Support Documentation

USER GUIDE FOR SFTP DATA EXCHANGES WITH ONPOINT

Table of Contents

Welcome to Data Exchanges with Onpoint via SFTP

Onpoint offers our clients' data suppliers two options for transmitting their data to Onpoint: (1) the upload tool available online in the secure Onpoint CDM portal and (2) secure file transfer protocol (SFTP). For delivering data to our clients' authorized data recipients, we leverage SFTP.

These file transfers – both to and from Onpoint – are conducted over an encrypted tunnel via SFTP, with easy access to Onpoint's SFTP server from a wide range of SFTP client utilities and open-source solutions (e.g., WinSCP, FileZilla, etc.).

SFTP data exchanges with Onpoint must be both encrypted using the OpenPGP standard and signed by the sender prior to transfer to ensure file integrity. SFTP data submissions to Onpoint additionally must be performed using Secure Shell (SSH) for an additional level of authentication.

This user guide provides a walkthrough of the key steps to establishing connectivity for data exchanges with Onpoint, including generating SSH and PGP keys and setting up PGP in Windows and Linux.

Step 1. Confirming Your Contact Information

The first step to connect to Onpoint's SFTP server is completing Onpoint's SFTP registration form so that we can credential the appropriate contact(s) for your organization. If you have not already provided a completed registration form to Onpoint, please use the following link to download a registration form to get started: http://www.onpointhealthdata.org/docs/sftp_pgp/onpoint_sftp_registration_form.docx

Step 2. Provide Onpoint with Your Public PGP Key

Prior to either packaging your files for delivery or decrypting data received from Onpoint, our organizations must exchange public PGP keys so that each party can authenticate that the exchanged data originated with one of our organizations. If you have not done so already, please email your public PGP key to Onpoint. (If you did not receive Onpoint's public PGP key as part of the welcome materials, please let us know so that we can provide you with a copy.)

Step 3. Provide Onpoint with Your Public SSH Key

In addition to your public PGP key, you will need to provide Onpoint with your organization's public Secure Shell (SSH) key. SSH is a cryptographic network protocol that allows an additional level of authentication while eliminating the need for passwords. If your organization does not have an existing SSH key or wishes to generate a new key – either for this specific project or for the first time – a walkthrough of the process using WinSCP is provided below. If you have not done so already, please email your public PGP key to Onpoint.

Important: Onpoint will never ask for your private PGP or SSH keys, which should remain securely within your organization. If you accidentally send Onpoint your private key(s), we will shred the key(s) and require that a new keypair be generated prior to the next data exchange.

Step 4. Prepare to Connect to Onpoint

Onpoint's SFTP server information is supplied below for registration in your system.

Item	Detail
Fully Qualified Domain Name	sftp://transfer.onpointhealthdata.org
Port (SSH)	22

⚠️ **Important:** Before we can proceed with next steps, your SFTP registration form, your public PGP key, and your public SSH key must be sent via email to Onpoint at support@onpointhealthdata.org.

Next Steps

After Onpoint has received and processed your (1) SFTP contact information, (2) public PGP key, and (3) public SSH key, we will send an email to the Project Contact identified in your registration form, providing their assigned username.

Using that username, they may access Onpoint's SFTP server at any time by logging in to the following site with their username and SSH key via any SFTP client utility: sftp://transfer.onpointhealthdata.org.

Upon successful login, we recommend uploading a simple text file with PGP encryption to verify that the connection is working successfully.

⚠️ **Important:** When preparing files for SFTP transfer to Onpoint, we highly recommend that each file's name be unique – for example by including a date/timestamp, incrementally numbered suffix, etc. This prevents file overwriting, allows for more effective tracking of received files, and facilitates any troubleshooting and responses to follow-up questions.

Getting Support

If you have any questions or need assistance, please contact us (support@onpointhealthdata.org | 207-623-2555). We look forward to working with you.

Additional Resources

FileZilla

Download Page .... https://filezilla-project.org/download.php?type=client

Installation Guide & Help Page .... https://wiki.filezilla-project.org/Documentation

Kleopatra/PGP for Windows

Download Page https://www.gpg4win.org/download.html

WinSCP

Download Page .... https://winscp.net/eng/download.php

Installation Guide & Help Page .... https://winscp.net/eng/docs/guide_install

SSH Key Generation Walkthough (WinSCP)

There are many ways to generate an SSH key pair, which includes both a public key (to be shared with Onpoint) and a private key (that should **never** be shared with Onpoint). The following steps walk through the key-generation process on a Windows platform using the free application WinSCP.

1. Download and install the version of WinSCP software appropriate for your system (https://winscp.net/eng/download.php).

2. Open the software and select Run PuTTYgen from the Tools drop-down.
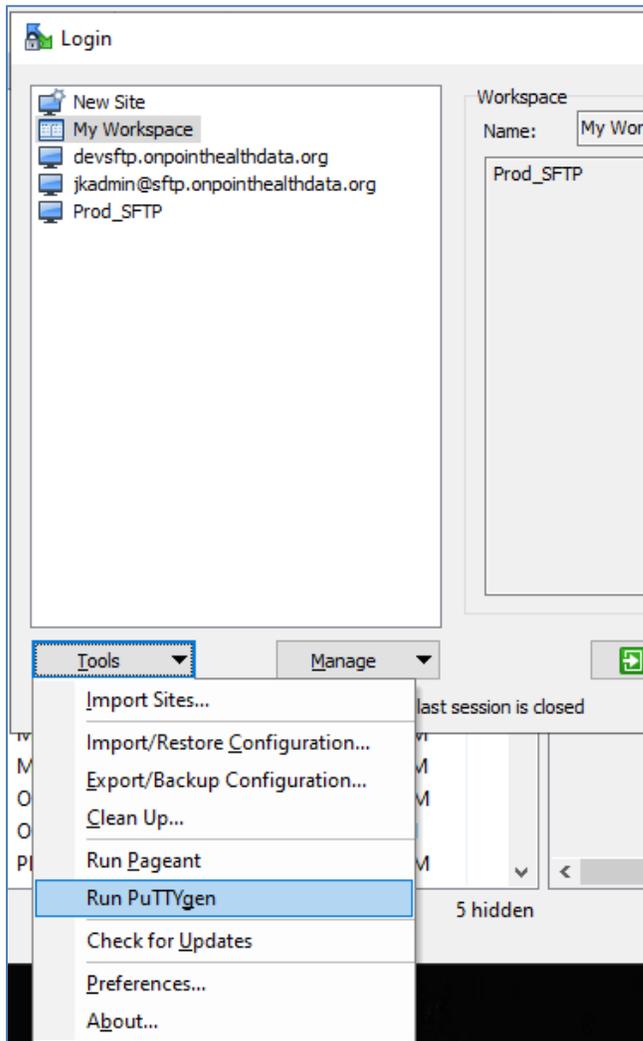
3. With the "PuTTY Key Generator" window open, select RSA from the "Parameters" section. Then click the Generate button from the "Actions" section. Note that while both the Key comment and Key passphrase are optional, the latter is encouraged for security purposes. (Note, however, that if you lose or forget your passphrase, a new key will need to be generated.)
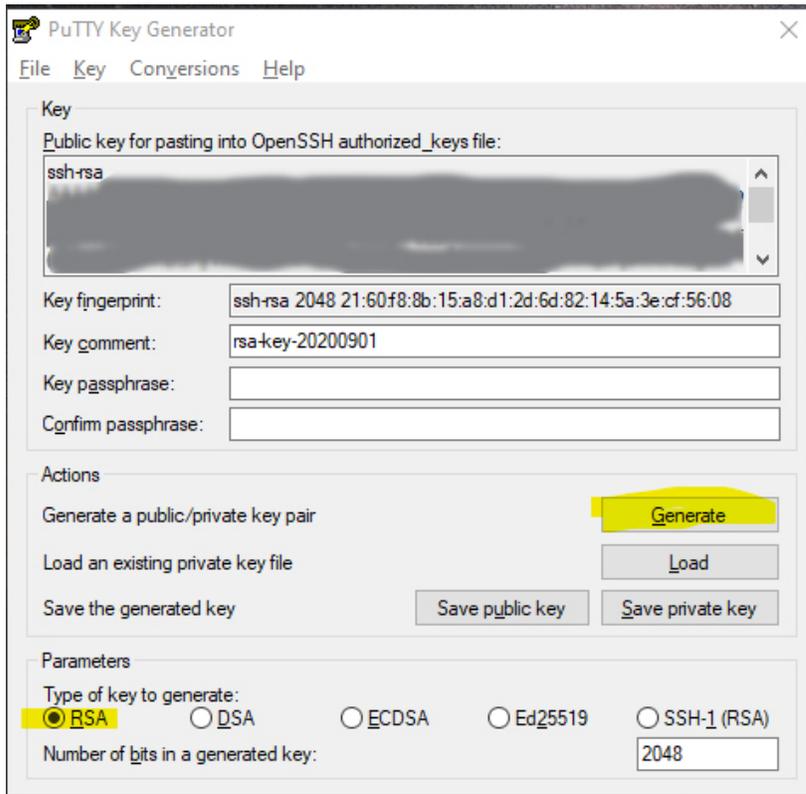
4. Save your public key file, **including the word "public" as part of the key's file name** to clearly differentiate it from your private key, which should never be shared with Onpoint.

5. Email this public-key file to Onpoint (along with your public PGP key) for registering in our system.

6. Save your private key file for use in combination with your assigned username to access Onpoint's SFTP server.

Getting Started with PGP

This guide provides an overview of PGP encryption and its functionality to support authorized data exchanges with Onpoint Health Data's servers. The following pages detail how to use PGP encryption successfully, including standards for secure exchanges, a review of general best practices, and examples for both Windows and Linux applications.

Many organizations already have PGP protocols and keys in use. If this is the case with your organization, please reach out to your internal technical team for implementation support and internal technical requirements in order to streamline encryption setup. Please refer to the contents below for more details about Onpoint's standard process and contact support@onpointhealthdata.org for more information if issues are encountered.

Founded in 1991 by Phil Zimmermann, PGP (an acronym for "Pretty Good Privacy") is an asymmetrical public-key cryptography system that allows for secure exchanges of messages between two parties even if those two parties have never had secure communications in the past. PGP is based on the premise of two keys — one that is *private* (sometimes referred to as *secret*) and one that is *public* — with the concept of the Diffie-Hellman key exchange underlying the security and integrity of the system.

When using PGP, the primary objectives are threefold:

1. **Confidentiality of communication.** The messages must be unreadable to anyone other than the intended recipient.

2. **Reliability of the source of information.** Only the expected source can actually be the true originator of a message; any other source will have a different signature.

3. **Integrity of a message.** The message must be complete and not changed accidentally or intentionally.

Onpoint employs an OpenPGP-compliant solution, which is compatible with most industry-standard PGP products today. As in any public-key cryptosystem, PGP public keys may be shared far and wide, so long as the private key is kept entirely secret; Onpoint should never possess your private key, so *keep it safe!* Note that any file encrypted with one half of the key pair may be decrypted with the corresponding other half of the key pair (whether the encryption key is public or private). This guide doesn't delve into the deeper technical aspects of PGP or public-key cryptography; please see the Additional Resources section for more information as desired.

PGP Key & Exchange Requirements

A secure and productive message exchange relies upon the effective use of private and public keys. There are several basic requirements that Onpoint has established for accomplishing this, most of which are based upon guidelines developed by the U.S. National Institute of Standards and Technology:

1. **Keys must be a minimum of 2,048 bits in length.** This requirement applies uniformly to the use of RSA, DSA, and ElGamal signature algorithms. Any signatures containing fewer than 2,048 bits under PGP usage is considered weak, with active exploits for key sizes of up to 1,024 bits already having been discovered. Onpoint recommends using 4,096 for the highest level of protection. *Note: Elliptic-curve (EC) keys of any length are not supported.*

2. **Any public key must be signed by its coordinating private key.** This is required for validation that the holder of the public key is also the party in possession of the private key.

3. **Any files exchanged must be signed by their sender.** In most applications, the signature is comprised of a hash of the file (i.e., a unique string of characters representing the file's contents). This hash is then encrypted using the sender's private key, making it able to be decrypted with the sender's public key. This, in turn, allows verification of the file's contents and the sender's identity in one step.

4. **All keys must be OpenPGP-compliant.** All phases of key generation, encryption, hashing, and signing must use standard algorithms and declare them per normal OpenPGP standards. If the application in use is OpenPGP compliant, operations are much more likely to function properly.

PGP Solution Walkthrough (Gpg4win & Kleopatra for Windows)

There are two primary recommended solutions, depending on the operating system: **Gpg4win**, a Windows-based application that is freely available, and **GnuPG** (GPG), a Linux-based command-line implementation built into most distributions.

⚠️ **Note:** While these two solutions are presented, any OpenPGP-compliant application will function as well. If your organization leverages such an application and has existing infrastructure, Onpoint will likely be capable of coordinating through it.

This section covers installing Gpg4win, configuring for first-time use, generating a key pair, importing Onpoint's public key, and encrypting/signing files.

Key Configuration & Management

1. Install Gpg4win on your Windows system. The installer is freely available through http://www.gpg4win.org/download.html. Note that all components must be installed for effective usage.

2. Open Kleopatra, which is the primary interface used for interacting with keys and files under the Gpg4win platform. Look over the user interface (UI) and note that there are several basic components:

   A standard menu bar for advanced commands:
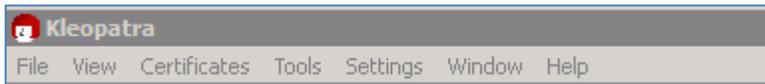
   A quick-access ribbon for common command:

   A primary window for displaying certificates/keys:
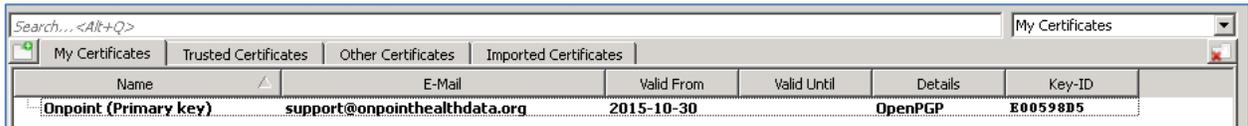
3. Navigate to File → New Certificate. A dialog box will appear. Select Create a personal OpenPGP key pair from the options Certificate Creation Wizard and click Next:

4. Enter the name of the key that will be used to encrypt data being sent to Onpoint, the email address of your technical contact, and any comments that might be useful for identifying the key in the future. Click on the Advanced Settings button and verify that the radio button for RSA 2,048 bits (default) is selected as the key material and that the + RSA 2,048 bits (default) sub-selection is checked.

This certificate/key will be used for multiple activities, so please ensure that the following three options have checkmarks in the Certificate Usage section: Signing, Encryption, and Certification. Then click the OK button to close the pop-up dialog box followed by the Certificate Creation Wizard's Next > button to advance to the next step.
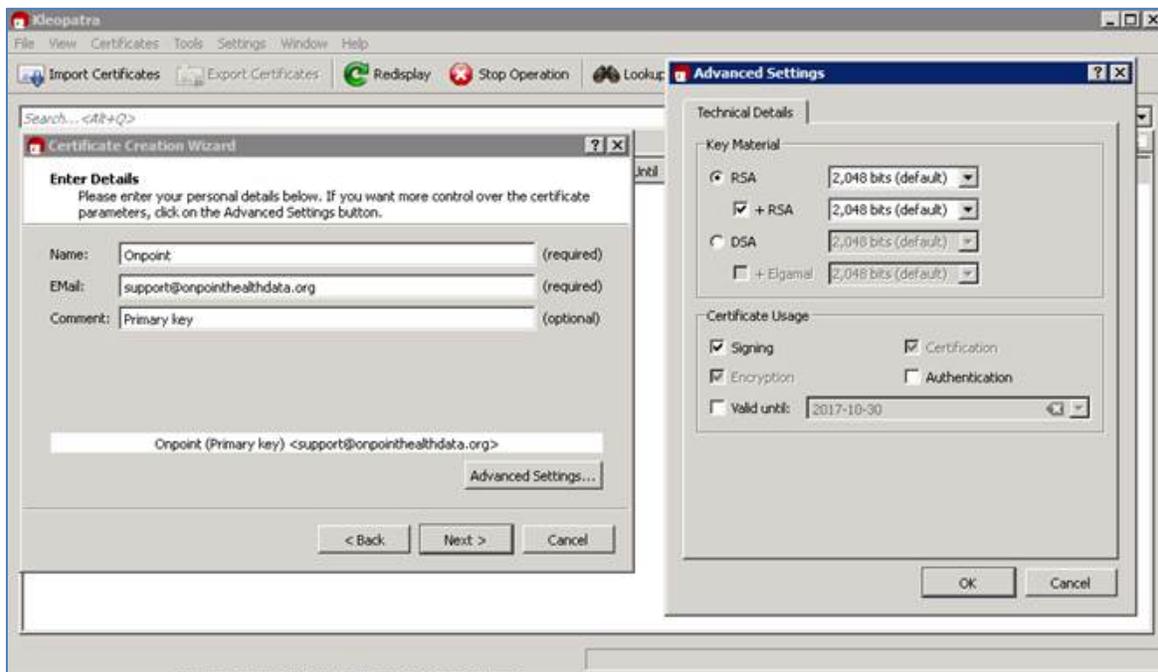
5. Verify the details on the next screen by checking the Show all details option at the bottom of the window to ensure that all details and settings have been correctly established. Click the < Back button if you need to make corrections; otherwise, click the Create Key button when done.
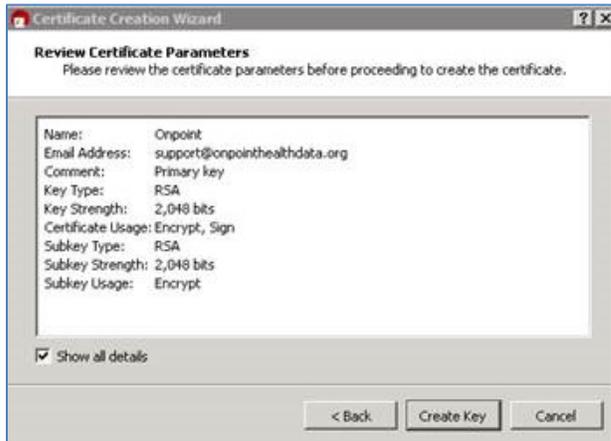
6. Enter a secure passphrase in the pop-up window, then re-enter it to confirm accurate entry. The wizard will prompt for random entries in a text-entry space to build entropy for the new certificates. Mash on the keyboard, start a high-disk-usage program, or don't do anything and let it use the regular system entropy pool. On some systems, this process can take several minutes if there has been a low level of recent activity.

7. Once the key pair has been successfully generated, the wizard will display a confirmation and fingerprint of the new key pair. After backing up the key pair as desired, click the Finish button.
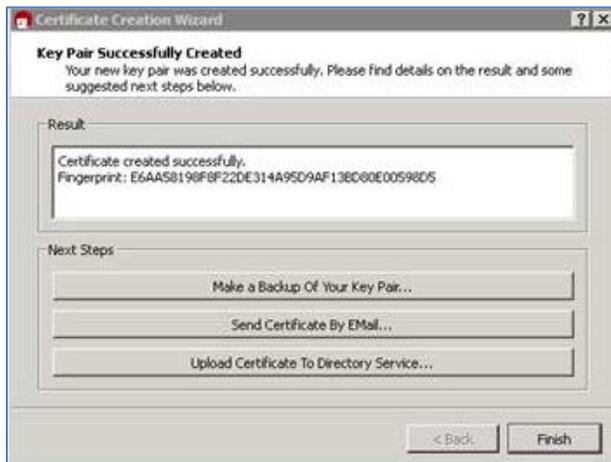
8. The new key pair will show up in the My Certificates tab of Kleopatra's main window:
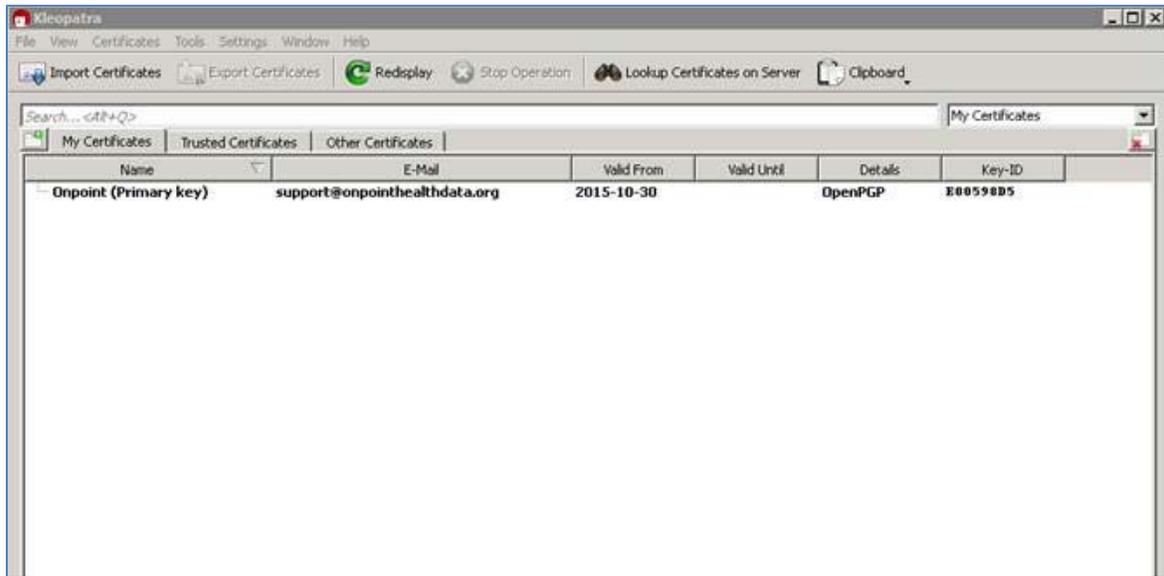
9. *Please note that your public key has already been signed/certified at this point by your secret key*. This is one of the baseline requirements for effective key distribution and trust under the OpenPGP and public-key cryptography concepts.

10. Notice the Key-ID value displayed in Kleopatra; this is the short ID of the key fingerprint, which is used to help identify the key when encrypting or verifying results.

11. It is now time to export the public key. Right-click the newly-generated key pair, click the Export Certificates option beneath the main window's menu, then choose a location to export the key file.
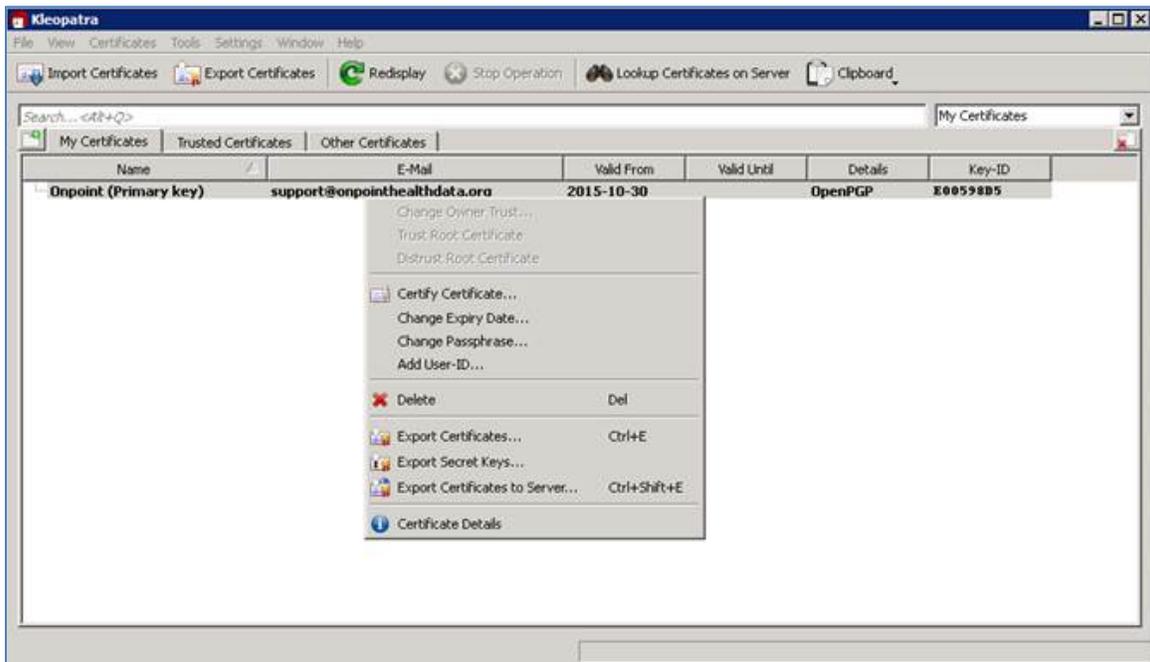
12. Opening the file, it can be seen that it's labelled as a public key; no private keys are included. For example, the demonstration public key file contents:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQENBFYzsekBCADsiSSBVuieM19pUbepVHODHwqIIfhWI1Dex42B6/kLrSWw8nPB
AV52Nar5wlPNmie0rOcklBKc7GrnmazqFg9+2SF0BBx21jBj/+OWXm7F7TKAmjEa
9ttW1Gyk56lpaX6gZOOSZl/0noApLZQOtcDjuTCmLt9Yv3ogB4aMrjRUAyTxX+60
ldCjPObxC9DBEuJCaKaqIBrFWQM6x2kv/R4b0jMQEYTu2PIXvymO/DEikh0cAwMp
2CDw0iaFE+2A8fm6cIELY0nmgYdL/Nzd6pHKMAD4vE+VIq8EW2RxkeZUGXxgSUAj
WQCdOOM1Lk9GEBr9yk0Zknt4AV7r5wWhBgFpABEBAAG0NU9ucG9pbnQgKFByaW1h
cnkga2V5KSA8c3VwcG9ydEBvbnBvaW50aGVhbHRoZGF0YS5jcmc+iQE5BBMBCAAj
BQJWM7HpAhsDBwsJCAcDAgEGFQgCCQoLBBYCAwECHgECF4AACgkQrxO9gOAFmNVO
hggAyRxA85yzATiZ6AFsXDxYcokY7t5d3SI9oLeZp4fNC81UliFYGxkVo9zgWMSn
HxESfWzSJT937Q95wil45jntveE/peZFpIYUZA9J8uR3vt4x+EoRu/VwcHtwA79c
+AYxNvlNN9mj0/J4XLU1KUzQxiR8dWS0cFfs1NIi9hZeBtzmJLSq3JGO9pCpSzBf
LtjBLhNL1ZsGbuIgmfaFMM/2EYW2Ya1PBq0t5KDnYonAYZ88UiIpUKWWJdoprZcg
mWgoxPwT2xE3zga84oH1myrZjaCw2yLmfljNqDuHRsizGZbB8W2MApxy1uPmbhO0
PHj/T74Algsb6BD+14jzOQypVLkBDQRWM7HpAQgA0OO6gOqB7yg1THht+1efz3AE
jJNLzo7wrhDo9cwC4vU03ep6rGD1MwpZP4B4a8teIhvJDb4sQ7BOvIt6I4byy5eb
Ov9hMyNDLGuSvqUZyR2fbkeE6BwN/O4ke6ys/wn6rdfpqZHscKeGobrPEY8bg9DD
QKisfiGSQkEkSaZzjaher8xhuToChl+u93CoBCM2K9rCEYZEPgGLXT4LEEEzw2JT
6sdanRBt2VzZQOnT3a9gkYs8h/YWY8u+QAPmw+HfOHg2LEoAKeJSzAa5024behtv
AC0zZGbItaXxzYoLkqG/vmU4HN0olXk8Vph6n+YhOLlRLmlIrE7LBTn1pHRlowAR
AQABiQEfBBgBCAAJBQJWM7HpAhsMAAoJEK8TvYDgBZjVPg4IAIvqTu1K8xoXZmaw
x2L/gynYQ8Ub61hyAteR1EMvLFBscm6HtcGCZ83GPeqxalJKj3THxzbIGMlIAXuQ
4dHIc5Ws6jqrEkBDNBktYCNeZWrvsrGludYycuSPYF94JpZMZaOPuqYsrrooynkB
C1eOSsvD4fZ/d5Hh0fx7Vot82rjFq7VMrfW+MACYvpxkz26gw0GvrRMvaITX20rU
+z72dPexkZGXF7mYWbY96mletoWlha3CUkTfjgaSDx8jU8oOrKjF6IVAh1NQGQts
r8grpbUW6oJT1p/V8U85g9tt7+pTHYIypp5/gX9321Nx7/Mn5cuHPpd0BIvV/zg0
TdfP6Fs=
=0FgR
-----END PGP PUBLIC KEY BLOCK-----
```

⚠ **Note:** The above key is for demonstration only. Do not use for encryption purposes.

⚠ **Important:** Kleopatra also provides methods for exporting private keys. **This is not recommended** unless needed for migration from one system to another or as a backup copy to be securely stored. A private key should never be revealed to anyone that would not be considered a true representative of the organization, as that party would then be able to sign messages successfully and act as the organization for all interactions. **Keep your private key safe and secure!**

13. The generated public key file is able to be distributed freely to any OpenPGP partners and should be remitted to Onpoint. In addition, the ONPOINT_HEALTH_DATA_PROD public key provided by Onpoint must also be imported into this new keyring. Click the Import Certificates option beneath Kleopatra's main window menu, and choose the provided public key file.
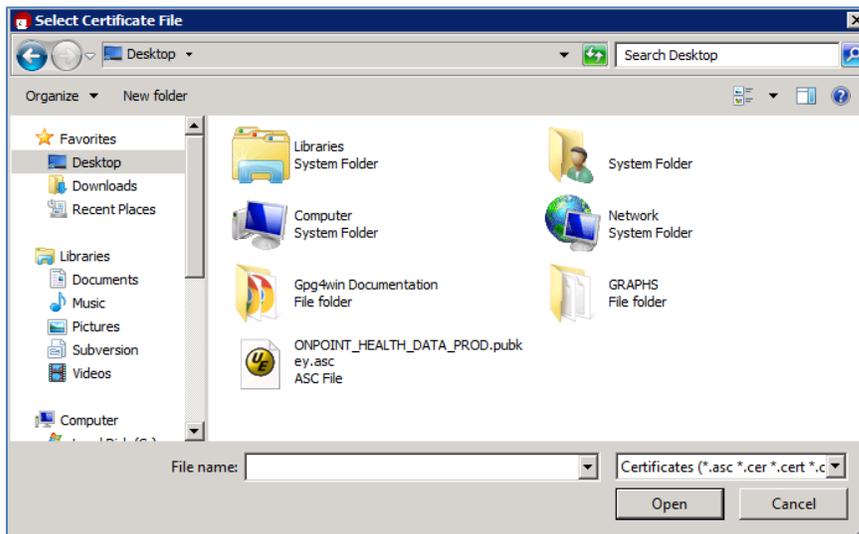
14. The certificate now should be visible in the Imported Certificates tab in Kleopatra's main window.
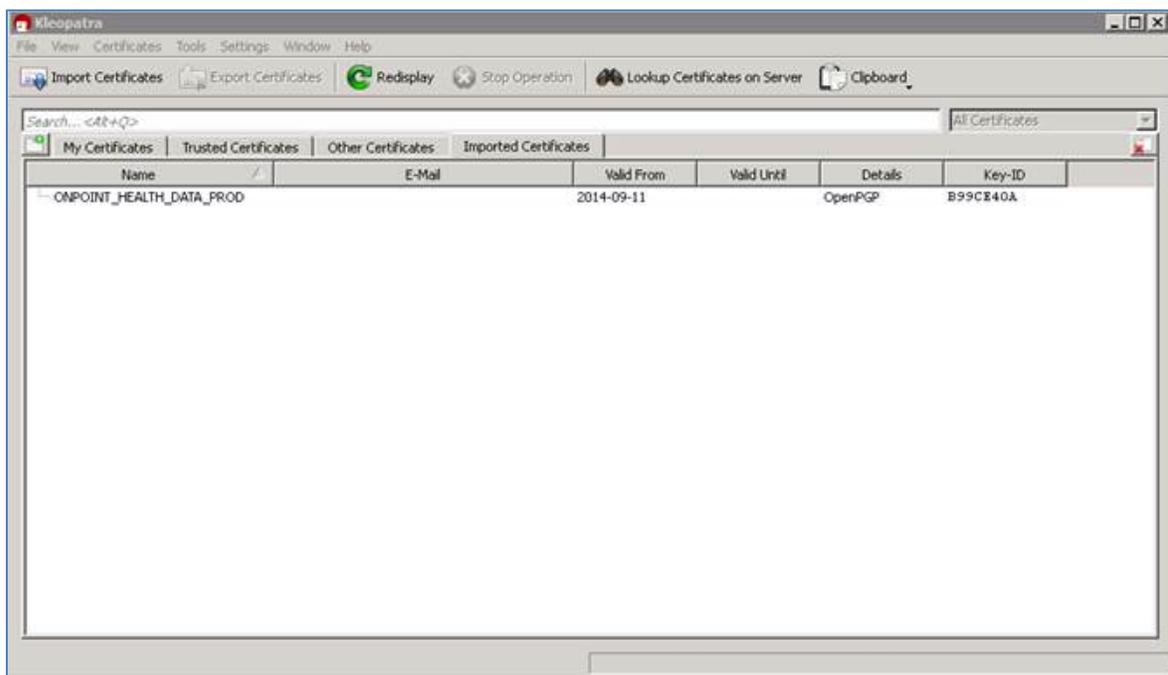
15. Congratulations! Kleopatra has now been fully configured, and all needed keys are in place.

Encrypting & Signing

1. From the File menu, select the Sign/Encrypt Files option, then locate and select the file that you plan to encrypt. In the following window, ensure that the file you wish to encrypt is shown in the Selected file: field. Choose the option to Sign and Encrypt (OpenPGP only), then click the Next > button.
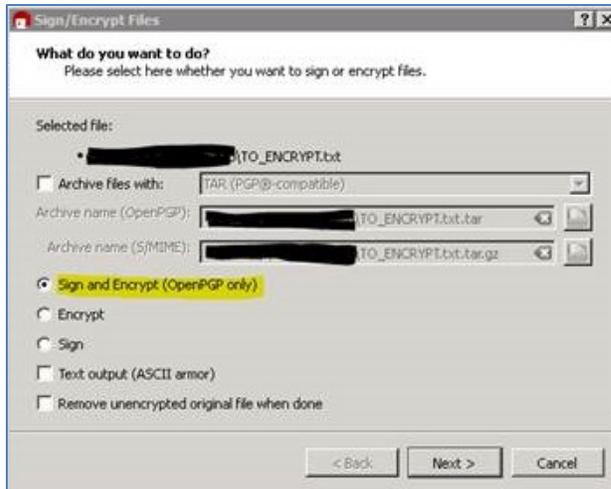
2. In this window, the prompt asks you to identify the recipient of your encrypted file. Select ONPOINT_HEALTH_DATA_PROD in the upper area, then use the Add button in the middle of the screen to move it to the lower pane, which lists the recipients for whom your file will be generated. Then, click the Next > button.
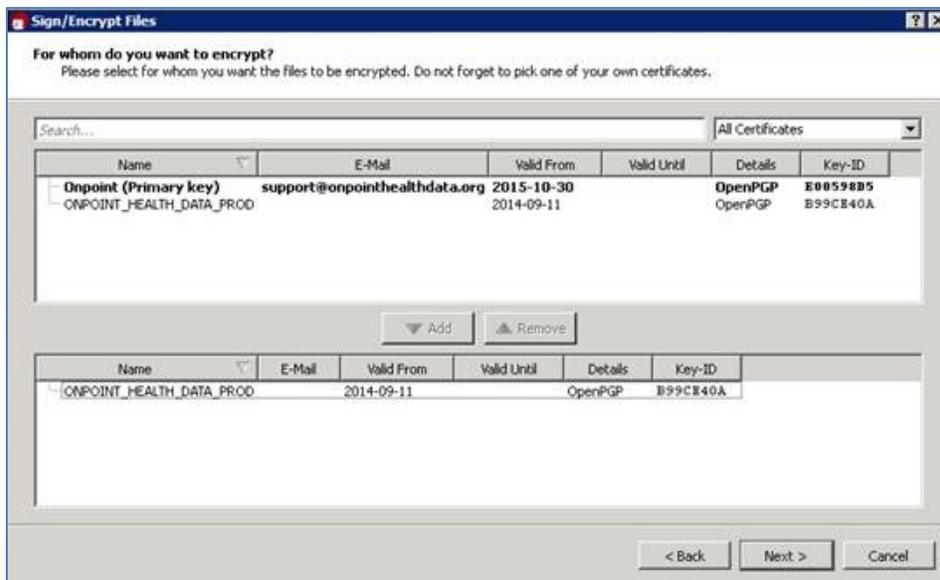
3. There will be an alert, warning you that once the encrypted file is generated, you will not be able to decrypt it. Unless configured differently, Kleopatra leaves the original unencrypted file in the source location, which ensures that data loss does not occur. Click the Continue button to proceed.

4. The final window will ask which signing certificate to use. **The key selected here must correspond with the public key registered with Onpoint; if a different key is selected, validation will fail and re-encryption will be required.**
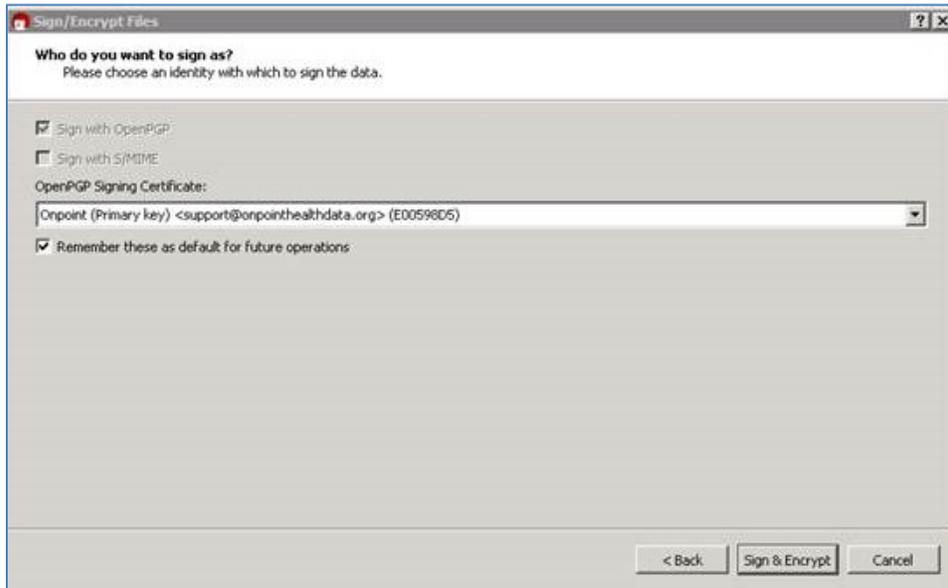
5. When ready, select Sign & Encrypt, then enter the password for the key listed in the OpenPGP Signing Certificate field. A success window will display once the encryption concludes.
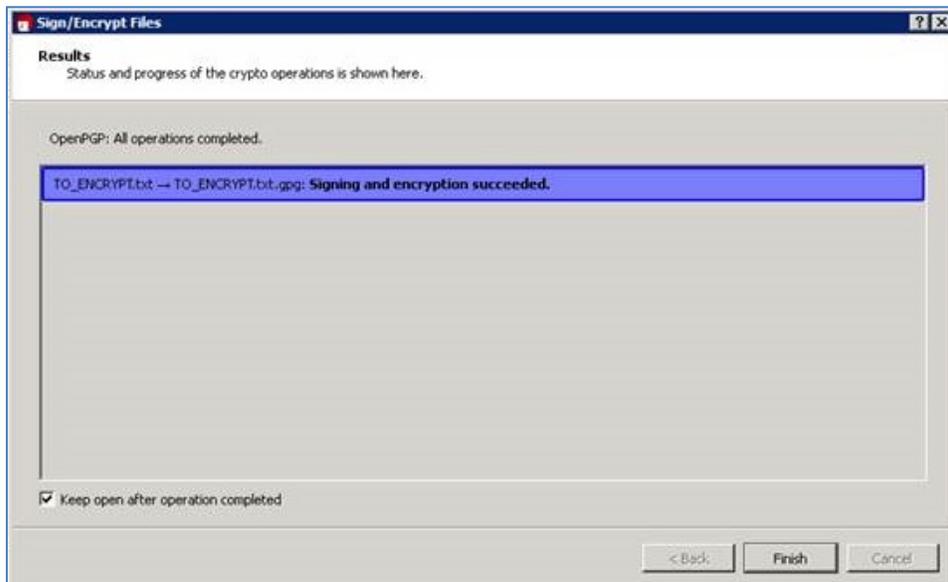
6. Done! The encrypted ".gpg" file is now ready to be remitted to Onpoint and is fully protected and signed with the appropriate key pair.

Decrypting

In this example, a public key has been imported previously for ONPOINT_HEALTH_DATA_PROD, *and a file called* TEST_DECRYPTION.txt.gpg *from that entity has been received.*

1. From the File menu, select the Decrypt/Verify Files… option, then select the file that should be decrypted. Ensure that the file that you wish to decrypt is shown in the Input file: field. In the Output folder: field, specify the storage location for the decrypted data, then select Next >.
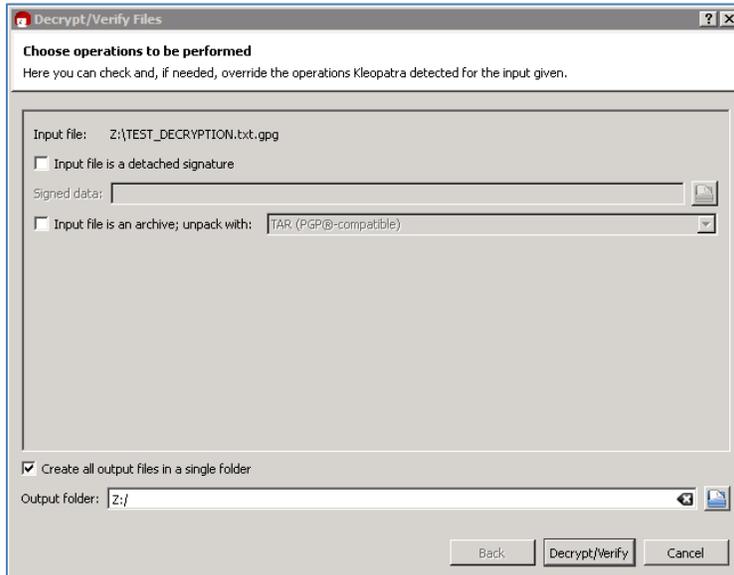
2. In the next window, a prompt will appear requesting the passphrase for unlocking the coordinating secret key based on the fingerprint found on the encrypted file. Select OK when complete and allow decryption to complete.
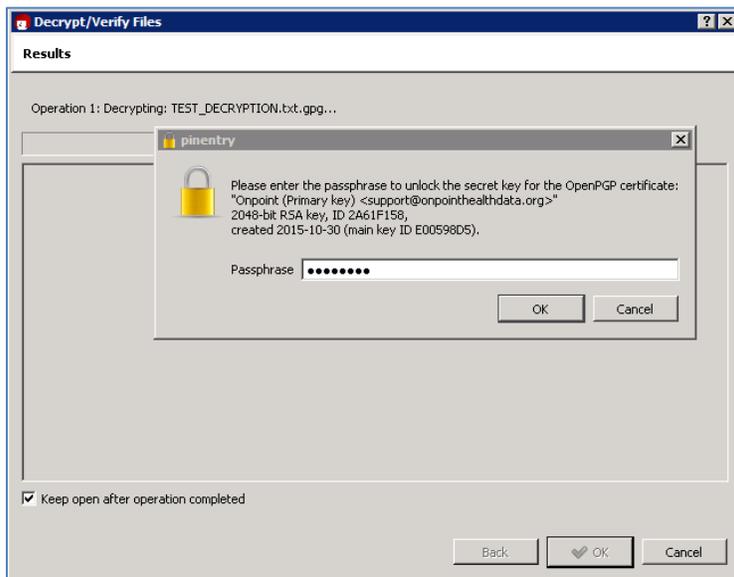
3. Done! The decrypted file can now be found in the output folder location.

PGP Solution Walkthrough (GnuPG for Linux)

This section covers assessing for GnuPG presence, configuring for first-time use, generating a key pair, importing another public key, and encrypting/signing files.

Key Configuration & Management

1. Log in to the Linux system that will be used for key management and encryption and ensure that the user who should own all of the keys is logged in properly. **No other user will be able to access the keys.**

2. In this walkthrough, commands are run on a Red Hat Enterprise Linux system, which leverages the yum platform. Installation- and package-related commands may be different if running on another distribution; please adjust to the appropriate procedure for the distribution (e.g., "apt-get", "zipper", "dnf").

3. Run a check to see if GnuPG (GPG) is installed already:

   ```
   [example@server1 ~]$ which gpg
   /usr/bin/gpg
   ```

   If the distribution is up to date and returns a result for the above, the system likely has GPG already installed; skip to Step 5. If the command does not return a path, continue to Step 4. Please see this FAQ (https://gnupg.org/faq/gnupg-faq.html#oses) for more details on whether your distribution has GPG installed by default.

4. Refer to the online instructions (https://gnupg.org/faq/gnupg-faq.html#get_gnupg) regarding how to install GPG on your particular distribution. Ensure that installation has completed successfully before continuing.

5. Identify the version currently installed. Running the version command also outputs information regarding supported public key formats, encryption ciphers, hash methods, and compression methods. Ensure that it looks much like the following, with a GPG minimum version of 2:

   ```
   [example@server1 ~]$ gpg --version
   gpg (GnuPG) 2.0.14
   libgcrypt 1.4.5
   Copyright (C) 2009 Free Software Foundation, Inc.
   License GPLv3+: GNU GPL version 3 or later
   <http://gnu.org/licenses/gpl.html>
   This is free software: you are free to change and redistribute it.
   There is NO WARRANTY, to the extent permitted by law.

   Home: ~/.gnupg
   Supported algorithms:
   Pubkey: RSA, ELG, DSA
   Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
   CAMELLIA128,
           CAMELLIA192, CAMELLIA256
   Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
   Compression: Uncompressed, ZIP, ZLIB, BZIP2
   ```

6. Proceeding to generate a key pair, the `gpg --gen-key` command will be issued. Select an `RSA and RSA` key (generally the first option) from the menu, then specify a key size of `2048` bits:

```
[example@server1 ~]$ gpg --gen-key
gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
```

7. Unless otherwise needed, specify `0` for the key expiration period to indicate that it does not expire, then confirm the selection as correct with a `y` value. GPG will then request a name for the key, an email address, and any optional comments; enter appropriate values for each. Once complete, proceed with `O` (for `Okay`):

```
Please specify how long the key should be valid.
        0 = key does not expire
     <n>  = key expires in n days
     <n>w = key expires in n weeks
     <n>m = key expires in n months
     <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: ONPOINT_EXAMPLE
Email address: support@onpointhealthdata.org
Comment: Example key
You selected this USER-ID:
    "ONPOINT_EXAMPLE (Example key) <support@onpointhealthdata.org>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

8. Enter a secure passphrase, which will be used for access to any private key functions, then re-type it to confirm accurate entry. GPG will proceed with key generation, requesting that a high degree of entropy be provided; per the instructions in the terminal, mash the keyboard, utilize the disks on the system, or perform other random activity until it completes for best results.

```
You need a Passphrase to protect your secret key.


We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 91925125 marked as ultimately trusted
public and secret key created and signed.


gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/91925125 2015-11-03
      Key fingerprint = EF36 77D3 077A 7D4C F7D9  9C0C D61A EA15 9192
5125
uid                  ONPOINT_EXAMPLE (Example key)
<support@onpointhealthdata.org>
sub   2048R/1642360F 2015-11-03
```

9. Take note of the pub and sub keys, as they combine to make the newly generated key pair. Leverage the --list-keys and --list-secret-keys commands:

```
[example@server1 ~]$ gpg --list-keys
/home/example/.gnupg/pubring.gpg
------------------------------
pub   2048R/91925125 2015-11-03
uid                  ONPOINT_EXAMPLE (Example key)
<support@onpointhealthdata.org>
sub   2048R/1642360F 2015-11-03
[example@server1 ~]$ gpg --list-secret-keys
/home/example/.gnupg/secring.gpg
------------------------------
sec   2048R/91925125 2015-11-03
uid                  ONPOINT_EXAMPLE (Example key)
<support@onpointhealthdata.org>
ssb   2048R/1642360F 2015-11-03
```

10. To export the public key, issue the `--export -a` command and redirect the output to a target file, being sure to cite the user name entered in the `UID` field from the list of public keys above (in this case, `ONPOINT_EXAMPLE`). Note that this will automatically sign the public key appropriately without any extra intervention and generates the key file in the specified location:

```
[example@server1 ~]$ gpg --export -a "ONPOINT_EXAMPLE" >
/example/keylocation/example_pubkey.asc


This public key file should be remitted to Onpoint as part of the
registration and encryption testing process.
```

11. Now that appropriate keys have been created, the ONPOINT_HEALTH_DATA_PROD public key also must be imported into the keyring for effective interaction with Onpoint.

    Ensure that a copy of the ONPOINT_HEALTH_DATA_PROD public key file is located on the system and is accessible to the user, then proceed to import it (using the correct path and file name in place of this example's `/example/keylocation/ONPOINT_HEALTH_DATA_PROD.pubkey.asc`):

```
[example@server1 ~]$ gpg --import
/example/keylocation/ONPOINT_HEALTH_DATA_PROD.pubkey.asc
gpg: key B99CE40A: public key "ONPOINT_HEALTH_DATA_PROD" imported
gpg: Total number processed: 1
gpg:                 imported: 1  (RSA: 1)
```

12. Check to ensure that the key was imported successfully by listing the public keyring again:

```
[example@server1 ~]$ gpg --list-keys
/home/example/.gnupg/pubring.gpg
-----------------------------
pub   2048R/91925125 2015-11-03
uid                   ONPOINT_EXAMPLE (Example key)
<support@onpointhealthdata.org>
sub   2048R/1642360F 2015-11-03

pub   4096R/B99CE40A 2014-09-11
uid                   ONPOINT_HEALTH_DATA_PROD
sub   4096R/CC7A8BC8 2014-09-11
```

13. Congratulations! A key pair has been generated securely in the keyring, the public key has been exported and signed, and the ONPOINT_HEALTH_DATA_PROD public key has been registered in the keyring.

Encrypting & Signing

1. Log in to the Linux system with the user configured with the appropriate GPG keyring.

2. Encryption of a file through GPG on Linux can be accomplished with a single-line command, including several important flags and switches:

   ```
   gpg -e --trust-model always -u "FULL UID OF SENDING PGP KEY" -r
   "RECIPIENT [in this case, ONPOINT_HEALTH_DATA_PROD]" -o
   /example/path/ENCRYPTED_OUTPUT.txt.gpg /example/path/TO_ENCRYPT.txt
   ```

   - `gpg -e` tells GPG that it is (-e)ncrypting data.

   - `--trust-model always` tells GPG to disregard assigned trust levels for keys in use on the ring. This switch is not required if trust levels are set and maintained correctly, though this guide will not delve into details of implementation for such usage.

   - `-u "FULL UID OF SENDING PGP KEY"` states the sending user name, signing the file with their private key.

   - `-r "RECIPIENT UID"` states the recipient user name, encrypting the file with their public key.

   - `-o /example/path/ENCRYPTED_OUTPUT.txt.gpg` specifies the output path of the file. Surround with quotes if any spaces are in the file name. Additionally, the file should always end with an extension followed by a `.gpg`. The above encryption is on a `.txt` file, so the extension is `.txt.gpg`, but a .csv file should be encrypted to `.csv.gpg`. **If the extension is improperly specified, the file will likely need to be renamed to enter Onpoint systems.**

   - `/example/path/TO_ENCRYPT.txt` specifies the source file that will have an encrypted output. Any file type may be encrypted as long as the user has access.

3. Issuing the above command shows no output, though validation of the file's presence should occur (for instance, by using the `ls` command).

4. Done! The data file is now encrypted, signed with the sender's private key, and ready for transmission to Onpoint.

Decrypting

In this example, a public key has been imported previously for ONPOINT_HEALTH_DATA_PROD, *and a file called* TEST_DECRYPTION.txt.gpg *from that entity has been received.*

1. Log in to the Linux system with the user configured with the appropriate GPG keyring.

2. Decryption occurs through the `gpg -o /target/directory/file -d /source/encrypted/file` command, with secret key selection and signature validation occurring automatically based on fingerprints found on the file:

   ```
   [example@server1 ~]$ gpg -o /example/path/decryption_output.txt -d
   /example/path/TEST_DECRYPTION.txt.gpg

   You need a passphrase to unlock the secret key for
   user: "ONPOINT_HEALTH_DATA_PROD"
   4096-bit RSA key, ID CC7A8BC8, created 2014-09-11 (main key ID
   B99CE40A)

   gpg: encrypted with 4096-bit RSA key, ID CC7A8BC8, created 2014-09-11
         "ONPOINT_HEALTH_DATA_PROD"
   gpg: Signature made Fri 13 Nov 2015 04:48:41 PM EST using RSA key ID
   E00598D5
   gpg: Good signature from "Onpoint (Primary key)
   <support@onpointhealthdata.org>"
   Primary key fingerprint: E6AA 5819 8F8F 22DE 314A  95D9 AF13 BD80 E005
   98D5
   ```

3. Issuing the `gpg -o "OUTPUT PATH" -d "INPUT PATH"` command will cause GPG to examine the fingerprint on the file and compare against available secret keys. If it finds one that matches, it will then prompt for the passphrase associated with that key; enter this passphrase in the prompt that appears:

   ```
   Please enter the passphrase to unlock the secret key for the OpenPGP
   certificate:
   ```

4. Once entered and validated, GPG will proceed to confirm the matched secret key, identify the type of encryption that was used, and validate the signature found on the file.

5. Congratulations! The file is now decrypted, validated, and ready for further use.

Getting Support

A reminder: If you have any questions or need assistance, please contact us (support@onpointhealthdata.org | 207-623-2555). We look forward to working with you.