etsi_qkd_014_client

Release 0.9.0

Yoann Piétri

Nov 27, 2022

QUICKSTART GUIDE

`etsi_qkd_014_client` is a library that implements the ETSI GS QKD 014 specifications and easily allows you to interact with a QKD server implementing the server side of these specifications.

Here is an example for a simple key exchange using the client

```
from etsi_qkd_014_client import QKD014Client

client_alice = QKD014Client(
   "192.168.10.101",
   "clientCert.pem",
   "clientKey.pem",
   "rootCA.pem",
   force_insecure=True
)

client_bob = QKD014Client(
   "192.168.10.106",
   "clientBobCert.pem",
   "clientBobKey.pem",
   "rootCA.pem",
   force_insecure=True
)

code, data = client_alice.get_key("SAEBOB") # By default, this request one key of 256␣
↪bits

print(code) # 200

print(data)

# Key id : 8c3c8d07-4827-47b7-a61b-db9b95f01cb9
# Key : H7prwEw/MNN8AcpMnSUyt2fIXguhofof3qLt2O9uc5U=

if code == 200:
   key_id = data.keys[0].key_id
   key_alice = data.keys[0].key

   code, data = client_bob.get_key_with_key_IDs("SAEALICE", [key_id])

   print(code) # 200

   print(data)

   # Key id : 8c3c8d07-4827-47b7-a61b-db9b95f01cb9
   # Key : H7prwEw/MNN8AcpMnSUyt2fIXguhofof3qLt2O9uc5U=

   key_bob = data.keys[0].key

   print(key_alice == key_bob) # True
```

Features

- Connect to a QKD server, using a custom CA, key and cert;
- Retrieve status of the QKD server with the `get_status()` command;

- Retrieve a secure key using the `get_key()` command;
- Retrieve a secure key knowing the key's ID using the `get_key_with_key_IDs()` command.

CHAPTER

ONE

INSTALLATION

The first part of any library of software is to get it installed. The easiest way to install the software is to use the PyPi repo.

1.1 Installation using PyPi

You can install the package from PyPi using the following command:

```
pip install esti-qkd-014-client
```

1.2 Getting the source code

The code is available on GitHub.

You can clone the repository:

```
git clone git://github.com/nanoy42/etsi-qkd-014-client.git
```

or download the tarball:

```
curl -OL https://github.com/nanoy42/etsi-qkd-014-client/tarball/main
```

Then you can add the module to your source code or install it with pip:

```
cd etsi-qkd-014-client
pip install .
```

CHAPTER

TWO

QUICKSTART

Once the software is installed, you can get the status of your KME with :

```
 from etsi_qkd_014_client import QKD014Client

 client = QKD014Client(
   "192.168.1.1",
   "clientCert.pem",
   "clientKey.pem",
   "rootCA.pem",
   force_insecure=True
)

_, rep = client.get_status("SAEBOB")

print(rep)
```

Then get a key with

```
_, rep = client.get_key("SAEBOB")

 print(rep.keys[0].key)
 key_id = rep.keys[0].key_id
```

and finally get the key on the other KME with

```
 client_b = QKD014Client(
   "192.168.1.2",
   "clientBobCert.pem",
   "clientBobKey.pem",
   "rootCA.pem",
   force_insecure=True
)

_, rep = client_b.get_key_with_key_IDs("SAEALICE", [key_id])

print(rep.keys[0].key)
```

If you want to know more, you can head to the next section that will go more in depth about the client.

CHAPTER

THREE

QKD014CLIENT

The complete API reference of the client can be found here : `QKD014Client`.

The `QKD014Client` is the main class that you will use with this module.

It is the one that implements the three main methods described in the specifications :

- `get_status()`
- `get_key()`
- `get_key_with_key_IDs()`

3.1 Initialization of the client

To initiate the client you need 4 required parameters :

- `kme_hostname` : ip address of url of the API;
- `cert_path` : the path of the certificate of the client, signed by the root client CA;
- `key_path` : the path of the key associated to the certificate;
- `ca_path` : the path of the root CA used to sign the server's certificates.

and 1 optional parameter

- `force_insecure` : wether to verify or not the certificate of the server (if `force_insecure` is `True`, the certificate of the server will not be checked).

The full documentation of this function is `etsi_qkd_014_client.client.QKD014Client.__init__()`

An example of initialization is given below :

```
from etsi_qkd_014_client import QKD014Client

client_alice = QKD014Client(
    kme_hostname = "192.168.10.101",
    cert_path = "clientCert.pem",
    key_path = "clientKey.pem",
    ca_path = "rootCA.pem",
    force_insecure = True
 )
```

that can also be called without the parameters name :

```
from etsi_qkd_014_client import QKD014Client

client_alice = QKD014Client(
  "192.168.10.101",
  "clientCert.pem",
  "clientKey.pem",
  "rootCA.pem",
  True
)
```

3.2 Using the client

3.3 Return values of the client

If you use one of the three public methods, they will return a tuple composed of an int that is the response code and an instance a class inheriting from `QKD014Data` that is the response data.

3.3.1 Response code

The response code is the same as the HTTP response codes.

Only a fraction of them can be returned according to the specifications and here they are :

Code	Description
200	Success
400	Bad request format
401	Unauthorized
503	Error on server side

In general, the code 200 will then be returned with a Data class corresponding to what was requested, and if another code is returned, it usually comes from with a `DataError` instance, that may hold additionnal information on the error.

3.3.2 Data

The main class `QKD014Data` is an abstract class from which inherits 6 classes :

- `DataStatus`;
- `DataKeyRequest`;
- `DataKey`;
- `DataKeyContainer`;
- `DataKeyIDs`;
- `DataError`.

The public methods will however return only one of the three following class :

- `DataStatus` for the `get_status()` method;
- `DataKeyContainer` for the `get_key()` and `get_key_with_key_IDs()` methods;

- `DataError` in case of an error while calling one of the three methods.

You will also, however, deal with the `DataKey` class since the `DataKeyContainer` instance will contain a list of those.

Warning: The `DataKeyRequest` and `DataKeyIDs` are helping classes for the requests, and should not be used as such by the end user.

DataError

An instance of the `DataError` class is returned in case an error occurred during the request. As described in the specifications it holds two fields :

- `message` that is the error message returned by the server;
- `details` that is an optional list of key/value pairs (`dict`) containing additional information on the error.

Note: It does not hold the response code, that is returned individually from the data.

DataStatus

An instance of the `DataStatus` class is returned in case of a successful request to `get_status()`. There are several attributes accessible :

- `source_kme_id`: KME ID of the KME;
- `target_kme_id`: KME ID of the target KME;
- `master_sae_id`: SAE ID of the calling master SAE;
- `slave_sae_id`: SAE ID of the specified slave SAE;
- `key_size`: Default size of key the KME can deliver to the SAE (in bit);
- `stored_key_count`: Number of stored keys KME can deliver to the SAE;
- `max_key_count`: Maximum number of stored_key_count;
- `max_key_per_request`: Maximum number of keys per request;
- `max_key_size`: Maximum size of key the KME can deliver to the SAE (in bit);
- `min_key_size`: Minimum size of key the KME can deliver to the SAE (in bit);
- `max_sae_id_count`: Maximum number of additional_slave_sae_ids the KME allows. “0” when the KME does not support key multicast;
- `status_extension`: (Option) for future use.

DataKeyContainer

An instance of the `DataKeyContainer` class is returned in case of a successful request to `get_key()` or `get_key_with_key_IDs()`. There are several attributes accessible :

- `keys`: Array of keys. The number of keys is specified by the "number" parameter in "Get key". If not specified, the default number of keys is 1. Each element in this array is an instance of the class `DataKey`;
- `key_container_extension`: (Option) for future use.

DataKey

Instances of `DataKey` are contained in the `keys` attribute of an instance of `DataKeyContainer`. There are several attributes accessible :

- `key_id`: ID of the key: UUID format (example: "550e8400-e29b-41d4-a716-446655440000");
- `key_id_extension`: (Option) for future use;
- `key`: Key data encoded by base64 [7]. The key size is specified by the "size" parameter in "Get key". If not specified, the "key_size" value in Status data model is used as the default size.
- `key_extension`: (Option) for future use.

CHAPTER

FOUR

A FULL EXAMPLE

4.1 Initiate client

```
from etsi_qkd_014_client import QKD014Client

client = QKD014Client(
    "192.168.10.101", # KME hosy
    "certificatesMatteo/encAlice/clientCert.pem", # Certificate of the client signed by␣
↪client's CA
    "certificatesMatteo/encAlice/clientKey.pem", # Private key associated to to the␣
↪certificate
    "certificatesMatteo/CA/rootCA.pem", # Server's root CA to verify the certificate of␣
↪the server
    force_insecure=True # This will force non verification of the certificate of server
)

print(client)

# QKD014Client
    #     KME : 192.168.10.101
    #     Client certificate : certificatesMatteo/encAlice/clientCert.pem
    #     Client key : certificatesMatteo/encAlice/clientKey.pem
    #     Root CA for server : certificatesMatteo/CA/rootCA.pem
    #     Force insecure : True
```

4.2 Get status

```
from etsi_qkd_014_client import QKD014Client

client = QKD014Client(
    "192.168.10.101",
    "certificatesMatteo/encAlice/clientCert.pem",
    "certificatesMatteo/encAlice/clientKey.pem",
    "certificatesMatteo/CA/rootCA.pem",
    force_insecure=True
)

code, data = client.get_status("SAEBOB")
```

(continues on next page)

(continued from previous page)

```
print(code) # 200

print(data)

# source_kme_id : KMSALICE
# target_kme_id : KMSBOB
# master_sae_id : SAEALICE
# slave_sae_id : SAEBOB
# key_size : 256
# stored_key_count : 86
# max_key_count : 100
# max_key_per_request : 1
# max_key_size : 256
# min_key_size : 128
# max_sae_id_count : 0
```

4.3 Key exchange

```
import base64
from etsi_qkd_014_client import QKD014Client

client_alice = QKD014Client(
    "192.168.10.101",
    "clientCert.pem",
    "clientKey.pem",
    "rootCA.pem",
    force_insecure=True
)

client_bob = QKD014Client(
    "192.168.10.106",
    "clientBobCert.pem",
    "clientBobKey.pem",
    "rootCA.pem",
    force_insecure=True
)

code, data = client_alice.get_key("SAEBOB") # By default, this request one key of 256␣
↪bits

print(code) # 200

print(data)

# Key id : 8c3c8d07-4827-47b7-a61b-db9b95f01cb9
# Key : H7prwEw/MNN8AcpMnSUyt2fIXguhofof3qLt2O9uc5U=

if code == 200:
    key_id = data.keys[0].key_id
```

(continues on next page)

(continued from previous page)

```
    key_alice = data.keys[0].key

    code, data = client_bob.get_key_with_key_IDs("SAEALICE", [key_id])

    print(code) # 200

    print(data)

    # Key id : 8c3c8d07-4827-47b7-a61b-db9b95f01cb9
    # Key : H7prwEw/MNN8AcpMnSUyt2fIXguhofof3qLt2O9uc5U=

    key_bob = data.keys[0].key

    print(key_alice == key_bob) # True

    print(key_alice) # H7prwEw/MNN8AcpMnSUyt2fIXguhofof3qLt2O9uc5U=

    print(f"".join(["{:08b}".format(x) for x in base64.b64decode(key_alice)]))

    #␣
↪00011111101110100110101111000000010011000011111100110000110100110111110000000000111001010010011001001
```

CHAPTER

FIVE

COMMAND LINE INTERFACE

The package is shipped with a command line interface, that is used to test a QKD server implementing the QKD specifications.

Warning: This command line interface is intended for testing purposes only ! It should not be used for production.

5.1 Accessing the command line interface

The command line can be accessed with the *cli.py* file and also with qkd014-client command line tool installed with the software.

5.2 Passing the required arguments

The required arguments, independently of which command is used, are :

- hostname (of the server implementing the QKD 014 specifications);
- cert : certificate file path to use;
- key : private key path associated with the certificate;
- ca : the root CA path that signed the server's certificate;
- force : whether to force insecure connections or not.

These parameters can be given either directly with the command line or using a configuration file.

5.2.1 Through the command line

Here is the way to give all the parameters :

- hostname can be given with `--hostname HOSTNAME` or `-H HOSTNAME`;
- cert can be given with `--cert CERT` or `-c CERT`;
- key can be given with `--key KEY` or `-k KEY`;
- ca can be given with `--ca CA` or `-r CA`;
- by default force is false. You can set it to true with `--force` or `-f`.

5.2.2 Through a configuration file

As an alternative to pass all those arguments on the command line, you can also create a configuration file that looks like

```
[etsi_qkd_014_client]
hostname = 192.168.1.1
cert = cert.pem
key = key.pem
ca = ca.pem
force = no
```

Then you should call the script using the `--config` option:

```
qkd014-client --config config.ini get_status
```

5.2.3 Precedence

Configuration file takes priority over the command line arguments. That means that, for instance if the `--config` is given with a wrong configuration file, the script will raise an exception even if all the other parameters were given to the command line.

You cannot mix command line arguments and configuration file.

5.3 Commands

There are 3 different subcommands :

- `get_status` command to get the status of the QKD server. This require the SAE ID of the slave SAE.
- `get_key` command to get one (or more) key(s). This require at least one additional parameter: the SAE ID of the slave SAE.
- `get_key_with_ID` command to get one (or more) key(s), knowing their ID. This require at least two additional parameters: the SAE ID of the slave SAE and the list of the ID(s) of the key(s).

5.3.1 Get status

You can get the status of SAEBOB with:

```
qkd014-client -H 192.168.10.101 -c clientCert.pem -k clientKey.pem -r rootCA.pem -f get_
↪status SAEBOB
```

5.3.2 Get key

You can get a key with:

```
qkd014-client -H 192.168.10.101 -c clientCert.pem -k clientKey.pem -r rootCA.pem -f get_
↪key SAEBOB
```

5.3.3 Get key with ID

You can get a key with the ID with:

```
qkd014-client -H 192.168.10.101 -c clientCert.pem -k clientKey.pem -r rootCA.pem -f get_
↪key_with_id SAEALICE 8c3c8d07-4827-47b7-a61b-db9b95f01cb9
```

CHAPTER

SIX

CLIENT

CHAPTER

SEVEN

DATA

This page presents the data structures that are used by the client. They mainly correspond to the ones described in section 6 “Data Format Specifications” of the specifications.

CHAPTER

EIGHT

CLI

CHAPTER

NINE

CONTRIBUTING

Contributions are welcomed.

You can see the issues and open new one here : https://github.com/nanoy42/etsi-qkd-014-client/issues.

Pull requests are welcomed if they follow the etsi standard. You should fork the repo, add your modification and create a pull request here : https://github.com/nanoy42/etsi-qkd-014-client/pulls.

CHAPTER

TEN

CITATION

If you use this software, please consider citation. Here is the biblatex entry :

```
@software{etsi_qkd_014_client,
author = {{Yoann Piétri}},
title = {ETSI QKD 014 client},
url = {https://github.com/nanoy42/etsi_qkd_014_client},
version = {0.1.1},
date = {2022-06-02},
}
```

If `@software` is not available, you can also use

```
@misc{etsi_qkd_014_client,
author = {{Yoann Piétri}},
title = {ETSI QKD 014 client},
url = {https://github.com/nanoy42/etsi_qkd_014_client},
date = {2022-06-02},
}
```

Plain text citation

> Yoann Piétri. (2022). ETSI QKD 014 client (0.1.1) [Computer software]. https://github.com/nanoy42/etsi_qkd_014_client

CHAPTER

ELEVEN

LICENSE

This software is distributed under the GNU Lesser General Public License v3 (GNU LGPLv3). A copy of the complete text of the license is included with the software (the LICENSE file).

The header of the license can be found below:

```
Copyright (C) 2022 Yoann Piétri
Copyright (C) 2022 LIP6 - Sorbonne Université

etsi-qkd-14-client is free software: you can redistribute it and/or modify
it under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

etsi-qkd-14-client is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License
along with etsi-qkd-14-client. If not, see <http://www.gnu.org/licenses/>.
```