Making Licensing of Content and Data Explicit with Semantics and Blockchain

Information Management and Big Data - 8th Annual International Conference, SIMBig 2021, Proceedings

Gatta, David; Hinteregger, Kilian; Fensel, Anna

https://doi.org/10.1007/978-3-031-04447-2_25

Making Licensing of Content and Data Explicit with Semantics and Blockchain

David Gatta[1], Kilian Hinteregger[1], and Anna Fensel[1,2(✉)]

[1] Department of Computer Science, University of Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria
{David.Gatta,kilian.hinteregger}@student.uibk.ac.at

[2] Wageningen Data Competence Center and Consumption and Healthy Lifestyles Chair Group, Wageningen University and Research, 6708 PB Wageningen, The Netherlands
anna.fensel@wur.nl

Abstract. Creation and reuse of content and data are on the rise. Tracing of who, when and how has created and modified content and data in an explicit manner becomes paramount for transparent, explainable, efficient and fair digital ecosystems. We specifically address a challenge that there is currently no uniform method to link data and content with licenses explicitly, immutably and in a traceable way (with authentication). Thus we have created a blockchain-based method which makes it possible to attach semantic licenses with data and content. Ethereum and the Data Licenses and Clearance Center - "DALICC", a software framework for automated clearance of rights, are used as a bases for our prototype implementation for the license annotation (see: https://github.com/kilianhnt/dalicc-license-annotator). Particular care is taken to ensure that the solution remains generic. Its practical feasibility, such as compliance with requirements, particularly, authentication, affordable deployment and usage costs, is positively evaluated.

Keywords: Data · Content · Licenses · Semantics · Immutability · Blockchain · Etherium

1 Introduction

Creation of derivative data works is often accompanied by legal uncertainty about usage rights and high costs in the clearance of licensing issues. As one of the solutions to lower the costs of rights clearance and stimulating the data economy, the Data Licenses Clearance Center - DALICC project [1] has developed a software framework that supports automated clearance of rights issues in the creation of derivative data works. The solution is based on semantic technology, to ensure explicitness in license representation and decision making processes.

This paper addresses a further challenge, namely, that currently there is no uniform method to link data and content with licenses in an explicit and traceable manner. Currently license information is often only available as a human-readable text, but in our digitized world, it would be advantageous if computers could also read and interpret the

J. A. Lossio-Ventura et al. (Eds.): SIMBig 2021, CCIS 1577, pp. 370–379, 2022.
https://doi.org/10.1007/978-3-031-04447-2_25

rights granted in order to clear license issues quickly, as enabled by DALICC. However, DALICC, solutions on which it is built (such as ODRL, that can be also used to make license information explicit [2]) and other potentially relevant solutions (such as PROV-O [3] and SOLID [4], for assisting people to define how their data are managed) are not providing a possibility to link assign semantic licenses to owners in an immutable and traceable manner. The latter, however, is possible to make on blockchain, in a practical manner for tracing the ownership and right information of primary and derivative works. Therefore, this work aims to find such appropriate method to attach semantic licenses to data and content. To achieve this, an approach with a blockchain was chosen to design the entire system in a decentralized and therefore trustworthy manner. Inherent traceability of a blockchain is especially important for the use cases such as content and data creation and reuse. Specifically, the Ethereum platform was chosen for the implementation, because it is one of the most popular platforms for decentralized applications in the world [5].

The paper is structured as follows. The problem statement, our approach with functional and non-functional requirements are discussed in the next section. Further, the implementation section explains the chosen technologies and provides details about backend and frontend of the implementation. Finally, the evaluation and the conclusion sections conclude the paper.

2 Problem Statement

DALICC helps to determine which content and data can be shared with whom to what extent under which conditions. There is yet currently no standardized way to license digital content to allow both - machines and humans - to read and interpret the rights granted. In this section, we discuss the requirements of an application that can address this problem, continuing with the description of the technologies we chose to realize a prototype of the application and finally our implementation and evaluation.

2.1 Functional Requirements

The following functional requirements are placed on the application to meet the given problem by us. An unauthenticated user is a user who does not have any credentials. Hence, an unauthenticated user must not be signed in to make some actions. In contrast, an authenticated user is a user who has confirmed his/her identity so that each of his/her actions can be assigned to him/her. A licensor is an authenticated user who has already licensed data or content and is only referred to as the licensor in the context of these data or content. The requirements, that relate to the basic functions of the solution, are as follows (see also Fig. 1):

- An authenticated user must be able to license a file.
- The licensor of a specific file must be able to change the license of the file.
- An authenticated user must be able to license the content of a URI.
- The licensor of a specific URI must be able to change the license of the URI.
- An unauthenticated user must be able to get the license information associated with a specific file.

- An unauthenticated user must be able to get the license information associated with a specific URI.

2.2 Non-functional Requirements

In addition to the above mentioned functional requirements, the application must meet additional non-functional requirements. These are provided to ensure that the application has a certain level of security and trustfulness:

- There must be no authority that can act independently and can falsify license information, i.e. the representation of the licensor and the representation of the license.
- The license information must be publicly stored somewhere, so that anyone can see it.
- The license information must be secured so that no one can change it, except the licensor itself.
- To save storage, only a unique reference of the licensed data should be stored and not the entire data.
- The licenses should be represented by a reference to a license in the DALICC License Library [6].
- The whole licensing process should cost acceptably for the users (e.g. less than 1€) to ensure a realistic and practical solution for different industries (e.g. creative industries).

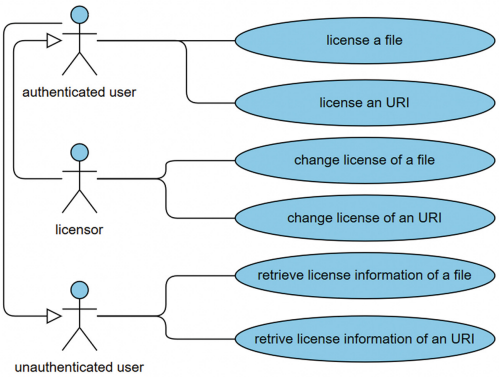

Fig. 1. Functional requirements visualized using a use-case diagram.

3 Implementation

The source code of the implemented application, as it is detailed further, is published openly on GitHub [7].

3.1 Technology Choices

According to Antonopoulos and Wood [8], a DApp is an application that is mostly or entirely decentralized. Additionally, a DApp has no downtime and is continuously available as long as the decentralized platform, which the application is using to provide the service, is operating. Thus DApp is well suited for the backend of the application. The application is mostly implemented using HTML, CSS and JavaScript.

Ethereum is a platform for building decentralized applications [9, 10]. It combines the blockchain technology, introduced by Nakamoto [11] and a Turing-complete programming language, to make it possible to execute computer programs called smart contracts. These properties of the Ethereum network make it a solid base to meet the application requirements.

The SHA-3 family consists of four cryptographic hash functions and was released by the National Institute of Standards and Technology (NIST) in 2015 after nine years of research [12, 13]. This hash algorithm is well suited for this application and its application Vue.js, CryptoJS, web3.js, Bignumber.js and Axios are used from javascript libraries.

3.2 Backend and Frontend

Backend Logic. The backend logic of the application is built on top of the Ethereum Platform. It is split up in three smart contracts, and is visualized in Fig. 2.

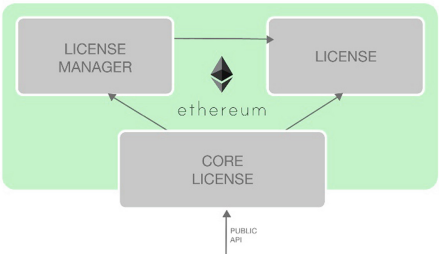

Fig. 2. Simplified visualisation of the smart contracts and their relations.

The components are interacting as follows when performing their core functions:

- *Creation of the Core-License contract.* Once the Core-License Contract is created, it also creates the License-Manager contract. The constructor of the License-Manager contract is called and as the parameter, the address of the Core-License contract is passed. This address later serves to check whether the call comes from the Core-License contract or not.
- *Licensing some data or content.* To license some data first the SHA3-256 hash of these data must be calculated. Then the function licenseData (uint hashValue, string memory licenseUri) of the Core-License contract must be called, which takes the hash value of the data as the first parameter and the URI of a license as the second parameter.

- *Changing the license of some data or content.* Changing the license of certain data can be done in the same way as described in the previous step. It should be noted that only the licensor itself can change the license.

Frontend. The frontend of the application was realized as mentioned above as a web application using Vue.js. The connections between the technologies are as follows web3.js is used to communicate with the Ethereum backend. The backend is set up before the application is published and the address of the Core License contract gets inserted into the slot provided in the code of the frontend. Bignumber.js is used to work with the hash values created by CryptoJS and to submit them to the backend using web3.js. The license titles and URI's get retrieved using an AJAX request from the DALICC License Library. In Fig. 3 we can see a simplified overview of the interactions between the frontend and the backend when a user licenses a file.

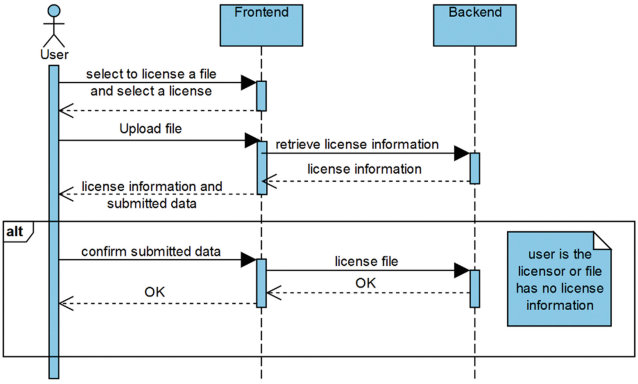

Fig. 3. Simplified sequence diagram for licensing a file.

The workflow for using the solution is as follows. To license a file or retrieve its license information, the user is guided through various steps. First, the user is prompted to choose if he/she wants to license content (step 1, 2 and 3), named as "license your work", or to retrieve license information about content (step 1 and 2), named as "retrieve license information", as shown in Fig. 4. This selection is made by clicking on the respective button. Subsequently, he/she has to follow these steps:

1. The user gets asked to choose whether he/she want license an URI or a file. If the user wants to license a URI or a file, then he/she can select a license from the DALICC License Library using an autocomplete combobox. The licenses are loaded from the DALICC License Library with an AJAX request. If the DALICC License Library is not reachable, the user gets displayed a warning and a predefined collection of licenses from the DALICC License Library. If the user only wants to retrieve the license information, the licenses do not get retrieved from the DALICC license library and the user cannot select a license.

2. In this step, the user must enter a URI in a text box or select a file using a file input field. Subsequently, he/she confirms his/her request by pressing a button. Then the SHA3-256 hash value of the URI or the file content is calculated which is used to retrieve the license information from the backend. To require no authentication and thus, a DApp able browser, the web3 provider Infura [14] is used. Now the license information is displayed to the user if it exists. If the user is the licensor or if the content is not already licensed then the user can continue with step 3. Otherwise, the user cannot continue with the process. To check if the user is the licensor the address given in the injected web3 API is compared with the licensors address retrieved from the backend.
3. Start licensing: in the last step the user can check his/her submitted data and then he/she can confirm and start the process by pressing the "start licensing" button. The SHA3-256 hash value is calculated and passed by the contract to the web3 Javascript interface. Depending on the user's DApp Browser, he/she receives a notification about the incoming transaction which he/she has to confirm. If he/she confirms the transaction, in our application the returned hash is displayed.

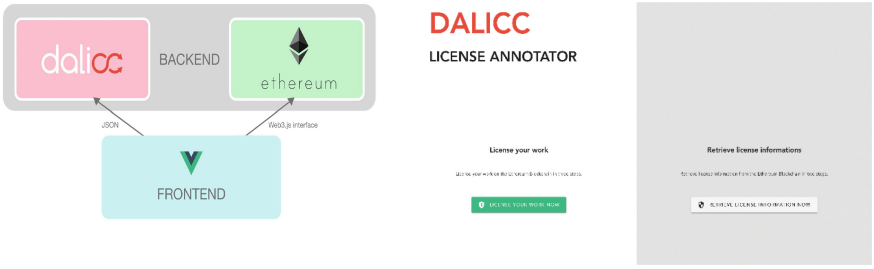

Fig. 4. Frontend positioning and entry page (licensing and license retrieval).

4 Evaluation

Building the application as a decentralized application on top of the Ethereum platform helped to meet the TRADE principles from the FAIR TRADE Framework [15]. Here, the property "Autonomous" of the FAIR TRADE Principle is not fully fulfilled, because the data in the blockchain is immutable and cannot be deleted. The blockchain is distributed and published all over the world and thus the data saved there can be accessed globally by anyone [8]. Since the only information that gets saved in the Ethereum network is the Ethereum address of a licensor, the hash value of the licensed data and the URI of a license, it does not get in conflict with the requirements. The licensor still has the choice of whether to remain anonymous behind the Ethereum address or indicate that he/she is the licensor.

Moreover, there are five cases, where the cost differs:

1) Deploying of the smart contract system, which has to be done only once.
2) License data or content when the smart contract for the license was not already created.
3) License data or content when the smart contract for the license was already created.
4) Modifying the license of some data or content when the smart contract for the new license was not already created.
5) Modifying the license of some data or content when the smart contract for the new license was already created.

These five cases were tested using the Rinkeby test network [16], an Ethereum test network that simulates the conditions of the main Ethereum network, so that developers can test their smart contracts for free, approaching the real life deployment settings. We have received the following results:

1. The smart contract system was deployed three times and each time the Gas (refers to the fee, or pricing value, required for the execution of code on the Ethereum platform) used was 1521249.
2. This case was tested five times with the results shown in Table 1.

Table 1. Gas costs for licensing where the license contract does not exist.

License URI	Hash value	Gas used
https://www.dalicc.net/license-library/CreativeCommonsAttribution20Austria	1	609177
https://www.dalicc.net/license-library/GnuFreeDocumentationLicense	2	609081
https://www.dalicc.net/license-library/BSD-4	3	587853
https://www.dalicc.net/license-library/GNU_GPL_v3	4	587913
https://www.dalicc.net/license-library/MIT	5	587829

3. Also this case was tested five times with the results shown in Table 2.

Table 2. Gas costs for licensing where the license contract exists.

License URI	Hash value	Gas used
https://www.dalicc.net/license-library/CreativeCommonsAttribution20Austria	10	78788
https://www.dalicc.net/license-library/GnuFreeDocumentationLicense	11	78692
https://www.dalicc.net/license-library/BSD-4	12	77782
https://www.dalicc.net/license-library/GNU_GPL_v3	13	77842
https://www.dalicc.net/license-library/MIT	14	77758

4. This case was tested five times with the results shown in Table 3.

Table 3. Gas costs for modifying the license of some data where the new license contract does not exist.

License URI	Hash value	Gas used
https://www.dalicc.net/license-library/Cc010Universal	10	570487
https://www.dalicc.net/license-library/OdcOpenDatabaseLicense	11	570583
https://www.dalicc.net/license-library/TheZlibLibpngLicense	12	570559
https://www.dalicc.net/license-library/Wtfpl	13	570379
https://www.dalicc.net/license-library/EclipsePublicLicense20	14	570583

5. This case was tested five times with the results shown in Table 4.

Table 4. Gas costs for modifying the license of some data where the new license contract already exists.

License URI	Hash value	Gas used
https://www.dalicc.net/license-library/MIT	10	60284
https://www.dalicc.net/license-library/CreativeCommonsAttribution20Austria	11	61314
https://www.dalicc.net/license-library/GNU_GPL_v3	12	60368
https://www.dalicc.net/license-library/BSD-4	13	60308
https://www.dalicc.net/license-library/GnuFreeDocumentationLicense	14	61218

Assuming a Gas price of 3 Gwei and assuming that 1 Ether has a value of 111 Euro, we get the prices shown in Table 5. Those results show us that we can expect that the cost will be acceptable for businesses (under 1€) and therefore, the non-functional requirements are met.

Table 5. Prices for interaction with the smart contract system.

Action	Price
Deploying contract	0.50657€
Licensing data with not existing license contract	around 0.20€
Licensing data with existing license contract	around 0.03€
Modifying License of data with not existing license contract	around 0.20€
Modifying License of data with existing license contract	around 0.02€

Finally, we must check if the implemented application also meets the functional requirements. Confirming these, we have ensured that the licensing attachment processes through the app are executable, in a user-friendly manner. As described in the implementation of the frontend, the user has the option of specifying a URI or selecting

a file, then retrieving the license information of the data, licensing the data or changing the license of the data, if he/she is allowed to. Since Infura is used as a web3 provider to retrieve the license information, the user does not need a DApp-enabled browser and hence, does not need to authenticate for retrieving license information. Now also the functional requirements have been met, so all the requirements that have been placed on the application are met.

5 Conclusion

A solution to explicitly attach licenses to data and content has been created and published as an open source, basing on the DALICC License Library. For this purpose, an approach using the Ethereum platform was designed and implemented to keep the system trusted, autonomous, distributed and decentralized. Thus, a licensing system has been created which preserves the integrity of the license information and also provides a high availability of that information. We also paid attention to usability during the implementation and tried to keep the learning curve flat. To do this, the user is guided through the licensing process step by step and unnecessary steps have been automated. We have also evaluated the costs for the user, identifying them as being acceptable for typical possible usage purposes. One limitation here is that we evaluated the simplest scenarios (one-step processes of license management): in more elaborated real life settings it would be necessary to study aggregated costs of multiple transactions (modifications of licensing, multiple files, etc.) typical of data publication scenarios. Further investigations may address the integration of the solution into practical settings, e.g. defining the authorities set up managing such traceable explicit licensing, as well as associated workflows and responsibilities. This may vary from domain to domain, and handled differently in different cases: some may be more centralized (e.g. insurance data, with few parties) and some more distributed (e.g. smart cities, with multiple parties).

Acknowledgements. This work is supported by the smashHit European Union project funded under Horizon 2020 (grant number: 871477) and by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) DALICC project (grant number: 855396).

References

1. Pellegrini, T., Mireles, V., Steyskal, S., Panasiuk, O., Fensel, A., Kirrane, S.: Automated rights clearance using semantic web technologies: the DALICC framework. In: Hoppe, T., Humm, B., Reibold, A. (eds.) Semantic Applications, pp. 203–218. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-55433-3_14. DALICC project: www.dalicc.net
2. Steyskal, S., Polleres, A.: Defining expressive access policies for linked data using the ODRL ontology 2.0. In: Proceedings of the 10th International Conference on Semantic Systems, pp. 20–23 (2014)
3. Lebo, T., et al.: PROV-O: the PROV ontology. W3C Recommendation (2013)
4. Buyle, R., et al.: Streamlining governmental processes by putting citizens in control of their personal data. In: Chugunov, A., Khodachek, I., Misnikov, Y., Trutnev, D. (eds.) EGOSE 2019. CCIS, vol. 1135, pp. 346–359. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39296-3_26

5. ADApp Statistics. https://www.stateofthedapps.com/stats
6. Panasiuk, O., Steyskal, S., Havur, G., Fensel, A., Kirrane, S.: Modeling and reasoning over data licenses. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 218–222. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_41
7. DALICC License Annotator. https://github.com/kilianhnt/dalicc-license-annotator
8. Antonopoulos, A.M., Wood, G.: Mastering Ethereum: Building Smart Contracts and DAPPs. O'Reilly Media, Newton (2018)
9. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**(2014), 1–32 (2014)
10. Buterin, V.: A next-generation smart contract and decentralized application platform. White paper 3.37 (2014). https://github.com/ethereum/wiki/wiki/White-Paper
11. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
12. Dworkin, M.J.: Sha-3 standard: permutation-based hash and extendable-output functions. Technical report (2015)
13. NIST: NIST releases Sha-3 cryptographic hash standard (2015). https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard
14. Infura. https://infura.io
15. Domingue, J., Third, A., Ramachandran, M.: The fair trade framework for assessing decentralised data solutions. In: Companion Proceedings of the 2019 World Wide Web Conference, pp. 866–882. ACM (2019)
16. Rinkeby Test Network. https://www.rinkeby.io