sproof-js Documentation

Release 1.0

Clemens Brunner, Fabian Knirsch

Sep 06, 2019

Contents

sproof is a decentralized application that uses a blockchain for managing digital documents. sproof is designed for scalability and processes a number of events (e.g., register a document, revoke a document) within a single transaction. sproof makes the process of digital document verification easy and fast and provides an open, privacy-preserving protocol following the principle “developers first”. An easy-to-integrate API makes developing applications fast and allows for digitalizing existing businesses, as well as for building all-new business models.

Contents:

CHAPTER 1

First steps

sproof is a decentralized open source protocol for registering data and documents in a public blockchain. To use sproof we provide clients in different programming languages. Currently we support javascript, Java and C are coming soon.

1.1 Create a node project

1.1.1 Install node and npm

You need to create a node project to use the `js-sproof-client`. Before creating a project you need to install the latest version of `npm` and `nodejs`.

Ubuntu

Install:

```
sudo apt update
sudo apt install nodejs npm
```

Windows

Coming soon. Feel free to edit the docs on github.

Mac OS

Coming soon. Feel free to edit the docs on github.

1.1.2 Create the project structure

Create a new folder for your project and open it with a terminal. Run `npm init` and follow the instructions. After that install the `js-sproof-client` with:

```
npm install --save js-sproof-client
```

Create a new file called `index.js` and `config.js` in your project folder.

1.2 Create an account

We provide two different methods to create your unique sproof account, which is basically a public-private key pair.

1.2.1 Standard

If you want to use your own Ether (Cryptocurrencies) to paid for your transaction you need to create your public-private key pair with the following code:

```
const { Sproof, Registration }  = require('js-sproof-client');
let sproof = new Sproof({
  uri: 'https://api.sproof.io/',
  chainId: '3',
  chain: 'ethereum',
  version: '0.42'
});
let credentials = sproof.newAccount();
console.log(credentials)
```

After that you need to request Ether. Currently sproof lives in the Ropsten Testnet. To get Ether you need to enter your address on te website.

Once your account has Ether you can register your stuff with the following command:

```
sproof.commit(callback)
```

1.2.2 Premium

If you don't want to request Ether you can use our premium api, where sproof acts as a proxy and forwards your secure data and your locally created signature to the blockchain. To use this service you need to create your account and a sproof profile with your **'webapp <https://app.sproof.io'_**.

Once your profile is created you can download your `sproof-code`, with 10 free uploads attached. If you need more uploads feel free to contact team@sproof.io.

Your sproof code is a mnemonic which contains 12 randomly chosen words.

Note: sproof does not stores your sproof-core. In case that you lose your `sproof-code` we cannot recover it.

1.3 Create a config

Add the following code to your `` `config.js` `` file and replace the `sproofCode`:

```
let config = {
    uri: 'https://api.sproof.io/',
    credentials: {
        sproofCode: 'word1 word2 word3 word4 word5 word6 word7 word8 word9 word10␣
↪word11 word12',
    },
    chainId: '3',
    chain: 'ethereum',
```

(continues on next page)

(continued from previous page)

```
    version: '0.42'
};
module.exports = config;
```

1.4 Register your first document

Take a look at the code examples on github or in the examples section.

CHAPTER 2

Examples

In the following you will find some examples to integrate and use sproof.

2.1 Profiles

Create a sproof profile

```
const { Sproof }  = require('js-sproof-client');

let sproof = new Sproof({
  uri: 'https://api.sproof.io/',
  chainId: '3',
  chain: 'ethereum',
  version: '0.42'
});

let credentials = sproof.newAccount();

let registerProfileEvent = sproof.registerProfile({
  name: 'new sproof account',
  profileText: 'Sproof Test Account',
  image: 'Qma34dB4B4N4eS5ibBkwtjTSTNCRdJrVY6E25DFuFuU8Sd'
});

sproof.commitPremium((err, res) => {
  if (err) console.error(err);
  else console.log(res);
});
```

2.2 Document

Create a profile and register a document

```
const { Sproof, Registration }  = require('js-sproof-client');

let sproof = new Sproof({
  uri: 'https://api.sproof.io/',
  chainId: '3',
  chain: 'ethereum',
  version: '0.42'
});

let credentials = sproof.newAccount();

let registerProfileEvent = sproof.registerProfile({
  name: 'new sproof account 1',
  profileText: 'Sproof Test Account',
  image: 'Qma34dB4B4N4eS5ibBkwtjTSTNCRdJrVY6E25DFuFuU8Sd',
  homepage: 'www.test.at'
});

let documentHash = '0xf1b1c24a69c4c726c8b1ec42ed924b7305f3eb53949fc2f64dd1ef7d0ee9b0e5
↪';
// documentHash = sproof.getHash(>>string or buffer <<<);

let registration  = new Registration({
  documentHash,
  validFrom: undefined, //unix timestamp
  validUntil: undefined, //unix timestamp
});

sproof.registerDocument(registration);


sproof.commitPremium((err, res) => {
  if (err) console.error(err);
  else console.log(res);
});
```

2.3 Publish and register a local Pdf File

Upload a Pdf to IPFS and secure it with the blockchain

```
const { Sproof, Registration }  = require('js-sproof-client');
const config = require ('./config/config_issuer');
const fs = require('fs');

let sproof = new Sproof(config)


let data = fs.readFileSync('./example.pdf');

sproof.uploadFile(data, (err,res) => { //upload file to ipfs
  if (res) {
```

(continues on next page)

(continued from previous page)

```
    let documentHash = sproof.getHash(data); //calculate hash of the file

    let registration  = new Registration({
      documentHash,
      name: 'mytestpdf',
      locationHash: res.hash, //add ipfs location hash
      validFrom: undefined, //unix timestamp
      validUntil: undefined, //unix timestamp
    });

    sproof.registerDocument(registration);

    sproof.commitPremium((err, res) => {
      if (err) console.error(err);
      else console.log(res);
    });
  }else
    console.error(err)
});
```

CHAPTER 3

Events

The sproof-core module is a state machine which lives on top of a blockchain. The state transitions is triggered by socalled events. An issuer can add the hash reference of a list of events in a transaction to the blockchain, the raw-data is stored in IPFS. In this section we describe all currently available events.

3.1 Profile

A profile schema has the following form:

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Profile",
  "description": "A sproof profile",
  "type": "object",
  "properties": {
    "name": {
      "description": "The name of a user",
      "type": "string",
      "minLength" : 3,
    },
    profileText: {
      description: "Additional text for the profile",
      type: "string",
      "title": "Add your profile text",
      "maxLength": 500,
      "validationMessage": "Don't be greedy!"
    },
    "image": {
      "description": "A base64 encoded string of a image",
      "type": "string",
      "minLength": 1
    },
    website: {
```

(continues on next page)

(continued from previous page)

```
      description: "The website of the user. To increase the trust, the user has to␣
↪upload a file on rootdomain/sproof.html",
      type: "string",
      pattern : "^((http\\:\\/\\/|https\\:\\/\\/)([a-z0-9][a-z0-9\\-]*\\.)+[a-z0-9][a-
↪z0-9\\-]*$)?$"
    },
    "socialMedia": {
      "type": "array",
      "description": "Array of social media post and user account. This is to␣
↪increase the trust of the users account",
      "items": {
        "type": "object",
        "properties": {
          "userId": {
            "description": 'The unique userid of a social media account.',
            "type": "string"
          },
          "messageId": {
            "description": "The unique message_id which contrains the public key of␣
↪the user",
            "type": "string"
          },
          "platform": {
            "type": "string",
            "description": "Name of the social media platform"
          }
        },
        "required": ["userId", "messageId", "platform"]
      },
      "minItems": 0,
      "uniqueItems": true
    },
  },
  "required" : ["name"]
};
```

3.1.1 Register

To register a profile (issuer) the `PROFILE_REGISTER` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Profile register",
  "description": "Register sproof profile event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["PROFILE_REGISTER"]
    },
    data: $ProfileSchema
  },
  "required" : ['eventType', 'data']
}
```

3.1.2 Update

To update a profile (issuer) the `PROFILE_UPDATE` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Update Profile",
  "description": "Update profile sproof event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["PROFILE_UPDATE"]
    },
    data: $ProfileSchema
  },
  "required" : ['eventType', 'data']
}
```

3.1.3 Revoke

To revoke a profile (issuer) the `PROFILE_REVOKE` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Profile revoke",
  "description": "Revoke a profile sproof event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["PROFILE_REVOKE"]
    },
    data: {
      type : 'object',
      properties: {
        reason: {
          description: "Description for revokation",
          type: "string",
          maxLength: 512,
        }
      }
    }
  },
  "required" : ['eventType', 'data']
}
```

3.1.4 UpdateKey

To update a profile's key the `PROFILE_UPDATE_KEY` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Update Profile Key",
```

(continues on next page)

(continued from previous page)

```
  "description": "Update profile key sproof event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["PROFILE_UPDATE_KEY"]
    },
    data: {
      type : 'object',
      properties: {
        : {
          description: "Description for revocation",
          type: "string",
          maxLength: 512,
          minLength: 512,
        }
      }
    }
  },
  "required" : ['eventType', 'data']
}
```

3.2 Document

A document can be any file with a hash reference. It is up to the user if the content of this file is publicly available or not. A document can have `0` to `n` receivers.

3.2.1 Register

To register a document the `DOCUMENT_REGISTER` event is needed.

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "Register a document",
    "description": "Register a document sproof event",
    "type": "object",
    properties : {
      "eventType" : {
        "type" : "string",
        "enum" : ["DOCUMENT_REGISTER"]
      },
      data:  {
        type : 'object',
        properties: {
          validFrom: {
            description: "Unix timestamp",
            type: "number",
          },
          validUntil: {
            description: "Unix timestamp",
            type: "number",
          },
          documentHash : {
```

(continues on next page)

(continued from previous page)

```
            description: "Hash of document to register",
            type: "string",
          },
          data: {
            type:'object',
          },

          locationHash: {
            description: "IPFS hash of document",
            type:'string',
          },
          name: {
            type: 'string',
            description: 'The name of the registration'
          },
          dependencies: {
            type: 'array',
            items: {
              type: 'string',
              description: 'Hashes of registration or receivers'
            },
          },
          receiverAttributes : {
            type: 'array',
            items: {
              type : 'string',
              description: 'The attributes which are linked to an receiver, e.g.,␣
↪name, email, dateOfBirth,...'
              }
          },
          receivers : {
            type: 'array',
            items: {
              type: 'string',
              description: 'Hashes of registration or receivers'
            }
          }
        },
        required: ['documentHash']
      }
    },
    "required" : ['eventType', 'data']
  }
```

3.2.2 Revoke

To revoke a document the `DOCUMENT_REVOKE` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Document revoke",
  "description": "Revoke a sproof document event",
  "type": "object",
  properties : {
    "eventType" : {
```

(continues on next page)

(continued from previous page)

```
      "type" : "string",
      "enum" : ["DOCUMENT_REVOKE"]
    },
    data: {
      type : 'object',
      properties: {
        documentHash: {
          description: "Hash of the registered document",
          type: "string"
        },
        reason: {
          description: "Description for revokation",
          type: "string",
          maxLength: 512,
        }
      }
    },
    required : ['eventId']
  },
  required : ['eventType', 'data']
}
```

3.3 Receiver

Documents can be issued to receivers. The receivers public representation is a pseudonous hash reference of its ID containing all attributes, and a timerange which defined the validity period.

3.3.1 Add

To add a receiver to a document the `DOCUMENT_RECEIVER_ADD` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Document receiver add",
  "description": "Add a receiver to a sproof document event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["DOCUMENT_RECEIVER_ADD"]
    },
    data: {
      type : 'object',
      properties: {
        receiverId: {
          description: "Id of the receivers hash",
          type: "string"
        },
        documentHash: {
          description: "Hash of the registered document",
          type: "string"
        }
      }
```

(continues on next page)

(continued from previous page)

```
    },
    required : ['receiverId', 'documentHash'],
  },
  required : ['eventType', 'data']
}
```

3.3.2 Revoke

To revoke a receiver of a document the `DOCUMENT_RECEIVER_REVOKE` event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Document receiver revoke",
  "description": "Revoke a sproof document receiver event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["DOCUMENT_RECEIVER_REVOKE"]
    },
    data: {
      type : 'object',
      properties: {
        receiverId: {
          description: "Id of the receivers hash",
          type: "string"
        },
        reason: {
          description: "Description for revokation",
          type: "string",
          maxLength: 512,
        }
      }
    },
    required : ['receiverId']
  },
  required : ['eventType', 'data']
}
```

CHAPTER 4

API

The sproof-api can be accessed by using the following domain: https://api.sproof.io/api/v1/profiles. To enable a fast integration into a node application we provide a js-sproof-client.

```
const {Sproof}  = require('js-sproof-client');
let sproof = new Sproof ({
    uri: 'https://api.sproof.io/',
    chainId: '3',
    chain: 'ethereum',
    version: '0.42'
});
sproof.newAccount();
```

In the following we describe the API calls for the sproof objects.

Note:

The params object provides fields to adjust the

- items per page : `per_page` : `Number`
- request page: `page` : `Number` Page to request
- entry : If you need only one specific entry use `id`:`String`

4.1 Transaction

```
sproof.getTransactions(params, callback)
```

Returns the transaction object.

Note: If the id property in params is set, this call returns the specified transaction, otherwise it returns a list of the last 10 entries.

4.1.1 Parameters

1. `Object` - params for call.
2. `Function` - Callback, returns an error object as first parameter and the result as second.

4.1.2 Returns

returns `Object` - A transaction object, or an error when no transaction was found:

- `_id` - `String`: Transaction hash.
- `blockNumber` - `Number`: Blocknumber of the block.
- `blockHash` 32 Bytes - `String`: Hash of the block where this transaction was in.
- `timestamp` - `Number`: Unix timestamp of the block creation time.
- `dhtHash` 32 Bytes - `String`: IPFS hash of the content.
- `events`- `List`: List of all events included in this transaction.
- `from` - `String`: Address of the sender.
- `publicKey` - `String`: PublicKey of the sender.

4.1.3 Example

```
sproof.getTransactions({id:
↪'0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b'} , (err, res)␣
↪=> {
    console.log(res);
});

> {
    "_id":"0xa8218f351a22c3660aeb4bdf1c94a6747bedf565f7b404c39060234a173a8234",
    "blockNumber":10031049,
    "timestamp":1547217372,
    "blockHash":"0x0a3513ed8cd714b199766b97de95845633e3a6b18189ac2de3d7d32183456cfe",
    "from":"0x683b44a82722d5cfd62e61c678ed2bfeb1f83cbb",
    "publicKey":
↪"0x2ab25035b3d357215c7d7656c9f3fa2d37a25e26dd0c75169dadb5b9292dfed3004b3094c8b4a5ba56e4550d77fabc1c
↪",
    "dhtHash":"Qmau18iJEcPA7qYwgk2WijS4spB3gAsDz3DTk9ptJZH8dc",
    "events":[
        "0x45c979da81c169057ac18d006ab3a0669aa6cd992bdb521b4a75d1779ba49486",
        "0xc5dd1d587d7f4cb83c75089f52a5d1e95c1faac044adca6d7d1adaa225434e16"
    ]
}
```

4.2 Event

```
sproof.getEvents(params, callback)
```

Returns the event object.

Note: If the id property in params is set, this call returns the specified event, otherwise it returns a list of the last 10 entries.

4.2.1 Parameters

1. `Object` - params for call.
2. `Function` - Callback, returns an error object as first parameter and the result as second.

4.2.2 Returns

returns `Object` - A event object, or an error when no event was found:

- `_id` - `String`: Event hash.
- `eventType` - `String`: Type of the event.
- `data` - `Object`: Events payload.
- `transaction` - `String`: Corresponding transaction hash of the event.
- `from` - `String`: Address of the sender.
- `timestamp` - `Number`: Unix timestamp of the block creation time.

4.2.3 Example

```
sproof.getEvents({id:
↪'0xac56a7953982dc8b066cfdcfd59a6b7d380c632aafd272a7da1863bfd49b3496'} , (err, res)␣
↪=> {
    console.log(res);
});

> {
    _id: '0x6f3c8113823f070b62905e979a9317e73dc218ed8d9b6d256190fe4e1144bfa8',
    eventType: 'DOCUMENT_REGISTER',
    data: { ... },
    transaction: '0x918ad9f8dd13bf3a309b0d10235bdb1fb7e9f7febd789b052c73fc6c97e442e5',
    from: '0x3b80e8e6756c26cae3062e7e07977403ced346e0',
    blockNumber: 9980757,
    timestamp: 1546855032
}
```

4.3 Profile

```
sproof.getProfiles(params, callback)
```

Returns the profile object.

Note: If the id property in params is set, this call returns the specified profile, otherwise it returns a list of the last 10 entries.

4.3.1 Parameters

1. `Object` - params for call.
2. `Function` - Callback, returns an error object as first parameter and the result as second.

4.3.2 Returns

returns `Object` - A profile object, or an error when no profile was found:

- `_id` - `String`: Address of profile owner.
- `data` - `Object`: Profile payload.
- `publicKey` - `String`: Profiles public key.
- `lastUpdate` - `Number`: Unix timestamp of the last interaction from this profile.
- `timestamp` - `Number`: Unix timestamp of the creation date.
- `valid` - `Boolean`: `TRUE` if the profile was not revoked.
- `registrations` - `Object`: List of registration events.
- `events` - `Object`: List of all events.
- `confirmations` - `Object`: Confirmation collection

4.3.3 Example

```
sproof.getProfiles({id: '0x86ec4f0b4e8ecc2f13f8ad86d9f6c2db30648b96'} , (err, res) =>
↪{
    console.log(res);
});

> {
    _id: '0x86ec4f0b4e8ecc2f13f8ad86d9f6c2db30648b96',
    data: { ... },
    publicKey:
↪'0x2ab25035b3d357215c7d7656c9f3fa2d37a25e26dd0c75169dadb5b9292dfed3004b3094c8b4a5ba56e4550d77fabc1c
↪',
    lastUpdate: 1545231020,
    timestamp: 1545231020,
    valid: true,
    registrations: [],
```

(continues on next page)

(continued from previous page)

```
    events:
    [
        '0xfe0bbd902a699a4d6546e20c2c199398f6f454354df9e93f17e780904ce794e9'
    ],
    confirmations: [ ... ]
}
```

4.4 Registrations

```
sproof.getRegistrations(params, callback)
```

Returns the registration object.

Note: If the id property in params is set, this call returns the specified registration, otherwise it returns a list of the last 10 entries.

4.4.1 Parameters

1. `Object` - params for call.
2. `Function` - Callback, returns an error object as first parameter and the result as second.

4.4.2 Returns

returns `Object` - A registration object, or an error when no registration was found:

- `_id` - `String`: Hash of the registration.
- `issuer` - `Object`: Address of the issuer.
- `event` - `String`: Corresponding event registration hash.
- `validFrom` - `Number`: Unix timestamp valid from.
- `validUntil` - `Number`: Unix timestamp valid until.
- `documentHash` - `String`: Hash of the registered document.
- `valid` - `Boolean`: `TRUE` if the registration was not revoked.
- `dependencies` - `Object`: List of dependencies.

4.4.3 Example

```
sproof.getRegistrations({id:
↪'0xb4af7c7b9d4ab6dbe222d4f1c5f8837159d3efbacfe34d1fb5e186ec59fafaec'} , (err, res)␣
↪=> {
    console.log(res);
});
```

(continues on next page)

(continued from previous page)

```
> {
    _id: '0xb4af7c7b9d4ab6dbe222d4f1c5f8837159d3efbacfe34d1fb5e186ec59fafaec',
    issuer: '0x86ec4f0b4e8ecc2f13f8ad86d9f6c2db30648b96',
    event: '0x74ff215595298423dd1569356e9c30540cd85ad941c17dce762fe52326a08c43',
    validFrom: null,
    validUntil: null,
    documentHash: '0xb4af7c7b9d4ab6dbe222d4f1c5f8837159d3efbacfe34d1fb5e186ec59fafaec
↪',
    valid: true,
    dependencies: []
}
```

4.5 Validation

```
sproof.getValidation(id, callback)
```

Returns the validation object.

4.5.1 Parameters

1. `String` - hash to verify.
2. `Function` - Callback, returns an error object as first parameter and the result as second.

4.5.2 Returns

returns `Object` - A registration object, or an error when no registration was found:

- `validation` - `Object`: Contains boolean values which indicates if the registration or the profile was revoked or not.
- `registration` - `Object`: Registration event.
- `profile` - `Object`: Issuer payload

4.5.3 Example

```
sproof.getValidation(
↪'0x5d7a02fda80aa4f70032c180ec3aa4a4f3f3075ae7abeb514186be1f104dd271' , (err, res) =>
↪ {
    console.log(res);
});

> "validation": {
        "registration":true,
        "profile":true
    },
    "registration":{ ... }
```

(continues on next page)

(continued from previous page)

```
    "profile" : { ...  }
}
```

CHAPTER 5

sproof API client

The sproof API client is a docker-container which encapsulates the `js-sproof-client` and can be access with standard API calls. Currently this module supports the registration and the revocation of arbitrary files.

After the setup phase the API can be access only with a valid access token, which is generated locally.

5.1 Register file

```
POST: https://{{yourDomain}}/api/v1/file/register?public=true&name=NameOfYourDocument&
↪accessCode={yourAccessCode}
```

The body must contain a file embedded into a `form-data` the name of the document must be `file`.

Returns the hash and the id of the document. Additionally if the parameter public is set to true, it will also return the location hash (IPFS reference, which can be accessed via https://ipfs.io/ipfs/{locationHash}.

5.1.1 Parameters

1. `accessCode` - Needed for authorization.
2. `public` - If the file should be publicly access able, set public to true. Default is false.
3. `name` - Optional a name for the file, this files could also be used as a tag.

5.1.2 Returns

- `hash` - `String`: The hash of the document.
- `location` - `String`: If the parameter public is set to true, it will return the IPFS location hash.
- `id` - `String`: The id of the file, calculated with the issuers address and the document hash.

5.2 Revoke file

```
POST: https://{{yourDomain}}/api/v1/file/revoke?accessCode={yourAccessCode}
```

The body must contain a file embedded into a `form-data` the name of the document must be `file`.

5.2.1 Parameters

1. `accessCode` - Needed for authorization.

5.3 Verify file

```
POST: https://{{yourDomain}}/api/v1/file/verify
```

The body must contain a file embedded into a `form-data` the name of the document must be `file`.

5.3.1 Returns

returns `List` - A list of registration objects, or an error when no registration was found:

- `validation` - `Object`: Contains boolean values which indicates if the registration or the profile was revoked or not.
- `registration` - `Object`: Registration event.
- `profile` - `Object`: Issuer payload

5.4 Commit

The commit to the sproof platform is performed according the defined schedule. If a irregular commit is necessary this call can be used.

```
GET: https://{{yourDomain}}/api/v1/commit?accessCode={yourAccessCode}
```

5.4.1 Parameters

1. `accessCode` - Needed for authorization.

5.4.2 Returns

returns `Object` - A object of all information which is send to the sproof platform. This includes all events and attached data.

5.5 State

Returns the current state of the client api. This includes also information about transaction and events including ids.

```
GET: https://{{yourDomain}}/api/v1/state?accessCode={yourAccessCode}
```

5.5.1 Parameters

1. `accessCode` - Needed for authorization.

5.5.2 Returns

returns `Object` - A object of all information which is stored of the premium users.

CHAPTER 6

Help

https://gitter.im/sproof