbDAAKCRATlWNo
KgINCsuFAP9BplWl813pi779V8OMsRGs/ynyihnOESft/H8qlM8PDQEAqIUPpIty
OX/OBFy2RIlIi7J1bTp9RzcbzQ/4Fk4hWQQ=
=qRfF
-----END PGP PUBLIC KEY BLOCK-----
```

and returns this confirmation request

```
From: key-submission@example.net
To: patrice.lumumba@example.net
Subject: Confirm your key publication
MIME-Version: 1.0
Content-Type: multipart/encrypted;
  protocol="application/pgp-encrypted";
  boundary="=-=01-wrzqued738dfx4x97u7y=-="
Date: Wed, 05 Oct 2016 10:16:57 +0000


--=-=01-wrzqued738dfx4x97u7y=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-wrzqued738dfx4x97u7y=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DkYWHjk/NdMASAQdAluQeqhECpU2T0zEyBAEbFzhLkpubN160wjkFCrtUc0Mw
FwYgM2fp9cvTMdJ/xjkvmAcIEOT4AY/hn1yFQ4z0KG0gCkSac+8mkDylnPdxlXYw
0sBSAXlbqpVA7eUpFuU2Zs10zbIXxlwe6osR5wUIJut/RCOsYQmfvxC55x8mUX5/
zgTnNzlMzye5ws4pTgAeQm2x0Yv018L8IZgY5KxwJLBzlss0wLZ45ZcS80hR11Fx
NCow1fKF8lMnOJxagTEOih807nctz8vT5bR1gx0d7N3LM+th8nAg9/6Ghf1XTpLo
MzwGW0FtOG7Dg1Uxbw2bjaOuRBeh6IIpmNAw1pmIfnNu7PpoRydU5w1K/R8MT06z
MKdJ7IW5mVGes9EGnG3e4mjuILvNaZhfYy+a73IhDSaPm3oqdl1Qx7tbNg6lGjn6
KStCYAcPGPp3m7aWkfsPGThOVRhEXqaFFywfwSVEj1pdIRjDFA==
=Cdjh
-----END PGP MESSAGE-----

--=-=01-wrzqued738dfx4x97u7y=-=--
```

The client decrypts this PGP/MIME message as

```
Content-Type: application/vnd.gnupg.wks
Content-Transfer-Encoding: 8bit

type: confirmation-request
sender: key-submission@example.net
address: patrice.lumumba@example.net
fingerprint: B21DEAB4F875FB3DA42F1D1D139563682A020D0A
nonce: f5pscz57zj6fk11wekk8gx4cmrb659a7
```

creates this response

```
Content-Type: application/vnd.gnupg.wks
Content-Transfer-Encoding: 8bit

type: confirmation-response
sender: key-submission@example.net
address: patrice.lumumba@example.net
nonce: f5pscz57zj6fk11wekk8gx4cmrb659a7
```

and sends it PGP/MIME Combined signed and encrypted to the server

```
From: patrice.lumumba@example.net
To: key-submission@example.net
Subject: Key publication confirmation
MIME-Version: 1.0
Content-Type: multipart/encrypted;
  protocol="application/pgp-encrypted";
  boundary="=-=01-iacqg4og4pqz11a5cg1o=-="
Date: Wed, 05 Oct 2016 10:18:52 +0000


--=-=01-iacqg4og4pqz11a5cg1o=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-iacqg4og4pqz11a5cg1o=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DUgLY5tvmW2sSAQdAlq98ugycHadQGRe0+055eGUzdQtORR+u5LuJU+oYXHkw
4V1z0S1QPO9BWixHA62PtjAOShT2xN+1v8T2gq3mdgCEMCHX/Nj6INuu+HXF8o0D
0sC5AfEwq24oKF/6Q8vb1L/KUzFeitnWBnxS1i9XONlG9FTpSGfBir9szqz3QtMu
8Sma+X4g/i/rbO5ZtY9v0r+NCh0fY+fMj8Iaqw8IJUcUWcL2oz+GaHU+CIaJWUyk
suqjw5Zw9WVPQ2nXHZTVOKPk4b8Y8f34GvoqP9ZyVFhZ+/9xcvE3fHOoZKeIK9Yi
4Bxza2HvWRkkKc48Orf5AjK45Wm/G72m72d/KiYfzBm0W4T5QkVqRnX+vpoQc+bo
thEE715ma9SnZMcY3fRcPnhjlDxDneB5DD7WNdiz+wZL0OiHW/kT8Eo4/OZnb72M
t44hd8xB8wbfhz5/zmgmlG4IGGA4MomZyg7G/fo24xaIqkjgnJ1GryWaztNQM6Xx
34kDLTF1fkjqmMZOtTEFKwC5dzrp1qb7B4ZWsFXC+bSLC5teaRajmOr4T5tXCFV7
TL0gNBsg/bRBU6wmFDaOaJjleoTsh/7YNJaMsoiMx7NrHe+uVqaEbE4HsWU=
=tlCO
-----END PGP MESSAGE-----

--=-=01-iacqg4og4pqz11a5cg1o=-=--
```

## Appendix B. Changes Since -13

- Fixed description of the confiration response

- Fixed an example to be signed+encrypted

- Clarified some inconsistencies

## Author's Address

Werner Koch
GnuPG e.V.
Rochusstr. 44
40479 Duesseldorf
Germany

Email: wk@gnupg.org
URI: https://gnupg.org/verein