drand Python Documentation

Release 0.1.0.dev3

Sylvain Bellemare

Mar 13, 2020

Contents

Python client to query a drand network for publicly verifiable, unbiased, and unpredictable random values.

To learn more about drand see drand's documentation.

WARNING: This software is currently only, strictly, and purely for experimental purposes. It was developed for prototyping and experimenting with the drand network, which is itself still experimental!

IMPORTANT: Currently only works with the drand server code from the `master` branch (as of March 8, 2020). To query the drand test network (e.g.: League of Entropy) using Python you may try drb-client.

CHAPTER 1

Install

```
$ pip install drand
```

CHAPTER 2

Usage

2.1 Prerequisite: Run a local drand network

First, run a local drand network. See devnet/README.md for more details.

```
$ cd devnet
$ ./run.sh
```

Get the addresses of the drand servers

```
from drand.utils import get_addresses_from_group_file

group_file = 'devnet/data/group.toml'
addresses = get_addresses_from_group_file(group_file)
```

```
>>> addresses
['172.15.238.2:8084',
 '172.15.238.3:8081',
 '172.15.238.4:8080',
 '172.15.238.6:8082',
 '172.15.238.5:8083']
```

2.2 Query a drand server

```
import drand
```

2.2.1 Get the public key of the network (/api/info/distkey)

Each node has a public share of this group key.

```
distkey = await drand.get_distkey(addresses[0], tls=False)
```

```
>>> distkey

↪'9509e2c2a5d04776bedce40839341375c89aa34a0372a1db273f562d89050b4ae54a76a276a26580166b0cd91e63f909
↪'
```

2.2.2 Get and verify a random value (/api/public)

The verification means verifying that the "randomness" value is the hash of the signature, and that the signature is valid for the public key (distkey) and the the message (round + previous)

```
res = await drand.get_and_verify(
    addresses[3], distkey=distkey, tls=False,
)
```

```
>>> res
{'round': 73,
 'previous':
↪'b894ccc3859d1fb6d2ce6722b7195d359fbe6b0a387a3693e539e4957f1c69025936919fff3bd89a303ccfbcb929aae10e
↪',
 'signature':
↪'817254f9267e5345f5160a794ad5ffca0a9a2295cbfedc8c3d19215f91c8ccd07faa8354564d18159905477757c21f8a05
↪',
 'randomness': '66c3554bc0927a4ccbfdd73856071be792e3ddec7c27193d2f2f4d482c78b6b2'}
```

Get a random value for round 5

```
res = await drand.get_and_verify(
    addresses[3], distkey=distkey, tls=False, round_=5
)
```

```
>>> res
{'round': 5,
 'previous':
↪'aab94951afa626c26af5e08baa111fb98b1f5300556dc472f5e976a1ca4ccb074ecb7778cf18e08272fb40e1421a630914
↪',
 'signature':
↪'ad3e4f0bf0ef93c2ced95c12e1e7b5d0adbc4791e5592a83ce6119e0b610b7de40786e639861aa62df9d3a01b0ac50f90c
↪',
 'randomness': 'baee3fd77cd09349325794f766c0c81c887987907ec2834ac09a8a46c2193747'}
```

Get random values for a range of rounds

```
import asyncio

from aiohttp import ClientSession

async def get_rands(rounds):
    async with ClientSession() as session:
        tasks = []
        for r in rounds:
            tasks.append(
                drand.get_and_verify(
```

(continues on next page)

(continued from previous page)

```
                    addresses[4],
                    distkey=distkey,
                    session=session,
                    tls=False,
                    round_=r,
                )
            )
        rands = await asyncio.gather(*tasks)
    return rands
```

```
>>> asyncio.run(get_rands(range(2, 5)))
[{'round': 2,
  'previous':
↪'b816229db70d3d7ab727bf0dc8ae3de27c354b066d5d931d3b6fb14d2fcf2433cd72f0271a9c47e7448de7c9589de2250c
↪',
  'signature':
↪'a515fe873dc18810d3aa446614786aa63567930f888c82b1edf66ea1e0f604c46948863dc349320219eba7d11a7848131
↪',
  'randomness': '185963dba81d25158bb60bc0bc16823b7687a87cca739a6a9e4a2bccac16c5f0'},
 {'round': 3,
  'previous':
↪'a515fe873dc18810d3aa446614786aa63567930f888c82b1edf66ea1e0f604c46948863dc349320219eba7d11a7848131
↪',
  'signature':
↪'81d3a98e63e8480d61e64ef7126dea5f83cc98303d43c66221f15edab8dc4e02d7c229a645f107ee76e0de11673569810
↪',
  'randomness': '0b7d6c4a465b4cd6099f4a888ea355c2173a8108ad749a7790c64592a9c2ee9f'},
 {'round': 4,
  'previous':
↪'81d3a98e63e8480d61e64ef7126dea5f83cc98303d43c66221f15edab8dc4e02d7c229a645f107ee76e0de11673569810
↪',
  'signature':
↪'aab94951afa626c26af5e08baa111fb98b1f5300556dc472f5e976a1ca4ccb074ecb7778cf18e08272fb40e1421a63091
↪',
  'randomness': '2dcc3e4894c91d092cdbcbe6daf777c5cbe2e6948cf8a18693009762273d52aa'}]
```

CHAPTER 3

Acknowledgments

The initial code interface for this package was based on the JavaScript client drandjs.

The `devnet` directory under the root of the repo was taken from the demo directory under the drand/drand repository, tree with commit hash a40dc25e1aec6822a79c72b4aaca12e65c700f01. The code was brought over using git-filter-repo in order to preserve the commit history.

The original boilerplate for this package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

Thanks to IC3 (The Initiative For Cryptocurrencies & Contracts) for supporting this work.

CHAPTER 4

Reminder & Future Work

This software is currently only, strictly, and purely for experimental purposes. It was developed for prototyping and experimenting with the drand network, which is itself still experimental!

The Github issue tracker will be used to plan and manage future work.

4.1 History

4.1.1 0.1.0.dev0 (2020-03-07)

- Made planning release on PyPI.

4.1.2 0.1.0.dev1 (2020-03-08)

- Added code.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search