Consent Management Using Blockchain

Nathaniel Aldred, Luke Baal, Graeham Broda, Steven Trumble

***Abstract*—With the rise and fall of bitcoin, blockchain as a technology has come to the front of leading developments within the cloud. However, currency is not the only possible application. With blockchains built-in security and removal of the need for third party trust, blockchain has started to see some use within contract applications among other things. We propose another use for blockchain which involves consent management and Hyperledger Fabric to provide a service that allows end-users of this proposed service to control and consent to who manages their all or a partial set of their sensitive information.**

***Index Terms*—Blockchain, Consent Management, GDPR, Hyperledger, Privacy**

I. Introduction

WITH the introduction of the GDPR in europe, a large new set of restrictions and rules have been placed upon data usage and collection, with large penalties if a business fails to comply. The GDPR has greatly increased the scope of how all businesses that process EU citizens data are affected. Businesses must provide a clear description of what is involved in their consent agreements, as well as having an easy method of consent withdrawal. In consideration of a customers data rights, there are several subject rights. In an attempt to summarize the rights described in the GDPR policy, we will list them and give a small description. The first right, Notification of Breach, a company must notify customers as soon as a data breach were to happen. Right to Access is the right that a customer must be notified if there data is being accessed for processing. The Right to be Forgotten, is the customers right to revoke access of a company to their data. The GDPR also promotes privacy by design which we will describe in more detail in the following paragraph. Finally, the GDPR will have Data Protection Officers, a group of people who will ensure that companies are complying with the appropriate data policies [1].

Even outside of the EU, there are many reasons why businesses would want to adopt these rules. In Canada, surveys have shown that it is a potential business advantage to leverage these rules to further protect private and secure data, such as the Office of the Privacy Commissioner of Canada [Canadian67:online]. Secure data can be defined in several ways. Any personal information relating to a customer is considered confidential and secure, so no one should be able to access it without permission. Information in this category includes, but is not limited to: Photographs and ip addresses, contact lists, voiceprint biometrics used in voice recognition apps, location information. This information can be collected by a malicious entity, combining many pieces of info creates an overview and insight into a victims life. When rules are in place to protect this data, customers have been shown to feel significantly more confident in the company which has their data, and their capabilities to protect it. Some examples of rules in place to protect customers include app liability. App liability states that Under Canadas private sector privacy legislation, an organization is accountable for personal information that it collects, uses and discloses, meaning that a business has a capital interest in treating customer information with care, and protecting it.

All of these have large impacts on how software and data management systems are designed, as previously companies had no reason to get rid of data upon a users request.

This leaves developers who would like to develop software using appropriate privacy policies, with several choices. The first choice is to develop their own in house solution, developing their own privacy policy, and assuring customers that they are being upfront and honest in their dealings. This solution may not be adequate for many customers, as there is no way to ensure the company is handling their data as they say. The next option, would be to use a third party solution. This solution would need to have its policies decided by a government organization, and companies wishing to participate would need to be accredited by this organization. This would ensure that companies complied with the given policies, and that the policies could be made or altered by the elected party in accordance with the citizens wishes.

This leads us to our next problem, the system enforcing these policies must open and transparent about its privacy practices, as outlined by the Office of the Privacy Commissioner of Canada. Companies and users participating in this system will want to ensure that the history of their transactions is immutable, yet remain

private. Blockchain seems like an excellent way to implement several of these features. The Blockchain is an immutable ledger, that provides a history of transactions, but allows the details to remain private [2]. It is a peer-to-peer network that removes the need of a third-party to authorize transactions. This fulfills most of our requirements but does not allow for a governing body to set policy on these transactions. In order to do this, we believe a permissioned blockchain with immediate finality [3] such as hyperledger-fabric is needed.

In this paper we will be discussing the implementation of the second option: a third party consent management system, whose policy is decided by a government agency. We will mostly be discussing the technical implementation of such a system, unless its implementation is directly affected by government policy. In order to do so we will break this document into the following sections: Section II. Background and Related Work, we will go over what makes blockchain a good solution, what is hyperledger-fabric and what makes it necessary for our implementation, what are some related areas of research, and what are some similarly implemented systems and their use. Section III. Proposed Solution, where we will discuss the our design and implementation of our prototype. This section will be broken down into the following subsections as follows: III-A will be the assumptions made in our design, III-B will be the architecture, III-C is the implementation of our prototype, in section III-D we will discuss the possible scalability of our solution, and finally in section III-E we will discuss some of the challenges we have faced in developing our prototype, and what we did to solve them. In Section IV. Evaluation and Results, we will critically evaluate our solution, and discuss our results. Finally, in Section V. Conclusion and Future Work, we will discuss our accomplishment, and discuss possible future improvements and implementations.

II. Background and Related Work

A. What makes blockchain a good solution?

One of the main premises that blockchain is built on, is the fact that security built into it removes the need for the trust of a third party. The trust is placed upon a distributed set of actors which as the added benefit of also making it hard to traceback to a user from a permission record of theirs. This reduces the impact of trust since many of the actors have different interests making a malicious consent hard to coordinate. Also the immutability makes block a great solution for consent management as once a permission is granted it is recorded in the leger and becomes immutable, the way to revoke this permission is to add another record to the chain stating that it has been done so. It is also possible to keep transactions anonymous, despite them being announced publicly. The public keys can be kept private, and the public will only be able to see that there is a transaction, and will not know the who the user of the public key is [2].

B. What is hyperledger-fabric?

Hyperledger is self coined, as a distributed ledger technology(DLT) [4]. It is a implementation of a modular blockchain architecture. It provides many features that we find may be suitable for developing a possible implementation of a consent management systems. Hyperledger-fabric provides an identity management system, which is to say they require every user to be authenticated to participate and transact on the blockchain. This means that different levels of permissions can be applied to different users [4]. The extra level of permission management allows a centralized authority to allow certain users to transact with the blockchain in different ways. For example a company may only be given read permission, and only users can manipulate the global state. This may at first seem to go against the idea of blockchain, a peer to peer network. However, if we look at the perspective of a government trying to regulate a privacy policy in order to protect their citizens rights, then a privacy policy that can be reinforced through code and developed by lawmakers, is an ideal solution. Additionally, due to the use of authentication, compute intensive consensus algorithms such as the proof-of-work concepts are not needed, which makes hyperledger a more scalable option.

C. What current research is being done?

Most of the current research is being done in the medical industry, as patients data is very lucrative in this field, and their rights must be protected. Genestier et al [5] developed an application for a distributed consent management in using patients info. In this model, the user can give permission for 3rd party companies to collect information on the patient, whether this is the use of records, or real time monitoring of the patient. This company used hyperledger for many of the same reasons as we did. The second example of research being done is by Benchoufi et al [6] [7] who have proposed using a blockchain solution in clinical trial research. Apparently, there has been some trouble with ensuring stakeholders do not manipulate results, and ensuring that trial properly collect consent from trial

members, and properly inform the members of any new information and renewal of product given in testing. Finally Gammon [8] wrote an article describing the benefits of consent management using blockchain, and some similar technologies that already exist, which we will got into detail in the following section.

D. What similar products already exist?

Most of these products that will be mentioned in this section are based off of the value of medical data [8]. In longenesis [9], they use a decentralized system to provide a medical record marketplace. Selling your data is essentially exchanging value on the blockchain, which is what the blockchain excels at. Nebula Genomics [10], is under a similar branch, as they will purchase a users entire genome, especially if the patient has unique health conditions. A system, MedRec [11] that more aligns with Genestier et als [5] design, and is used for sharing medical records. Estonian e-health has also already implemented a blockchain system giving patients access control of their data [8].

III. Proposed Solution

A. Assumptions

The major assumption made is that our technology will be supported by a trusted organization. For example, ISO/PC 317 [12] Consumer protection: privacy by design for consumer goods and services, could provide an accreditation for a privacy policy management system. This accreditation would be similar to ISO 9001:20015 [13] which requires an organization to meet several quality requirements. Using this accreditation system and hyperledger-fabrics permission based system, we could require that companies wishing to participate in our blockchain are accredited. Our policy would need to include several requirements from the companies participating. Companies would need to comply with any updates to user permissions, ie if a user were to revoke access their phone number the company would need to delete it from their internal database. Or even better yet, the company would only every access the data on our database, updating the ledger in a request for a key, thereby alerting the user that their data is being accessed. Once a company has accessed a users data, they would be able to keep it forever. A development of a good privacy policy would entail the particulars of how that data is handled, and how to hold a company liable for it. This includes any breaches to the users data, and would motivate companies to leave the storage of data to the centralized database of our server. We are also under the assumption that users would not want their permission settings to be public to other companies. This means that given a permission asset, it must be impossible to determine what user that belongs to. The final assumption that we make, is that companies are not concerned with being connected to permission item, thereby reducing computational complexity.

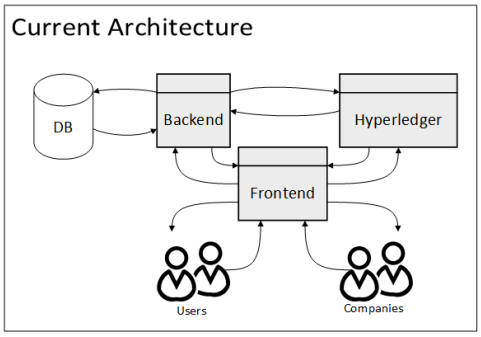

Fig. 1. System Architecture

B. Architecture

The overall Architecture of our solution can be described as having three main components: The blockchain network, the REST API consortium(composer REST API, our REST API, and frontend), and the external database. Figure 1 describes the current architecture of our application. As the reader can see, both the user and the companies need to register on the front end. The front end handles all the requests to Hyperledger, and all requests for data from the backend. These interactions will be described in more detail in the following subsections.

1) Blockchain Network: Hyperledger Fabric, which is the underlying blockchain is configured and managed by Hyperledger composer [4] [14]. Some configuration of Fabric is not done by composer and this is the network architecture. This is in regards to the organization, orderer, peer, and CouchDB modules. To handle this configuration, a basic sample network was used as it fills the needs of the prototype. The fabric network used is illustrated in [4, Fig. 2]. It involves one organization, which is responsible for the management of the blockchain network, which is used by Composer. It also uses one orderer node, which is responsible for the creation of the blocks that get put on the blockchain, this is abstracted from the peer nodes in order to improve performance [14]. The network also features one peer node, which is what hosts the ledger (ie. the blockchain). Finally a CouchDB node is used

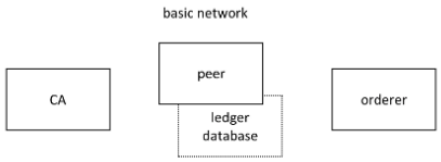

Fig. 2. Hyperledger Fabric Basic Network

as the datastore in order to manage data and allow data persistence.

2) Composer REST API, Our REST API, and Frontend: In order to perform operations on the network or to just query data, Hyperledger Composer allows for the generation of a REST API to handle requests [14]. For simplicity, the REST API uses an Admin credential card with full access to the network to perform requests. The idea being to use an external server/application as a gateway to this REST API. Our own REST API was developed in order to make use of the database (described later) and to facilitate authentication for the frontend. Finally a frontend was developed to allow users to register/login and add/edit permissions as well as view companies. Companies can also register/login to the frontend to view the data they have been given permission to see.

3) External Database: Another key issue, as previously mentioned, is implementing the Right to be forgotten within a blockchain system. The actual blockchain within Fabric is an immutable ledger, this means all versions of every asset are always going to be in the blockchain, with the most up-to-date versions being in the global state of Fabric [4]. This means storing the user's info in the blockchain is not viable, since once it was added, it could never be deleted, and might be visible to unauthorized parties. To solve this, a separate database is used to store the actual user info, which is why the permissions assets in the blockchain use boolean flags instead of the real info. A Mongo database was used as the database [15]. This database also generates a unique ID for each user entry, which is the user ID used to make pair key hashes described earlier.

C. Implmentation

1) Hyperledger Composer: The first question needing to be answered to implement this system was what should and should not be stored in the blockchain. As mentioned in our assumptions, users should not be able to be traced from their permissions, nor should companies be able to trace users they have to the permissions said users have set for other companies. It is still required, however, to be able to link users to their permissions entries, but only to authorized parties. Hyperledger Fabric allows for data to be represented as key-value pairs [4]. Assets are therefore required to be identified by a reference field, which will act as the key. For the permissions assets, the reference field is a hash of the users ID and the companys ID. The origins of these respective IDs will be described later. The assets also have a reference to the company they are associated with, which allows for the companies info to be retrieved upon querying the permissions, rather than just getting the companys ID and requiring a second query to get the name of the company. The remaining fields of the permissions assets are boolean flags which indicate whether the company is allowed access to the respective information. For example email: false, name: true, indicates the company is allowed access to the users name, but not their email address. The full permissions asset schema is presented in [Tab. I]. The other kind of data in the blockchain network are companies. It is assumed companies want their info to be readily available, so it was put into the blockchain, which better allows permissions to be linked to them. A company entry is simply some arbitrary info fields designed to inform the user about the company, as well as a unique ID, which is implemented with uuid version 4. This unique ID is the company ID used in the pair key for the permissions assets. The full company schema is presented in [Tab. II].

TABLE I
PERMISSIONS ASSET SCHEMA

Field	Data Type
Pair Key	string
company	string
name	boolean
email	boolean
phone	boolean
address	boolean
age	boolean

TABLE II
COMPANY PARTICIPANT SCHEMA

Field	Data Type
companyId	string
name	string
description	string
address	string
phone	string
email	string

2) Database, REST API, Frontend: The database is responsible for storing the login information, which is the users email or the companies name, as well as the password hash (plus a salt) of the user/company. Our REST API is then used to interact with this database. The REST API is responsible for handling user/company registration and login. Users enter their info on registration to have it added to the database. The info they enter follows the schema described in table III. Companies, upon registering, only need to give their company name and a password, which is described in table IV. Once they have logged in after registering, they are prompted to enter their company info, which, along with their company name, is added to Hyperledger. Once a user logins, they can view the list of companies in the system, so they can better identify the companies they want to give permissions to. They can then add permissions setting for a given company, as well as edit existing ones. Finally, users can also delete their account, which sets all permission settings to all be false, deletes their info from the database, and then logs the user out. It is worth noting the user warning and is prompted to type DELETE and click confirm before any deletion occurs. The Hyperledger Composer REST API does allow for delete requests, however, we opted for setting all permissions to false, for two reasons. Firstly, the underlying blockchain will always have the history of the permission settings, so true deletion is not possible. Secondly, companies would have to recognize that permission entries were deleted in order to determine which set of info to delete. By setting all permissions to false, a message can be given to the companies telling them that the entry with the given pair key has been deleted, so they recognize that they must delete the information they have with that pair key. In order for companies to see the data they have access to, first the frontend sends a request to the REST API. The REST API then queries the Composer REST API for all permissions assets that have the same company ID as the currently authenticated company. The REST API uses the permission flags to determine what user info the get from the database and to send back to the frontend. Upon receiving the information, the frontend then renders each permission entry separately. The pair keys can be used by the companies to identify the individual entries.

TABLE III
USER SCHEMA

Field	Data Type
id	string
name	string
email	string
password	string
address	string
phone	string
age	number
pairkeys	string array

TABLE IV
COMPANY DATABASE SCHEMA

Field	Data Type
id	string
name	string
password	string

D. Scalability

1) Hyperledger Fabric: The underlying blockchain software used, Hyperledger, allow our program to be highly scalable. Hyperledger Fabric, which is the actual blockchain system, operates using several docker containers [4]. Furthermore, having more peers improves the availability, given each peer hosts an instance of both the ledger and any chain code. This availability, however, is limited to querying. Any updates to the ledger require all peers to reach consensus to ensure the transaction is valid. Adding more peers, therefore, will slow down updates, however, it is also making it harder for the network to be fooled by a malicious user and prevents the system from having a single point of failure [4].

2) Hyperledger Composer REST Server: The REST server generated by Composer can also be used as a Docker container. Being a REST API, the server can simply be duplicated and have multiple instances of itself running to handle more requests. A common way to do this efficiently would be to use either Docker swarm [16] of Kubernetes [17]. Likewise, our REST API can be similarly scaled using Docker swarm or Kubernetes. The frontend can also be served by our REST API, allowing it to have more availability too.

E. Challenges and Solutions

There were many challenges in developing this solution, which mainly fell in to three main categories: overhead of learning concepts and technology related to blockchain, technical issues dealing with the various implementations of that technology, using the previous two challenges as constraints and still solving our original objective. The solution to the first challenge

was just due diligence on our part. We researched any problems or misunderstandings we had in order to fill out our understanding of the subject. For example, if all transactions are transparent, how will a user remain private? It turns out this can easily be done by keeping the public key anonymous [2]. We also had to research several development platforms and attempt a preliminary design in order to flush out possible problems with the platforms. For example, Ethereum, did not have the permission based control we felt we needed. The next problem we ran into was understanding and developing on the platform we decided on: Hyperledger. We realised right away that developing on Hyperledger would be a very grueling and painstaking endeavour, if we built from the ground up. We found a solution to this by using Hyperledger-Fabric, as a framework for building on. However, this still required vast amount of knowledge, and quite a large amount of work to build the appropriate network for our application, and that is without even taking our original objective into consideration. In order to circumvent this problem, and focus on our objective, we found Hyperledger-Composer, which uses the Hyperledger-Fabric framework, but is easily configurable. Obviously, this caused its own set of problems. This is because such a high level of abstraction requires a large knowledge of the underlying components when errors and bugs start to occur during development. One instance of this was when we found that hyperledger was not responding. When we traced the error through the extensive system we found that it was possibly because of GRPC connection timeouts. Obviously a more in depth knowledge of the system, built over time of development on it, would give us a better idea of what was going wrong, and how to troubleshoot it. The end result, was that it there was a comma missing in the config file that relates to the GRPC connection. The third challenge we faced, was a set of problems that came with our actual objective. We needed to keep a reference to users in permission items, without having them be traceable to the user. The solution to this was that we kept an id of permission items in a hash of the user id and the company id. In this case, the user id must be known to trace the user to the corresponding permission item, which can only be found in our database. The second issue in this set, was implementing the right to be forgotten to allow users to delete their account and data. This is a problem when it comes to blockchain and more specifically Hyperledger Fabric, given the ledger is an immutable history of transactions [4]. The solution we found to this was to update all permission to false, thereby requiring a change in the global state, and then deleting the user from our database. This solves the problem, because once the user id is gone, there is no way to trace it back to the user.

IV. Evaluation and Results

The best way to evaluate this project is according to three main metrics. Those metrics being: Performance, Usability for Users, and Usability for Companies. In regards to performance, the POST/PUT requests to composers REST api seem very slow. The requests tend to take roughly 3 seconds, even when all nodes are on localhost. Hyperledger-Fabric is a permission based blockchain, and is able to use more computationally efficient methods than proof-of-work to reach consensus, however, these methods still take a considerable amount of time. All transactions must reach a consensus by a pre-set number of nodes in order to change the global state. The second metric, Usability for Users, seems to fill many of the requirements set out by the Office of the Privacy Commissioner of Canada [18] in their design for privacy. The interface is simple, and the user can give permission just as easily as they can remove it. The user can browse companies in a single list, and add/edit permissions as they wish. The third metric, Usability for Companies, seems to also be quite successful. The interface is very basic, and not much work is needed beyond writing a script to get whatever info they can. They can use our rest API to automate the process, or just use our frontend as their needs dictate. Overall, we feel this project is very successful in building the groundwork for a third party regulated consent management system, and feel there is a great deal of applications this can be used for.

We also evaluated our project based on the Privacy by Design 7 Foundational Principles as outlined in [19]. The 7 foundational principles are as follows.

1) Proactive not Reactive
2) Privacy as the Default Setting
3) Privacy Embedded into Design
4) Full Functionality – Positive-Sum, not Zero-Sum
5) End-to-End Security
6) Visibility and Transparency
7) Respect for User Privacy – Keep it User Centric

What follows, therefore, is our evaluation of these principles.

A. Proactive not Reactive

In order for a company to be able to be a part of this system they must be accredited first by a set of criteria so that way we can validate their processes in terms of handling the data that is allowed to them through the

system. This way we can assure privacy and security of the data before it is given out, instead of afterwards when they have already been given the data.

B. Privacy as the Default Setting

Privacy as the default settings, means a users info is private by design and if they want to make it available, they need to explicitly allow it [19]. For example, for a social networking site, users cannot see you information unless you allow other users to see it, rather than the default being they can see it and you have to explicitly turn it off. With this definition in mind, our system follows this principle, given that a company has no knowledge whatsoever about a user until they add permissions for that company. Even then, the company can only see the information theyve been given access to by the user.

C. Privacy Embedded into Design

Our system was built with privacy and security as top priorities. The system was designed from the ground up using privacy and security principals to ensure that privacy was embedded into our design. Each element within our architecture keeps these ideas in mind, and we ensured that no sections violated these principals.

D. Full Functionality – Positive-Sum, not Zero-Sum

This principle is based off the assumption that there is typically a trade off between privacy, and security [19]. In our system, we take this principal to heart, and have tried to create the highest amount of security and privacy possible. With our design, the data is only able to be accessed with user permission, and can only be understood with user consent, making our system meet both the requirements for security and privacy.

E. End-to-End Security

This principle involves the assurance that all aspects of the system are secure [19]. Our system, being a prototype, does not meet this fully, given there is little security between the backend and database, as well as between the frontend/backend and the Composer REST API. This was done for convenience and the technologies involved all allow for much better methods of security, which could certainly be utilized. Therefore, even though our system as it stands does not meet complete end-to-end security, it would be very reasonable for it to meet this principle if needed.

F. Visibility and Transparency

The idea being visibility and transparency is to ensure a system is secure simply because no one knows, except the creators of the system, how it works [19]. By making the system open source for example, you allow others to verify the security of the system [19]. For our project to follow this principle we would need to make it open source. All the tools we used, specifically Hyperledger, are all open source tools, allows this system to follow this principle.

G. Respect for User Privacy – Keep it User Centric

The idea for this principle is that the user should be involved and have a significant say in their own privacy [19]. Our system design directly implements this idea by giving the user full control over their data and privacy. The user is the only one who can access their data without explicit permission, maintaining this principal.

V. Conclussion and Future Works

A. Conclussion

Based on the work above, we believe that we have created a system that meets the guiding principles set down by GDPR in regards to data privacy. Blockchain presents inherent issues in regards to the removal of data, however our system has provided a secure way to avoid putting blockchain in the first place, while maintaining the privileges of data access in the secure blockchain. Our system makes sure that privileges are kept secure and anonymous on the blockchain, with no way of retrieval after the customer removes their data. this provides a succinct way to protect customer rights, while maintaining the integrity of the blockchain.

B. Future Work

While our solution is comprehensive, there are many areas where it can be scaled up and improved. We have several major areas where future work can be conducted. The first area that we can work on, is the scaling of hyperledger. By design, our solution should be able to scale to a fairly large size. While doing so, it must remain responsive, and practical. This will need to be simulated and tested as best as possible. The next area is to provide access cards for a user/company to set their permissions with. This will make it easier to authenticate with the composer REST server. Another addition, would be the use of a second key pair in transactions as an additional firewall. This would prevent the ability to link transactions to a single

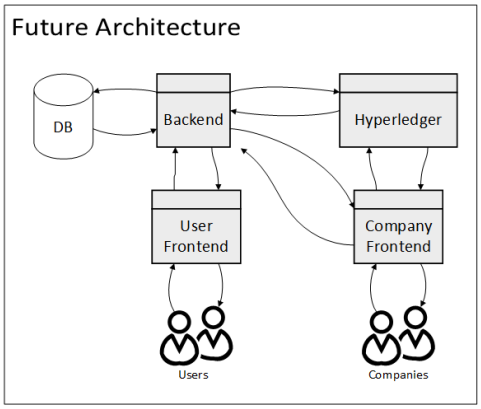

Fig. 3. Alternative System Architecture

public key [2]. The final area we would like to see work done, is that of developing a standard privacy policy, and ensuring that any companies partaking in the blockchain consent management system are properly accredited.

We also considered an alternate architecture, where each the user would have their own application to interact with hyperledger and the data requests from the backend. This would be to allow a company to interact with our application through the use of an API as illustrated in Figure 3. Companies would be able to integrate our solution into their own applications how they would like, and have all interactions controlled programmatically.

In addition, we have identified 2 potential use cases where our solution would be able to be used in real world applications. First, our solution could be used in medical and clinical trials. Using our solution would ensure stockholders do not tamper with trial results and ensure test subjects are informed on every step. This would enhance patient privacy and control over their data, while simultaneously guaranteeing the validity of the data being used in the experiments.

The second use case we have identified, is having a company which customers can use to store their data, which third part companies would have to go through to access their data. Instead of each company holding on to their individual copy of the data, the customer data could be consolidated in a single location, with the customers in control of permissions. This of course bring us back to the idea of a committee ensuring that any company using the database, is properly accredited by a standards organisation.

References

[1] "Key changes with the general data protection regulation eugdpr," https://eugdpr.org/the-regulation/, (Accessed on 04/09/2019).

[2] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[3] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, 2016.

[4] "Why hyperledger fabric? fabricdocs 1.0 documentation," https://fabrictestdocs.readthedocs.io/en/latest/whyfabric.html, (Accessed on 04/09/2019).

[5] P. Genestier, S. Zouarhi, P. Limeux, D. Excoffier, A. Prola, S. Sandon, and J.-M. Temerson, "Blockchain for consent management in the ehealth environment: A nugget for privacy and security challenges," *Journal of the International Society for Telemedicine and eHealth*, vol. 5, pp. GKR–e24, 2017.

[6] M. Benchoufi, R. Porcher, and P. Ravaud, "Blockchain protocols in clinical trials: Transparency and traceability of consent," *F1000Research*, vol. 6, 2017.

[7] M. Benchoufi and P. Ravaud, "Blockchain technology for improving clinical research quality," *Trials*, vol. 18, no. 1, p. 335, 2017.

[8] K. Gammon, "Experimenting with blockchain: Can one technology boost both data integrity and patients' pocketbooks?" 2018.

[9] "Longenesis," http://longenesis.com/, (Accessed on 04/09/2019).

[10] "How it works — nebula genomics," https://www.nebula.org/how_it_works, (Accessed on 04/09/2019).

[11] "Medrec," https://medrec.media.mit.edu/, (Accessed on 04/09/2019).

[12] "Iso/pc 317 - consumer protection: privacy by design for consumer goods and services," https://www.iso.org/committee/6935430.html, (Accessed on 04/09/2019).

[13] "Iso 9001:2015 - quality management systems – requirements," https://www.iso.org/standard/62085.html, (Accessed on 04/09/2019).

[14] "Hyperledger composer," https://hyperledger.github.io/composer/latest/introd 2018, (Accessed on 04/09/2019).

[15] "Mongodb," https://www.mongodb.com/, 2019, (Accessed on 04/09/2019).

[16] Docker, "Swarm mode overview," https://docs.docker.com/engine/swarm/, 2019, (Accessed on 04/09/2019).

[17] T. L. Foundation, "Kubernetes," https://kubernetes.io/, 2019, (Accessed on 04/09/2019).

[18] "Canadian businesses and privacy-related issues - office of the privacy commissioner of canada," https://www.priv.gc.ca/en/opc-actions-and-decisions/research/explore-privacy-research/2012/por_2012_01/, (Accessed on 04/09/2019).

[19] A. Cavoukian, "Privacy by design: the definitive workshop. a foreword by ann cavoukian, ph. d," *Identity in the Information Society*, vol. 3, no. 2, pp. 247–251, 2010.