


OW2con'21

# OpenPGP Web Key Directory

<http://dashohoxha.fs.al/web-key-directory/>

Dashamir Hoxha  
dashohoxha@gmail.com



# What is WKD?

If you use OpenPGP/GnuPG to secure your email communication, you want to share your public key with your contacts.

[Web Key Directory](#) is a way for sharing your key by publishing it on the WWW, on a well-known URL, where email clients can find and download it automatically.

# WKD Benefits

Once you publish your key on the WKD, your contacts will be able to:

- Verify your signed emails automatically, because their mail clients can find and fetch your public key automatically.
- Send you immediately encrypted messages, without asking for your public key first, because their mail clients can find and fetch it automatically.

WKD is supported by almost all the mail clients (Thunderbird, KMail, Outlook, etc.)

# WKD Alternatives

Some traditional ways for sharing your public key are:

- Export the key and attach it to the first email.
- Publish it on the so-called "keyserver network", where your contacts can find and fetch it.

Both of these methods have their problems and are not recommended.

**WKD is the recommended way.**

# How WKD Works

Before building a WKD, let's try to understand first how it works by following the steps that a mail client would do to locate the public key corresponding to a certain email address.

Let's say that the email address is: **user@example.org**

## How WKD Works: 1. Check for WKD existence

A mail client will check first whether a WKD for the domain **example.org** exists.

1. It does it by checking first for the presence of:

<https://openpgpkey.example.org/.well-known/openpgpkey/example.org/policy>

2. If it fails, then it checks for the presence of:

<https://example.org/.well-known/openpgpkey/policy>

The first one is called the **advanced** method, and the fallback one is called the **direct** method. The file **policy** is usually just an empty file.

## How WKD Works: 2. Get the key from WKD

Depending on which WKD method exists for **example.org**, it tries to download the public key from one of these locations:

- For the **advanced** method:

<https://openpgpkey.example.org/.well-known/openpgpkey/example.org/hu/nmxk159crbcuk3imqiw13gkjmfwd8mqj?l=user>

- For the **direct** method:

<https://example.org/.well-known/openpgpkey/hu/nmxk159crbcuk3imqiw13gkjmfwd8mqj?l=user>

The directory **hu** stands for *hashed-userid*, and **nmk159crbcuk3imqiw13gkjmfwd8mqj** is indeed a kind of hash of **user**, the local-part of the email address. For now we don't need to know how to calculate or find out this hash.

# Building a WKD

First, let's build a WKD with the direct method, since it is a bit simpler than the advanced method.

The **direct** method requires that **we have access to the webserver of the domain `example.org`**, and that **the server supports HTTPS**.

The **advanced** method, additionally, **requires access to the DNS server of the domain `example.org`**



## Building a WKD: 1. Create the directory of public keys

1. On the web server create the web key directory like this:

```
mkdir -p /var/www/html/.well-known/openpgpkey/  
touch /var/www/html/.well-known/openpgpkey/policy
```

2. Export the public key in binary format (not ASCII armored):

```
gpg --no-armor --export \  
user@example.org > nmxk159crbcuk3imqiw13gkjmfwd8mqj
```

3. Transfer it to the web server, and save it at:

```
.well-known/openpgpkey/hu/nmxk159crbcuk3imqiw13gkjmfwd8mqj
```

## Building a WKD: 1. Create the directory of public keys

We can get the *hashed-userid* from a **gpg** command like this:

```
gpg --with-wkd-hash --fingerprint user@example.org
gpg --with-wkd -k user@example.org
```

The output will be something like this:

```
pub   rsa3072 2021-04-22 [SC] [expires: 2023-04-22]
      901D C530 A8E1 DEBA FED0 6C25 C802 3646 1A8D CFC2
uid   [ultimate] user@example.org
      nmzk159crbcuk3imqiw13gkjmfwd8mqj@example.org
sub   rsa3072 2021-04-22 [E]
```

## Building a WKD: 1. Create the directory of public keys

The same way we can publish the public keys for more email addresses, like **user1@example.org**, **user2@example.org**, etc.

At the end, the directory should look like this:

```
/var/www/html/.well-known/  
├── openpgpkey  
│   ├── hu  
│   │   ├── nmzk159crbcuk3imqiw13gkjmfw8mqj  
│   │   ├── sxpkq64cy1wikgh8o8eddrx6bg8urzu8  
│   │   └── wgrbabzq3fs5uryhxq96e8nnwxae78fw  
└── policy
```

The web server of WKD:

- Should disable directory listing.
- Should have the right CORS headers.
- Should use **application/octet-stream** as the **Content-Type** for the data of public keys.

For apache2 the configuration should look like this:

```
<Directory "/.well-known/openpgpkey/hu">  
    Options -Indexes  
    ForceType application/octet-stream  
    Header always set Access-Control-Allow-Origin "*"   
</Directory>
```

**Note:** It requires that the *apache2* modules **mime** and **headers** are enabled.

# Testing the WKD

We can test the WKD by downloading a published key with **wget**.

First of all we need to find out the *hashed-userid*, and we can use **gpg-wks-client** for this:

```
apt install gpg-wks-client  
alias gpg-wks-client='/usr/lib/gnupg/gpg-wks-client -v'  
gpg-wks-client --print-wkd-hash user@example.org
```

The output looks like this:

```
nmxk159crbcuk3imqiw13gkjmfwd8mqj user@example.org
```

## Testing the WKD (2)

Now we can construct the URL manually and download the public key:

```
wget -q -O user-example-org.pubkey \  
    https://example.org/.well-known/openpgpkey/hu/  
                                nmxk159crbcuk3imqiw13gkjmfw8mqj?l=user  
gpg --import user-example-org.pubkey
```

---

Other ways for testing that the published key is accessible via WKD are these:

- Using `gpg --locate-keys` like this:  

```
env GNUPGHOME=$(mktemp -d) \  
    gpg -v --locate-keys user@example.org
```
- Using a WKD validator like this one:  
<https://metacode.biz/openpgp/web-key-directory>

# Publishing the keys of an organization

We have seen already how to publish keys manually one by one, but as the number of keys to be published gets large, it becomes tedious and error-prone to manage them.

However it is possible to publish them in bulk (using **gpg-wks-client**).

Let's assume that we have in our GnuPG keyring the public keys of the members of an organization (for example they were sent to us by attachment, and we imported them).

## Publishing the keys of an organization (2)

We can export all these keys into a WKD format like this:

```
mkdir wkd  
gpg --list-options show-only-fpr-mbox \  
  --list-keys "@example.org" \  
  | gpg-wks-client --install-key --directory wkd/
```

The GnuPG keyring is searched for all public keys (**--list-keys**) matching the defined pattern (**@example.org**), and the output shows only *fingerprint* and *user\_id* values, like this:

```
901DC530A8E1DEBAFED06C25C80236461A8DCFC2 user@example.org  
D67E52B28F276D2AE5250B4EFF09740FC5FD1300 user1@example.org  
38186EF3FDA463907B494A4AEDF1E29CB878858F user2@example.org
```



## Publishing the keys of an organization (3)

With the input from the first command, **gpg-wks-client** creates a WKD directory structure that looks like this:

```
$ tree wkd/
```

```
wkd/  
├── example.org  
│   ├── hu  
│   │   ├── nmzk159crbcuk3imqiw13gkjmfw8mqj  
│   │   ├── sxpkq64cy1wikgh8o8eddrx6bg8urzu8  
│   │   └── wgrbabzq3fs5uryhxq96e8nnwxae78fw  
│   └── policy
```

## Publishing the keys of an organization (4)

We can use **rsync** to synchronize the directory **wkd/example.org/hu/** with the one on the webserver, for example:

```
rsync -a --delete \  
  ./wkd/example.org/hu/ \  
  webserver:/var/www/html/.well-known/openpgpkey/hu/
```

# The advanced method of WKD

With the **direct** method the keys are published in a location like:

<https://example.org/.well-known/openpgpkey/hu/>

With the **advanced** method the keys are published in a location like:

<https://openpgpkey.example.org/.well-known/openpgpkey/example.org/hu/>

Everything else is the same.

## The advanced method of WKD (2)

The **advanced** method requires the subdomain **openpgpkey.example.org** to be resolvable.

If we want to use the **direct** method, we should make sure that this subdomain is **not resolvable**, since WKD clients will first try the **advanced** method, and only if the **openpgpkey** subdomain is not resolvable will fall back to the **direct** method.

While the **direct** method is *simpler* because it does not need any DNS modifications, the **advanced** method is *more flexible* because:

- It allows us to use a WKD server that is different from the web server.
- It allows us to support more than one email domain in the same WKD server.

# Web Key Service (WKS)

So far we have seen how to publish keys manually (one by one or in bulk). For a large organization it is more suitable if each member can publish and update his key himself. One way to implement this is with a web interface where users can upload their public key. Another one is to send the public key to WKD by email.

WKS allows users to publish their public key by email, through a specific email protocol.

Thank you for your attention!  
Any questions or comments?

<http://dashohoxha.fs.al/web-key-directory/>

Dashamir Hoxha  
dashohoxha@gmail.com