GENERATION OF A NEW PGP KEY

Overview

PGP is an encryption program that provides cryptographic privacy and authentication for data communication. PGP is often used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications. PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography; each step uses one of several supported algorithms.

In UniPay, PGP encryption can be used in the following cases:

- PGP encryption of client's batch request files. It is used in cases when tokenization of PAN data is not used.
- PGP encryption of batch files submitted from UniBroker to UniPay and back. It is used in UniPay deployment configurations with UniBroker.
- PGP encryption of processor request files going from UniBroker to a processor's sFTP location and back. It is used when PGP encryption is mandatory with the integrated batch processor.
- PGP encryption of raw card data in real-time transaction requests. It is used when the built-in tokenization functionality is not used.

Please note, that these use cases are not mandatory, and will become required only if a particular logic is used as part of the system configuration.

How to generate a new PGP key

1. To download and install GnuPG for Windows, please follow this link: https://files.gpg4win.org/gpg4win-2.3.3.exe. To download and install GnuPG for Linux, please follow this link: ftp://mirror.switch.ch/mirror/gnupg/gnupg/gnupg-2.1.9.tar.bz2
2. Execute the following command:
   - gpg --gen-key *(for Windows users)*
   - gpg --full-gen-key *(for Linux users)*
3. Follow the instructions and input values as indicated:

```
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? 4
```

```
RSA keys may be between 1024 and 4096 bits long.
```

```
What keysize do you want? (2048) 4096
```

```
Requested keysize is 4096 bits

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; the software constructs
the user ID
from the Real Name, Comment and Email Address in this form:
"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
Real name: [Your Name]
Email address: [Your E-mail Address]
Comment:
You selected this USER-ID:
    "[Your Name] <[Your E-mail Address]>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.
(after you have created the passphrase)
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: NOTE: you should run 'diskperf -y' to enable the disk statistics
gpg: NOTE: you should run 'diskperf -y' to enable the disk statistics
gpg: NOTE: you should run 'diskperf -y' to enable the disk statistics
gpg: NOTE: you should run 'diskperf -y' to enable the disk statistics
.....+++++
.....+++++
gpg: key 0C9ED241 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: 7 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 7u
pub 4096R/0C9ED241 2014-10-25
   Key fingerprint = 48AC 0D74 6C12 2ABA 3348 8E9F 138D 76EA 0C9E D241
uid       John Smith
```

```
Note that this key cannot be used for encryption. You may want to use
the command "--edit-key" to generate a subkey for this purpose.
```

```
gpg --edit-key "John Smith"
```

```
gpg (GnuPG) 2.0.22; Copyright (C) 2013 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.
```

```
Secret key is available.
```

```
pub 4096R/0C9ED241 created: 2014-10-25 expires: never    usage: SC
      trust: ultimate    validity: ultimate
[ultimate] (1). John Smith
```

```
Command> addkey
```

```
Key is protected.
```

```
You need a passphrase to unlock the secret key for
user: "John Smith "
4096-bit RSA key, ID 0C9ED241, created 2014-10-25
```

```
Please select what kind of key you want:
   (3) DSA (sign only)
   (4) RSA (sign only)
   (5) Elgamal (encrypt only)
   (6) RSA (encrypt only)
```

```
Your selection? 6

RSA keys may be between 1024 and 4096 bits long.

What keysize do you want? (2048) 4096

Requested keysize is 4096 bits
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

Really create? (y/N) y

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: NOTE: you should run 'diskperf -y' to enable the disk statistics
gpg: NOTE: you should run 'diskperf -y' to enable the disk statistics
gpg: NOTE: you should run 'diskperf -y' to enable the disk statistics
gpg: NOTE: you should run 'diskperf -y' to enable the disk statistics
..+++++
.....+++++

pub 4096R/0C9ED241 created: 2014-10-25 expires: never    usage: SC
      trust: ultimate    validity: ultimate
sub 4096R/77E904F9 created: 2014-10-25 expires: never    usage: E
[ultimate] (1). John Smith

Command> save
```

4. Execute the following commands:
    - gpg --output mycompany-prod-pub.asc --armor --export "John Smith <jsmith@myportal.net>"
    - gpg --output mycompany-prod-pri.asc --armor --export-secret-keys "John Smith <jsmith@myportal.net>"

PGP keys are stored in one of the following directories:

- unipay/resources/PGP
    - unibroker/resources/PGP