FutureTrust

White Paper

gTSL Search Engine

Document Identification	
Date	13/07/2017
Status	Public
Version	Version 0.1

Abstract: The present document provides an overview of the search engine defined for the development of the Global Trust Service Status List that is part of the Future Trust project. It also specifies the technologies and methodologies used to implement the search operations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

1. Executive Summary

The present document provides an overview of the search engine defined for the development of the Global Trust Service Status List that is part of the Future Trust project. It also specifies the technologies and methodologies used to implement the search operations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

2. Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

3. About FutureTrust

Against the background of the regulation 2014/910/EU on electronic identification (eID) and trusted services for electronic transactions in the internal market (eIDAS), the FutureTrust project, which is funded within the EU Framework Programme for Research and Innovation (Horizon 2020) under Grant Agreement No. 700542, aims at supporting the practical implementation of the regulation in Europe and beyond.

For this purpose, the FutureTrust project aims to address the need for globally interoperable solutions through basic research with respect to the foundations of trust and trustworthiness, actively support the standardisation process in relevant areas, and provide Open Source software components and trustworthy services which will ease the use of eID and electronic signature technology in real world applications. In particular, the FutureTrust project will extend the existing European Trust Service Status List (TSL) infrastructure towards a “Global Trust List”, develop a comprehensive Open Source Validation Service as well as a scalable Preservation Service for electronic signatures and seals and will provide components for the eID-based application for qualified certificates across borders, and for the trustworthy creation of remote signatures and seals in a mobile environment.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 700542

4. List of the different approaches

4.1 Use orbit-db

4.1.1 Description

orbit-db is a serverless, distibuted, peer-to-peer database. It has been developed on top of IPFS. Data in orbit-db can be stored in a Key-Value, Eventlog, Feed, Documents or Counters store. The implementation is written in Node.js.

4.1.2 Advantages

The main advantage is that orbit-db is developed on top of IPFS, the technology use in the project in order to store the data.

4.1.3 Disadvantages

orbit-db can only be used in a Javascript environment. It cannot provide a way to manage the different versions of the gTSL. The implementation is not optimized, it is based on a Logger in which each data is appended to the end of a JSON file which is stored in IPFS. It means that you need to load the whole file to perform actions on the database. Furthermore, orbit-db don't use indexes to retrieve data in an optimized way and there is no concurrency control.

4.2 Use aolog

4.2.1 Description

aolog is an append only log on IPFS with indexing. It is implemented in Javascript.

4.2.2 Advantages

It is based on IPFS and uses a Finger Tree as a data structure and a Bloom filter as a search method.

4.2.3 Disadvantages

It is implemented using append-only log storage, it means that it is impossible to update or remove data. Futhermore, it does not take into account the structure of the data, it is only lines containing data without links between those lines, so it impossible to deal with specific data structure.

4.3 Use YaCy

4.3.1 Description

YaCy is a distributed search engine that anyone can use to build a search portal for thier intranet or to help search the public internet. It is implemented in Java.

4.3.2 Advantages

It is a distributed search engine and anyone can deploy an instance in order to grow up the network of the search engine.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

4.3.3 Disadvantages

YaCy cannot be used on top of IPFS, unless if we modify the code to adapt it to IPFS. Futhermore, it only allow full-text search in the whole referenced pages and does not take into account the data structure used in the gTSL implementation.

4.4 Use Elasticsearch

4.4.1 Description

Elasticsearch is a distributed, RESTful search and analytics engine. It performs indexes on data and indexes are stored in JSON format. It is queryable and have a lot of options.

4.4.2 Advantages

The search operations are performed quickly using indexes. The queries can be customized depending on the data.

4.4.3 Disadvantages

Elasticsearch stores all the indexes locally, it means that we need to have a copy of an indexed gTSL. Elasticsearch can be distributed between multiple nodes but this increases the complexity of the application. Futhermore, it is needed to synchronize all Elasticsearch instance when CRUD operations are performed on the gTSL.

4.5 Reimplement an existing database on top of IPFS

4.5.1 Description

The idea is to use an existing implementation of a open-source database such as MongoDB and adapt it to allow data to be stored in IPFS.

4.5.2 Advantages

This is the most optimized way to perform search operations and store data. All data are stored in IPFS but on top of it there is a layer which allows us to query those data.

4.5.3 Disadvantages

It is very complicated to implement such a database, so it will take months or years and it is not the purpose of the project.

4.6 Implement a search engine based on ipfs-search

4.6.1 Description

ipfs-search is a search engine for IPFS. It performs searches on the whole IPFS public network. It is implemented in Node.js and Go.

4.6.2 Advantages

ipfs-search uses Elasticsearch to index data. It can be a starting point to implement a search engine based on Elasticsearch and IPFS.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

4.6.3 Disadvantages

ipfs-search only allows full-text search and does not take into account the file type. It means that a search can be performed in a JSON file as well as an image. The problem which can be raised is that we need to distribute all indexes between the nodes and to synchronize all instances of Elasticsearch.

4.7 Implement a custom graph-based search engine

4.7.1 Description

The idea is to represent the gTSL as a graph and perform search operations throughout this graph using well-known algorithms. Each node of the graph will represent a specific piece of data (chunks) which can be a Trusted List, a Trust Service Provider or a Trust Service. A chunk is not stored as the data itself but only its hash representation is stored which makes the whole graph lighter.

4.7.2 Advantages

Use hashes make the algorithm more efficient because if a data is not interesting the chunk will not be loaded in memory so only relevant data are kept in memory. This representation allows parallel actions when searching. It is possible to deal with different versions just by building the graph corresponding to the choosen version.

4.7.3 Disadvantages

The implementation has to be designed from scratch, even if existing algorithms from graph theory can be used. Another inconvenient is that the code will not be reusable entirely for other use cases because it will be optimized for the gTSL.

4.8 Implement a custom search engine based on inverted index

4.8.1 Description

The idea is to index data from the gTSL using inverted indexes. The purpose of this technique is to reference a set of values for a specific. For example, if we want to filter by countries (or Trusted Lists), we can register a hash table in which the user provide the country and the engine returns the set of Trust Service Providers attached to this country.

France	{ANTS, Caisse des dépôts, CertEurope, La Poste, Ministère de la Justice...}
Luxembourg	{LuxTrust}
Belgium	{Certipost, Zetes, QuoVadis BVBA}

You can optimize the following by only store hash representation of the data.

4.8.2 Advantages

It allows searches to be performed quickly because retrieving data is performed in a complexity of O(1).

4.8.3 Disadvantages

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

It is needed to store indexes in IPFS and maintain the hashes of the current in a smart-contract. This increases the complexity of adding, updating and removing data. The indexes have to be synchronize between all nodes to be efficiently used and concurrency control have to be implemented. Furthermore, this means to define a specific data structure in order to perform searches. It is mandatory to find a way to deal with versions. Another inconvenient is that the code will not be reusable entirely for other use cases because it will be optimized for the gTSL.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542

5. References

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700542