

Quantum Blockchain's „QRNGBase” – Entropy-as-a-Service API

Summary

In this communication we describe Quantum Blockchains implementation of Entropy-as-a-Service Web API which uses hardware Quantum Random Number Generator to deliver streams of random bits with high entropy for various cryptographic applications in general and for Blockchain applications in particular. We have built the QRNGBase using ID Quantique Quantis QRNG hardware generator¹.

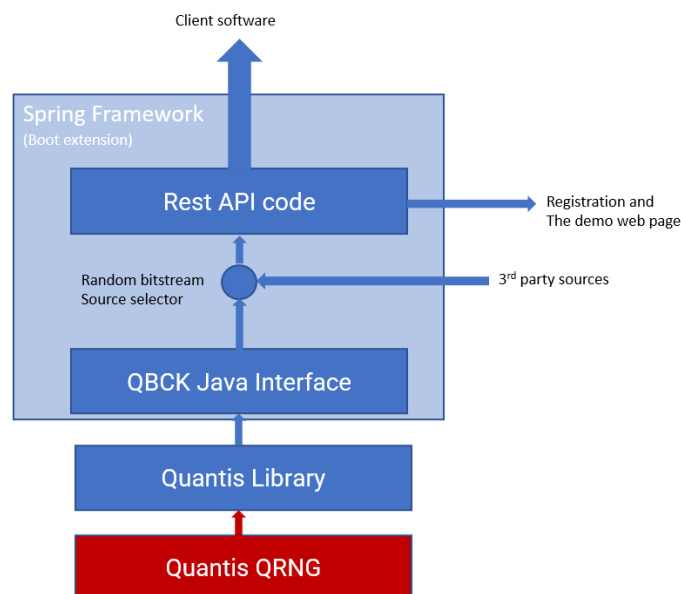
On the software side we have built multi-layered software stack, that transforms the random bitstream generated by the hardware device into multiple formats of data structures and makes them available via well defined REST based WEB API. We also made a demonstration page on our website where the generated numbers can drive graphical/visual² and numerical³ demonstrations.

It must be stressed that QRNGBase is production ready software. All the mechanism of typical SaaS services, like users' registration, access token generation and SSL protection of random streams have already been implemented.

The unique feature of our solution is the ability to deliver the random numbers based on another sources of entropy made available by third party providers via their webservices. We have initially integrated services of QNU Labs and RealRandom companies into our service.

QRNGBase architecture

The service architecture is depicted in the following diagram:



¹ <https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator/>

² <https://www.quantumblockchains.io/current-services/qrng/>

³ <https://www.quantumblockchains.io/current-services/qrng-numeric/>

The QRNGBase layers' description

We use bottom-up approach to describe the layers:

- 1) Quantis QRNG – the USB interfaced Quantum Random Number Generator based on photon detection manufactured by ID Quantique (See Appendix A for specification)
- 2) Quantis Library – a low-level, C++ based Linux library created by the device manufacturer with Java wrapper
- 3) QBCK Java Interface – the layer written by our company that provides bridge between Quantis layers (hardware and software) and the API generation code. Written in Java as a Spring (Boot) framework application.
- 4) Random bitstream software selector – Java code written by our company that allows for selection of the bitstream source: from Quantis hardware/software and from external sources (via their respective APIs)
- 5) Rest API code – a Java code that parses API requests and returns responses (using SSL secured HTTP protocol)
- 6) The webserver is Spring Boot embedded Tomcat server.
- 7) Registration and demo web pages – the client front-end for registration and simple online generation of random numbers.

The entire stack runs out if bare-metal server with Linux Ubuntu OS and is hosted in a physical data-center.

The QRNGBase API calls

The API assumes use of HTTP GET call pattern with the following functions:

bytes_range - returns various numerical types of random numbers in the desired range (min, max). The types the client can request are: int (or short), long – for integer types and float and double for floating-point types.

byte_types - returns random bytes in various representations: bigint, hex, bin, base64, base56.

bin_file - generates a file with random binary content and allows to download the file.

The API calls' parameters

bytes_range:

```
var APIKey = "<the client API key>"; // THE AUTHORIZATION KEY
var provider = "qbck"; // QRNG provider code
var type = "short"; // "int" or "short" or "long" or "float" or "double"
var size = 10; // the number of random numbers generated
var min = 0; // min value of the numbers
```

```

var max      = 1000;      // max value of the numbers
var file     = ""        // if empty, the result is returned in JSON format.
                                // if given the result goes to the file

//=====

```

byte_types:

```

var APIKey   = "<the client API key>"; // THE AUTHORIZATION KEY
var provider = "qbck"; // QRNG provider code
var type     = "hex";   // "bigint" or "hex" or "bin" or "base64" or "base56"
var size     = 10;     // the number of random numbers generated
var length   = 16;     // the size of the random number in bytes
                                // (for bigint, hex and bin) or
                                // characters (for base64 and base56)
var file     = ""      // if empty, the result is returned in JSON format.
                                // if given the result goes to the file

//=====

```

bin_file:

```

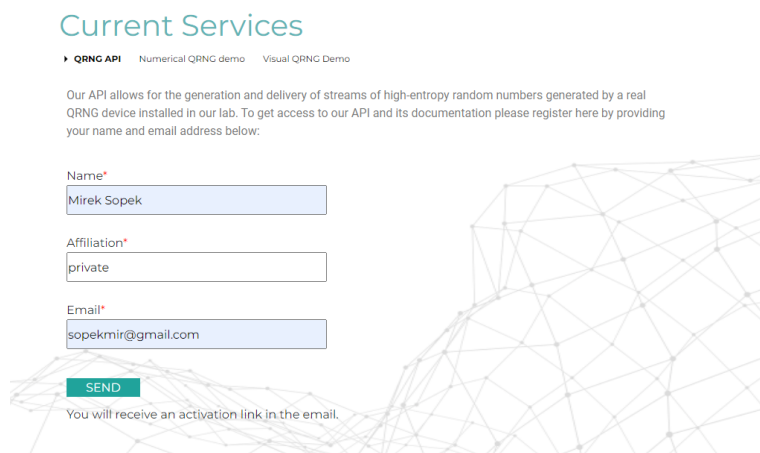
var APIKey   = "<the client API key>"; // THE AUTHORIZATION KEY
var provider = "qbck"; // QRNG provider code
var length   = 32;     // the file size in bytes
var file     = "random.bin" // if empty, the result is stored in the default
                                // (random.bin) file.
                                // If given the result goes to the file

//=====

```

The registration and the demo pages

The registration page (<https://www.quantumblockchains.io/current-services/api/>) allows the clients to subscribe to the services:



Current Services

[QRNG API](#)
 [Numerical QRNG demo](#)
 [Visual QRNG Demo](#)

Our API allows for the generation and delivery of streams of high-entropy random numbers generated by a real QRNG device installed in our lab. To get access to our API and its documentation please register here by providing your name and email address below:

Name*

Affiliation*

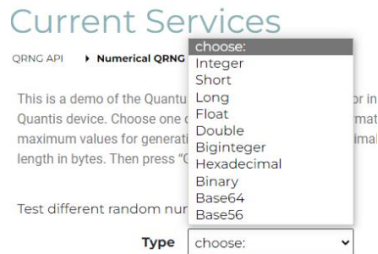
Email*

You will receive an activation link in the email.

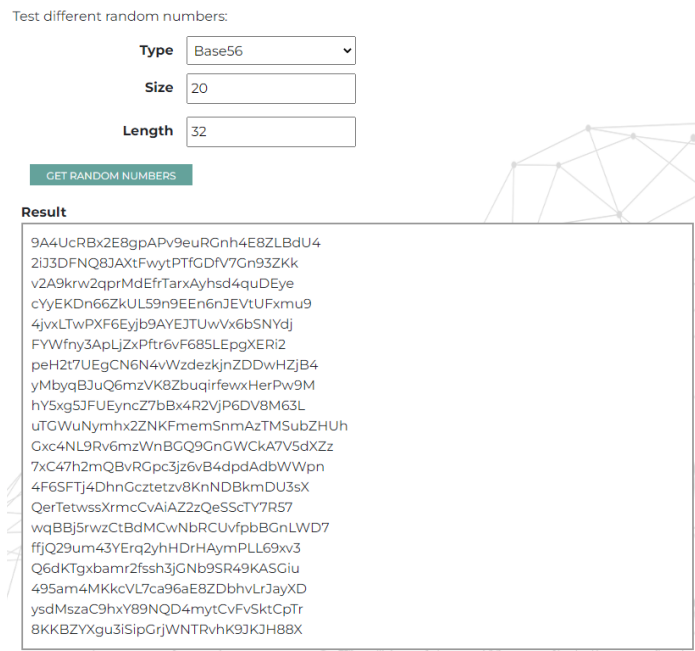
After successful registration the client gets the mail with documentation, Postman collection description for easier tests, and the link which when followed generates the API authorization key.

There are two demo pages available:

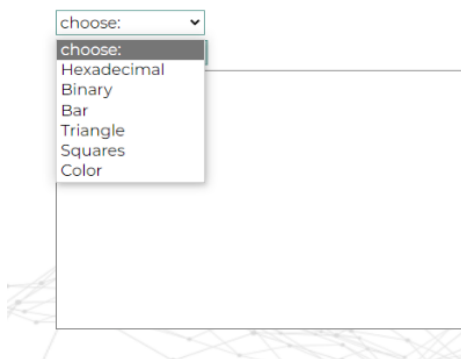
- Numerical QRNG demo (<https://www.quantumblockchains.io/current-services/qrng/>) which allows for the following choices:



and returns (for Base56):



- Visual QRNG Demo (<https://www.quantumblockchains.io/current-services/qrng/>) which allows for the following choices:



and returns (for “Squares”):



The 3rd party sources available

Currently we have successfully tested our mechanisms with two existing providers:

- QNU Labs (<https://www.qnulabs.com/tropos-quantum-random-number-generator/>)
- RealRandom (<https://realrandom.co/apis-overview/>)

Both provide good source of random numbers with high entropy.

We are also in advanced talks with SeQure company⁴ to include their random sources into our API. Our analysis revealed that SeQure’s QRNG which offers self-testing QRNG is possibly superior to other solutions.

⁴ <https://www.sequire-quantum.com/>

Appendix A – Quantis QRNG

Quantis QRNG (USB interfaced):



QUANTIS USB



Quantis-USB-4M

GENERAL SPECIFICATIONS

1: Hardware bit rate prior to randomness extraction

Random bit rate¹	4 Mbit/s ± 10% (Quantis-USB-4M)
Thermal noise contribution	< 1% (Fraction of random bits arising from thermal noise)
Storage temperature	- 25 to + 85°C
Dimensions	61 mm x 31 mm x 114 mm
USB specification	2.0
Requirement	PC with available USB connector
Power	Via USB port

QUANTIS Principle



Based on Quantum Physics :

Photons - light particles - are sent one by one onto a semi-transparent mirror and detected. The exclusive events (reflection - transmission) are associated to « 0 » - « 1 » bit values.

