sproof-js Documentation

Release 1.0

Andreas Unterweger, Clemens Brunner, Fabian Knirsch

Feb 06, 2020

First steps

CHAPTER 1

Create your sproof profile

In order to use the sproof infrastructure, a profile is required. Creating a profile is simple and consists of the following steps:

1.1 Create a new account

Visit https://app.sproof.io/#/signup to create your account. Each account will be associated with a profile.

1.2 Back up your sproof-code

Export your `sproof-code` in the Settings page (https://app.sproof.io/#/settings/) after signup. Please note that it is **very** important that you download and back up your `sproof-code`. It cannot be replaced or restored if it gets lost, unless you (the account owner) have a backup of it.

1.3 Verify your Website (recommended)

You need to enter your Website in the signup phase. In order to verify that you are in control of its domain, you need to upload a file to your Web server, which is accessible under your root domain. You can find more detailed instructions on the Web app (https://app.sproof.io/#/verify-domain).

This process is necessary to provide additional evidence that you are in fact the organisation you created the profile for. It also helps others to trust and confirm your identity (see below). Your identity is linked to your profile.

1.4 Confirm other profiles (recommended)

Now that your identity is verified, you can confirm other profiles in the sproof network and other profiles can confirm yours. Confirming other sproof profiles helps the sproof community to build a strong network of profiles trusting each other. This helps every user of the sproof network to judge the authenticity of profiles. A profile with a verified Website that is confirmed by another profile with a verified Website is more likely to be trusted by a user if the user knows either profile and/or their Website.

1.5 Get started

You are now ready to get started and use sproof. sproof provides a free testnet (*sproof networks*) for developers for testing purposes. You can change the network in the Settings page (https://app.sproof.io/#/settings/) as well as in all configuration files.

Regardless of whether you just want to test sproof or integrate it right away, you can find your next steps here: *Integrate sproof into an existing application*.

CHAPTER 2

Integrate sproof into an existing application

This page is targeted at developers who want to integrate sproof into an existing application for signing and timestamping digital data, e.g., certificates or diplomas.

For testing and production, two separate networks are available, see *sproof networks*.

Hint: Using the test network is free and allows for easy testing and integration.

In order to fully integrate sproof into your software application, four steps are needed:

2.1 1. Create a sproof profile

If you have not done so already, please create a sproof profile first. Visit *Create your sproof profile* for more detailed instructions. You will need your `sproof-code` in order to configure your docker client in the next step.

2.2 2. Install the docker client

In order to make use of the sproof API, the sproof API client needs to be set up on a physical or virtual machine. The client is available as a ready-to-use docker image. Please see *Docker client setup* for setup instructions.

Hint: If you plan to use the docker client in production, we recommend to host this service on a public domain over TLS.

After the docker client setup, you get your access code for use in the next step.

2.3 3. Integrate the client API

You can now access the sproof API through the docker client that has been set up in the previous step. Examples and instructions on how to use the client API are provided in the docker API documentation (*docker client API*). You need your access code from the previous steps for most of the API calls.

Hint: Depending on your use case, it may be sufficient to only use the *verify* API call, which is one POST request. Fully ready-to-use examples are provided in multiple common programming languages.

Note that, for testing and production, two separate networks are available, see *sproof networks*.

Hint: Using the test network is free and allows for easy testing and integration.

2.4 4. Embed an iframe into your Website

After you have finished integrating the API and everything works as desired, you can embed the Webpage of the docker client into your application, e.g., by using an iframe.

Port 6001 of the machine on which you set up the docker client (step 2) features a HTTP server with a small verification page. Use, http://localhost:6001 or http://<your-machine-domain>:6001, respectively, to display this verification page. Alternatively, you can embed it into another page as described above.

If you are looking for an easier option for providing document verification that does not require managing certificates or performing other maintenance of the Docker client, please refer to *How to quickly embed the sproof verifier*.

CHAPTER 3

How to quickly embed the sproof verifier

The sproof verifier can be used to let anyone verify your documents on your website. If you confirmed others in the sproof network, their documents can also be verified this way.

The verifer is usually shipped with a docker container that provides an embeddable iframe. If you prefer a quicker way for testing and/or deploying, this description of the quick setup also provides a customizeable iframe in the end, albeit without the need to set up the docker container yourself.

3.1 Configuration

The source of the iframe to embed can be specified by a verifier URL with URL parameters. Depending on the URL and its parameters, you can customize the verifier. The full URL with parameters is:

```
{verificationUrl}/#/verify/{lang}?color={color}&btnFontColor={btnFontColor}&profile={profil
```

The URL (`verificationUrl`) can either be `https://verify.sproof.io/` for the mainnet or `https://ropsten.sproof.io/` for the testnet or `yourDomain` on which you host the docker API client. Please refer to *sproof networks* for a more detailed description about the sproof networks.

The following URL parameters may be specified:

- `lang` - `String`: The verifier page is available in English (value `en`) and German (`de`). The default language is English (`en`).
- `color` - `String`: Enter the main color of the verifier UI in hexadecimal encoding without #, e.g. `ffffff` for white. The default color is `sproof color`.
- `btnFontColor` - `String`: The font color for the Upload button. The default value is white.
- `profile` - `String` (strongly recommended): Enter your sproof profile address here (you can find it at the bottom of your Settings page at https://app.sproof.io/#/settings/). This address ensures that only documents registered by your profile or those of friends you confirmed in the sproof network are shown als valid. If no `profile` is specified, sproof's profile is used. This fallback behavior is not recommended.
- `font` - `String`: The font used for the verifier UI. If the browser displaying the verifier does not support it, a browser-dependent default font may be used. If no `font` is specified, the default font `Catamaran` is used.

An example URL is `https://verify.sproof.io/#/verify/de?color=000000&btnFontColor=ffffff&profile`
A full example for embedding can be found below.

3.2 Full example

The following example creates an embeddable iframe with a verifier in the mainnet.

```
<iframe src="https://verify.sproof.io/#/verify/de?color=000000&btnFontColor=ffffff&
↪profile=0x711d88f76c98a023a7ecf27e167df3f533661626&font='sans-serif'" style="width:␣
↪100%;height: 300px;border: none;"></iframe>
```

CHAPTER 4

sproof networks

sproof currently runs on two networks. One is for production and one is for testing purposes. They are distinguished by their *chainID*. Do not use any other *chainID* value than the ones listed below. Using other *chainID* values results in undefined behavior.

4.1 Ethereum main chain

chainId=1

sproof runs on the Ethereum main chain for use in prodution. This *chainId* is also used in the provided configuration files. Please note that you need to select a plan on the sproof Website (https://app.sproof.io) in order to interact with the main chain. You can choose a plan on the Settings page (https://app.sproof.io/#/settings/).

4.2 Ethereum Ropsten Testnet

chainId=3

The Ropsten testnet is an Ethereum blockchain designed to test decentralized applications before deploying them. You can use this testnet to submit transactions and to interact with the sproof network for free. This allows you to implement and test your integration of the sproof API without additional costs.

Note that using the testnet does not give you any of the guarantees of the Ethereum main chain, which you would normally expect from a blockchain. In particular, there is no mining and no support, meaning that data may be changed or lost without prior notice. Do **not** ever use the testnet in production.

CHAPTER 5

Docker client setup

The sproof API client is capable of registering, revoking and verifying data with an existing sproof account. It comes as a ready-to-use docker image that needs to be set up before its use. This document explains the prerequisites, the actual setup and the operation of the docker client.

5.1 Prerequisites

The sproof API client is based on Docker Compose and requires a sproof account.

5.1.1 Docker Compose

Before you can set up the docker client, you need to install Docker and Docker Compose on the machine where you want to install the API client. Please refer to the linked documentation for installation instructions for your operating system.

5.1.2 sproof account

In addition, a sproof account is required. Visit *Create your sproof profile* for instructions on how to create an account. Note that testing is possible with a free account, while production-level operation requires a premium account.

5.2 Setup instructions

5.2.1 1. Get *sproof-client-api*

The sproof docker API client is available at https://github.com/sproof/sproof-client-docker. You can obtain the client either through `git` or through a regular download.

git

If you have `git` installed on your machine, you can use it to download the client:

```
git clone https://github.com/sproof/sproof-client-docker.git
```

Regular download

Download the source file archive and unpack the downloaded archive, e.g., with the following commands on Linux:

```
wget https://github.com/sproof/sproof-client-docker/archive/master.zip
unzip master.zip
rm master.zip
```

On Windows, the following PowerShell commands (require v5 or higher) do the same:

```
wget https://github.com/sproof/sproof-client-docker/archive/master.zip -
↪OutFile master.zip
Expand-Archive master.zip -DestinationPath .
Remove-Item master.zip
```

After this step, you have a folder named `sproof-client-docker` containing the source files of the docker client. This folder is the basis for all subsequent steps.

5.2.2 2. Set the docker client configuration

Set your sproof code and (optionally) other parameters based on your needs in the `data/config.js` file. Please find a detailed description of the parameters in *Configuration of the docker image*.

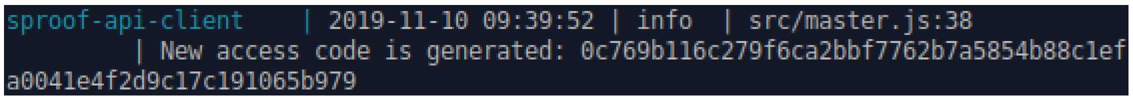
5.3 Operation instructions

Once the required configuration files are in place, the client can be started. As soon as it is running, you can access the docker client API through it.

In order to start the docker API client, run

`docker-compose up`

After the build process completes successfully, an access code will be output both, on the console and in `data/accessCode.json`. The output on the console looks like this:



```
sproof-api-client    | 2019-11-10 09:39:52 | info  | src/master.js:38
          | New access code is generated: 0c769b116c279f6ca2bbf7762b7a5854b88c1ef
a0041e4f2d9c17c191065b979
```

The access code is a token which secures your API endpoint. It is is necessary for sending and processing most external API requests.

The client, it will be started automatically. It will host an API endpoint as well as the verifier on port 6001 through a Web server.

5.3.1 Accessing the API

Once the docker client is running, you can access the client API through it. Note that, for most API calls, you need the access code that has been output during the start process. For a detailed documentation of the API endpoint as well as examples in multiple common programming languages, please see *docker client API*.

CHAPTER 6

Configuration of the docker image

The sproof API client can be configured based on your needs. The configuration must be specified in the *data/config.js* file before the docker image is run. An example file as well as an explanation of all configuration parameters are provided below.

6.1 Example configuration file

The following is an example configuration file (also available in the sproof-api-client respository). It contains a sample configuration which requires only setting your sproof code (see below) and can be readily used without further changes in most common setups. The parameters are explained below.

```
{
  api:{
    port: 6001
  },
  commit: {
    //time: '13:13',
    interval: 10
  },

  validateOnlyConfirmedIssuers: true,

  sproof: {
    sproofPremium: true,
    uri: 'https://api.sproof.io/',
    credentials: {
      sproofCode: 'YOUR SPROOF CODE'
    },

    chainId: '3',
    chain: 'ethereum',
    version: '0.42',
```

(continues on next page)

(continued from previous page)

```
  },
};
```

6.2 Parameters

There are three types of parameters - mandatory profile-related parameters, network-related parameters and blockchain-related parameters.

6.2.1 Profile-related parameters

- `sproofCode` - `String`: Enter your sproof code here. Without a valid sproof code, the client API will not work.
- `sproofPremium` - `Boolean`: If this parameter is set to `true`, your locally hosted client will synchronize pending events with the Web app. The client will also send keep-alive messages to the sproof servers so that you can see the status of your docker client in the Web app.

6.2.2 Network-related parameters

- `port` - `Number`: The *internal* (docker-side) HTTP port used to expose the API. If you want to change the HTTP port used to expose the API externally, you need to change the port mapping in the *docker-compose.yml* file.
- `uri` - `String`: The address where the backend (`sproof-core`) is running. At the moment, only the official sproof backend (https://api.sproof.io/) can be used.

6.2.3 Blockchain-related parameters

- `time` - `HH:MM`: The time of day when to commit your documents to the blockchain through a transaction. If this parameter is set, the `interval` parameter must not be set at the same time.
- `interval` - `Number`: The time interval in minutes for commiting your documents to the blockchain. For example, a value of 10 means that committing happens every ten minutes. If this parameter is set, the `time` parameter must not be set at the same time.
- `validateOnlyConfirmedIssuers` - `Boolean`: If this is `true`, only documents which are issued by yourself or from a sproof profile which you confirmed can be validated. Otherwise, your locally hosted Web UI will validate all documents registered in the sproof network.
- `chainId` - `String`: The network to be used for all operations. For details about sproof networks, see *sproof networks*.

CHAPTER 7

docker client API

The sproof API client is a docker container which encapsulates the `js-sproof-client` and can be accessed through regular API calls. Currently, this module supports the registration and the revocation of arbitrary files. The general structure of the API and the available operations are described below, followed by sample code in different programming languages.

7.1 General API structure

The API is REST based and uses GET and POST requests. The general structure of the API is as follows:

```
{method}: https://{yourDomain}/api/v{version}/{operation}?{parameters}&accessCode=
↪{yourAccessCode}
```

- `method`: The HTTP method (GET or POST).
- `yourDomain`: The domain on which you host the docker API client.
- `version`: The API version (currently `1`).
- `operation`: The API operation that you want to call, e.g., `commit` (see below).
- `parameters`: The parameters required for the operation (see below).
- `yourAccessCode`: After the setup phase, the API can be accessed only with a valid access token. This token is generated locally. It must be provided in most calls to the API.

The return code of any API call is a HTTP status code. The status code indicates whether the call was successful or, in case it was not, why. The following status codes may be returned:

- `200` (OK): The call was successful.
- `400` (Bad Request): There are either parameters missing or the parameter values are invalid (e.g., out of range).
- `401` (Unauthorized): You do not have permission to perform this operation. Make sure that you provided your access code.

- `402` (Payment Required): You are trying to perform an operation for which you do not have sufficient funds. If you are not a premium user, choose a plan. If you are already a premium user, consider upgrading your plan. A plan can be chosen in your profile settings (https://app.sproof.io/#/choose-plan/).
- `403` (Forbidden): You do not have permission to perform this operation. Your access code is valid.
- `404` (Not Found): The ID, e.g., document or profile ID, that you specified does not exist.
- `406` (Not Acceptable): The Web server does not support the set of constraints you specified in the Accept headers of your request. Remove or loosen the constraints, i.e., send more liberal Accept headers.
- `500` (Internal Server Error): An error occured while processing the call. Please contact us (*Contact*) if you receive this error so that we can investigate and fix it.
- `501` (Not Implemented): This feature is not implemented (yet). If you think that this feature should be implemented, please contact us (*Contact*).

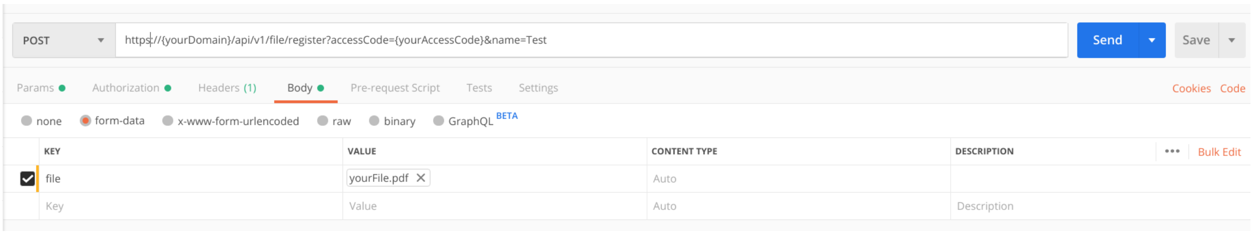
The following sections list the available API operations and their parameters. Sample code in different programming languages is available at the end (*API call example*).

7.2 Register file

This call registers a file in the sproof system and returns the hash as well as the ID of the registered document.

```
POST: https://{yourDomain}/api/v1/file/register?public=false&name={name}&accessCode=
↪{yourAccessCode}&validUntil=1590819300&tags=tag1,tag2,tag3
```

The body must contain a file embedded into a `form-data` field. The name of the document must be `file`, as illustrated below:



	KEY	VALUE	CONTENT TYPE	DESCRIPTION
✓	file	yourFile.pdf	Auto	
	Key	Value	Auto	Description

7.2.1 Parameters

1. `youAccessCode` - Needed for authorization (see *General API structure*).
2. `public` - If the file should be publicly accessible, set `public` to `true`. The default is `false`.
3. `name` (optional) - A name for the file. This name may also be used as a tag.
4. `validFrom` (optional) - A unix timestamp to indicate when the document is valid. If nothing is provided the document is valid after the registration.
5. `validUntil` (optional) - A unix timestamp when the registration expires.
6. `tags` (optional) - A comma separated list of tags for your document. Note: A tag contains only letters from A-Z and numbers. All other characters will be removed.

7.2.2 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. IPFS references can be accessed via https://ipfs.io/ipfs/{locationHash}.
- `id` - `String`: The ID of the file, calculated from the issuer's address and the document hash.

7.3 Revoke file

This call revokes a previously registered file.

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

The body must contain a file embedded into a `form-data` field. The name of the document must be `file` (see *Register file*).

7.3.1 Parameters

1. `youAccessCode` - Needed for authorization (see *General API structure*).

7.4 Verify file

This call checks whether the provided file has been registered and, if so, whether it and its issuer are valid.

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

The body must contain a file embedded into a `form-data` field. The name of the document must be `file` (see *Register file*).

7.4.1 Returns

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

- `validation` - `Object`: Contains two boolean values which indicate whether the registration and the profile that performed it are valid. If both boolean values are `true`, the registration and the profile are valid, i.e., they have not been revoked. If either value is `false`, the registration or the profile has been revoked, respectively.
- `registration` - `Object`: The registration event where the file had been previously registered.
- `profile` - `Object`: Information about the issuer of the file.

7.5 Commit

Commits to the sproof platform are performed according the defined schedule. If an irregular commit is necessary, this call can be used.

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

7.5.1 Parameters

1. `youAccessCode` - Needed for authorization (see *General API structure*).

7.5.2 Returns

The call returns `Object` - an object about all information which is sent to the sproof platform to perform the commit. This includes all registrations, events and attached data.

7.6 State

This call returns the current state of the client API.

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

7.6.1 Parameters

1. `youAccessCode` - Needed for authorization (see *General API structure*).

7.6.2 Returns

The call returns `Object` - an object about all information which is stored about the premium user. This includes information about the current transaction and events, including IDs.

7.7 API call example

The following is sample code to submit a (*Register file*) POST request to the API. The sample code is available in different, commonly used programming languages.

PHP

```
<?php

    $document = '{YOUR PDF FILE}';

    //write file to filesystem
```

(continues on next page)

(continued from previous page)

```
    $tempFileName = tempnam(sys_get_temp_dir(), 'pdfDocForSproof');
    file_put_contents($tempFileName, $document);


    $ch = curl_init(
        'https://{yourDomain}/api/v1/file/register?' . http_build_query([
            'name' => 'Example Name',
            'accessCode' => '{yourAccessCode}'
        ])
    );

    curl_setopt($ch, CURLOPT_POST, 1);

    curl_setopt($ch, CURLOPT_POSTFIELDS, [
        'file' => curl_file_create($tempFileName)
    ]);

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $server_response = curl_exec($ch);

    if (!curl_errno($ch)) {
      echo "Response: {$server_response}";
     } else {
       echo "Error: {$server_response}";
      }

    curl_close ($ch);
    unlink($tempFileName);

?>
```

C#

```
HttpClient client = new HttpClient(){
    BaseAddress = new Uri("{yourDomain}"),
    Timeout = TimeSpan.FromMilliseconds(5000)
};;

byte[] data = File.ReadAllBytes("{PATH TO FILE"});
const string apiPath = "https://yourDomain/api/v1/file/register?name={NAME_
↪OF_FILE}&accessCode={yourAccessCode}";
var content = new MultipartFormDataContent();
content.Add(new ByteArrayContent(data), "file", "file");

try{
    var response = await client.PostAsync(apiPath, content);
    if (response.StatusCode != System.Net.HttpStatusCode.OK){
        //handle Error
    } else {
        string jsonString = await response.Content.ReadAsStringAsync();
        dynamic json = JsonConvert.DeserializeObject(jsonString);

        if (json.error != null){
            //handle Success
        } else {
            //handle Error
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
catch{
    //handle error
}
```

Javascript

```
const FormData = require('form-data');
const fetch = require('node-fetch');
var fs = require('fs');

let path = 'PATH TO FILE'
let accessCode = 'yourAccessCode'

var form = new FormData();
var readStream = fs.createReadStream(path);

form.append('file', readStream);
fetch(`https://{yourDomain}/api/v1/file/register?accessCode=${accessCode}&
↪name=test`, {
  method: 'POST',
  body: form
})
  .then(res => res.json())
  .then(result => {
    console.log('result', result);
  })
  .catch(error => {
    console.error('error', error);
});
```

Java

Coming soon. Feel free to edit the documentation on GitHub.

CHAPTER 8

JavaScript sproof client

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.

8.1 Create a node project

8.1.1 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.

8.1.2 Create the project structure

Create a new folder for your project and open it in a terminal. Run `npm init` and follow the instructions. After this step is complete, 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.

8.2 Create an account

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

8.2.1 Standard

If you want to use your own Ether (cryptocurrencies) to pay 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 the website.

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

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

8.2.2 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 our Web app: 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 consisting of 12 randomly chosen words.

sproof does not store your `sproof-code`. In case that you lose your `sproof-code` we cannot recover it.

8.3 Create a config file

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',
    version: '0.42'
```

(continues on next page)

(continued from previous page)

```
};
module.exports = config;
```

8.4 Examples

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

8.4.1 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);
});
```

8.4.2 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({
```

(continues on next page)

(continued from previous page)

```
  name: 'new sproof account 1',
  profileText: 'Sproof Test Account',
  image: 'Qma34dB4B4N4eS5ibBkwtjTSTNCRdJrVY6E25DFuFuU8Sd',
  homepage: 'https://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);
});
```

8.4.3 Publish and register a local PDF file

Upload a PDF file 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) {
    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);
    });
```

(continues on next page)

(continued from previous page)

```
  }else
    console.error(err)
});
```

CHAPTER 9

API

The sproof API can be accessed through the domain https://api.sproof.io/api/v1/profiles. To enable a fast integration into a node application, we provide a JavaScript client for sproof named `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.

The `params` object provides fields to adjust the

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

9.1 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.

9.1.1 Parameters

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

9.1.2 Returns

Returns `Object` - a transaction object, or an error when no transaction was found:

- `_id` - `String`: Transaction hash.
- `blockNumber` - `Number`: The number of the block.
- `blockHash` 32 Bytes - `String`: Hash of the block where this transaction is contained 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`: Public key of the sender.

9.1.3 Example

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

> {
    "_id":"0xa8218f351a22c3660aeb4bdf1c94a6747bedf565f7b404c39060234a173a8234",
    "blockNumber":10031049,
    "timestamp":1547217372,
    "blockHash":"0x0a3513ed8cd714b199766b97de95845633e3a6b18189ac2de3d7d32183456cfe",
    "from":"0x683b44a82722d5cfd62e61c678ed2bfeb1f83cbb",
    "publicKey":
↪"0x2ab25035b3d357215c7d7656c9f3fa2d37a25e26dd0c75169dadb5b9292dfed3004b3094c8b4a5ba56e4550d77fabc1
↪",
    "dhtHash":"Qmau18iJEcPA7qYwgk2WijS4spB3gAsDz3DTk9ptJZH8dc",
    "events":[
        "0x45c979da81c169057ac18d006ab3a0669aa6cd992bdb521b4a75d1779ba49486",
        "0xc5dd1d587d7f4cb83c75089f52a5d1e95c1faac044adca6d7d1adaa225434e16"
    ]
}
```

9.2 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.

9.2.1 Parameters

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

9.2.2 Returns

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

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

9.2.3 Example

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

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

9.3 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.

9.3.1 Parameters

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

9.3.2 Returns

Returns `Object` - a profile object, or an error when no profile was found:

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

9.3.3 Example

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

> {
    _id: '0x86ec4f0b4e8ecc2f13f8ad86d9f6c2db30648b96',
    data: { ... },
    publicKey:
↪'0x2ab25035b3d357215c7d7656c9f3fa2d37a25e26dd0c75169dadb5b9292dfed3004b3094c8b4a5ba56e4550d77fabc1
↪',
    lastUpdate: 1545231020,
    timestamp: 1545231020,
    valid: true,
    registrations: [],
    events:
    [
        '0xfe0bbd902a699a4d6546e20c2c199398f6f454354df9e93f17e780904ce794e9'
    ],
    confirmations: [ ... ]
}
```

9.4 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.

9.4.1 Parameters

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

9.4.2 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 for the start of the validity period.
- `validUntil` - `Number`: Unix timestamp for the end of the validity period.
- `documentHash` - `String`: Hash of the registered document.
- `valid` - `Boolean`: `TRUE` if any only if the registration was not revoked.
- `dependencies` - `Object`: List of dependencies.

9.4.3 Example

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

> {
    _id: '0xb4af7c7b9d4ab6dbe222d4f1c5f8837159d3efbacfe34d1fb5e186ec59fafaec',
    issuer: '0x86ec4f0b4e8ecc2f13f8ad86d9f6c2db30648b96',
    event: '0x74ff215595298423dd1569356e9c30540cd85ad941c17dce762fe52326a08c43',
    validFrom: null,
    validUntil: null,
    documentHash: '0xb4af7c7b9d4ab6dbe222d4f1c5f8837159d3efbacfe34d1fb5e186ec59fafaec
↪',
```

(continues on next page)

(continued from previous page)

```
    valid: true,
    dependencies: []
}
```

9.5 Validation

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

Returns the validation object.

9.5.1 Parameters

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

9.5.2 Returns

Returns `Object` - a registration object, or an error when no registration was found:

- `validation` - `Object`: Contains boolean values which indicate whether or not the registration or the profile were revoked.
- `registration` - `Object`: Registration event.
- `profile` - `Object`: Issuer payload

9.5.3 Example

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

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

CHAPTER 10

Schema

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

10.1 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 posts and user accounts. This is to␣
↪increase the trust of the user's account",
      "items": {
        "type": "object",
        "properties": {
          "userId": {
            "description": 'The unique user ID of a social media account.',
            "type": "string"
          },
          "messageId": {
            "description": "The unique message ID which contains 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"]
};
```

10.1.1 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 a sproof profile event",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["PROFILE_REGISTER"]
    },
    data: $ProfileSchema
  },
  "required" : ['eventType', 'data']
}
```

10.1.2 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']
}
```

10.1.3 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 revocation",
          type: "string",
          maxLength: 512,
        }
      }
    }
  },
  "required" : ['eventType', 'data']
}
```

10.1.4 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 of revocation",
          type: "string",
          maxLength: 512,
          minLength: 512,
        }
      }
    }
  },
  "required" : ['eventType', 'data']
}
```

10.2 Document

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

10.2.1 Register

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

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "Register a document",
    "description": "sproof event to register a document",
    "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 the document to register",
              type: "string",
            },
            data: {
              type:'object',
            },

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

10.2.2 Revoke

To revoke a document, the DOCUMENT_REVOKE event is needed.

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Document revoke",
  "description": "Event to revoke a sproof document",
  "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 of revocation",
          type: "string",
          maxLength: 512,
        }
      }
    },
    required : ['eventId']
  },
  required : ['eventType', 'data']
}
```

10.3 Receiver

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

10.3.1 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": "Event that adds a receiver to a sproof document",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["DOCUMENT_RECEIVER_ADD"]
    },
    data: {
      type : 'object',
      properties: {
        receiverId: {
          description: "ID of the receiver's 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']
}
```

10.3.2 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": "Event to revoke a sproof document receiver",
  "type": "object",
  properties : {
    "eventType" : {
      "type" : "string",
      "enum" : ["DOCUMENT_RECEIVER_REVOKE"]
    },
    data: {
      type : 'object',
      properties: {
        receiverId: {
          description: "ID of the receiver's hash",
          type: "string"
        },
        reason: {
          description: "Description for revocation",
          type: "string",
          maxLength: 512,
        }
      }
    },
    required : ['receiverId']
  },
  required : ['eventType', 'data']
}
```

CHAPTER 11

Contact

For questions, please contact developers@sproof.io or use our gitter channel: https://gitter.im/sproof