Architecture to Manage and Protect Personal Data Utilising Blockchain

Jens Leicht and Maritta Heisel
paluno - The Ruhr Institute for Software Technology, University of Duisburg-Essen, Duisburg, Germany

Keywords: Data Protection, Privacy, Blockchain, Data Management.

Abstract: Many Internet users employ a multitude of online services. Many services require the same data to be entered and users enter it repeatedly. Instead of entering information for every new service a user wants to use, we propose a system that allows users to simply share a set of information with any service they want to use. The information is entered once and stored in a distributed storage system. Users can easily share the data with any service provider, in order to use a service. Our proposed system makes use of the distributed ledger, provided by blockchains, to manage access rights. By taking the data away from the service providers, the personal data is also protected against unwanted data leaks.

1 INTRODUCTION

A tremendous amount of online services access personal data from their users. The data provided by the users often must be entered repeatedly, due to several services requiring the same information. As an example, users make purchases at multiple e-commerce providers. Every time they submit an order in a new online shop, shipping and payment information needs to be re-entered. However, this is error prone, due to typing errors, and might be considered annoying. In this paper we propose a system called *Data Protection and Management System* (DPMS) that allows users to manage their data with a decentralized system. Data only must be entered once and can afterwards be accessed by service providers. Rights management and logging of data access are realised with a distributed ledger in a blockchain (Nakamoto, 2008). For the storage of users' data, a distributed hash table (DHT), with additional redundancy is used.

In a real-world scenario our system could for example be used by online merchants. Every merchant could be providing and using our DPMS interface and users can store their billing and shipping information in the DPMS. Every time users place an order, they can allow the merchant to access their shipping and billing information and thus do not have to enter their information repeatedly.

The paper is structured as follows: At first some necessary background information is presented in Section 2, before we introduce the architecture of our proposed system in Section 3. In a next step, the encoding of access policies is explained in detail in Section 4. Afterwards, in Section 5 we explain some procedures from our system, that allow a better understanding of how it is going to work. Next, we present a short discussion of our system in Section 6 and related work in Section 7. Finally, Section 8 presents a conclusion and future work.

2 BACKGROUND

The system that we propose utilises multiple distributed techniques, which are presented in a short overview in this section. First, the blockchain technology and its use for the proposed system is explained, followed by a brief introduction to distributed hash tables.

2.1 Blockchain

Blockchain technology is mostly known through the crypto-currency system bitcoin (Nakamoto, 2008). The bitcoin system uses the ledger functionality of a blockchain to store transactions in an immutable manner. Meaning that transactions cannot be altered, after they have been verified by the participants of the blockchain. These participants are called miners, because they get rewards for the computing power they spend on the verification of the transactions. To remove the trust needed in a single miner, all miners verify the transactions.

Leicht, J. and Heisel, M.
Architecture to Manage and Protect Personal Data Utilising Blockchain.
DOI: 10.5220/0007724203400349
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 340-349
ISBN: 978-989-758-375-9

The system is called blockchain, because all transactions are stored in so called blocks, and each new block references its predecessor, thus creating a chain of blocks. Each block consists of the hash of the preceding block, the root of a hash tree of the transactions stored in that block and the transaction data itself. The transaction data states how much currency is transferred from one address to other addresses. The chaining ensures that past blocks cannot be altered without replacing all the following blocks, because the hashes contained in the following blocks would not match the content of the manipulated block.

This, however, requires several blocks to be created, after a transaction has been stored, to prevent miners with high computing power to manipulate the block and all currently following blocks. Only after a certain number of blocks has been created, following the block containing the transaction, a transaction is considered persistent.

Two distinct methods for the creation of new blocks have been developed. On the one hand the *proof-of-work* method can be used, which requires miners to spend their computing power and thus high amounts of electricity on computing a high number of hashes. This technology is used by the bitcoin system. On the other hand, the *proof-of-stake* method (Kiayias et al., 2017) can be used, which does not require the miners to calculate many hashes, but instead allows a random miner to create a new block by just calculating one hash. Besides the difference in energy consumption (one hash compared to millions of hashes), the time needed to create a new block is reduced by the *proof-of-stake* method.

When two blocks are created at the same time, a so-called fork happens. The blockchain gets extended at two ends in parallel. After some time, the longest chain is accepted as the actual chain and transactions from the forked chain are placed in later blocks of the accepted chain.

The addresses used in transactions are the public part of an asymmetric key pair, which is generated by the users of the blockchain. To create new transactions the user needs to use the private part of the key pair, to sign previous transactions. There is no need for certificate authorities (CAs) as the key pair is only needed on the users' side and is generated by the users themselves.

Some blockchain systems provide a so-called faucet, that provides some amount of free crypto-currency to users, who are interested in using the system.

Although blockchain technology is widely known through the bitcoin hype, it can also be used outside the bitcoin and crypto-currency territory. We propose a use case for a blockchain in a privacy and data management context. Our system is based on the work from (Zyskind et al., 2015a), which is further explained in the related work described in Section 7.

2.2 Distributed Hash Table

A distributed hash table (DHT) is a system that allows efficient localisation of data in a distributed and decentralized peer-to-peer system. Nodes and data are addressed through hashes.

DHTs can, for example, be used for file sharing systems, distributed file systems or content distribution systems. An example DHT protocol is Chord (Stoica et al., 2001), and Kademlia (Maymounkov and Mazières, 2002) is an implementation of a DHT. Based on these protocols several applications have been developed, for example RetroShare[1], a secure communications application, or GlusterFS[2], a distributed file system.

Through distribution of the data and removal of a central authority, the users do not have to trust a single party in maintaining the confidentiality, availability, security and integrity of their data. Additionally, redundant storage, on multiple nodes, decreases the risk of data loss.

3 SYSTEM ARCHITECTURE

Figure 1 shows the architecture of our proposed system. The DPMS consists of a *data manager API*, a *DHT*, a *blockchain* and a *naming service*. A *service provider* is providing a *service*, which is used by a *user*.

3.1 API: Data Manager

This is the main component of our proposed system, which manages access policies on the one hand and user data on the other hand. This part of the system could be implemented as a small library that can be used in applications and web services. In order to manage access to their data, users need to have their private key, which is used by the blockchain system to store transactions containing the access policies.

The data manger is an application programming interface (API) provided to service providers, who use the provided API to allow their users to make use of the DPMS.

[1] http://retroshare.net/

[2] https://www.gluster.org/

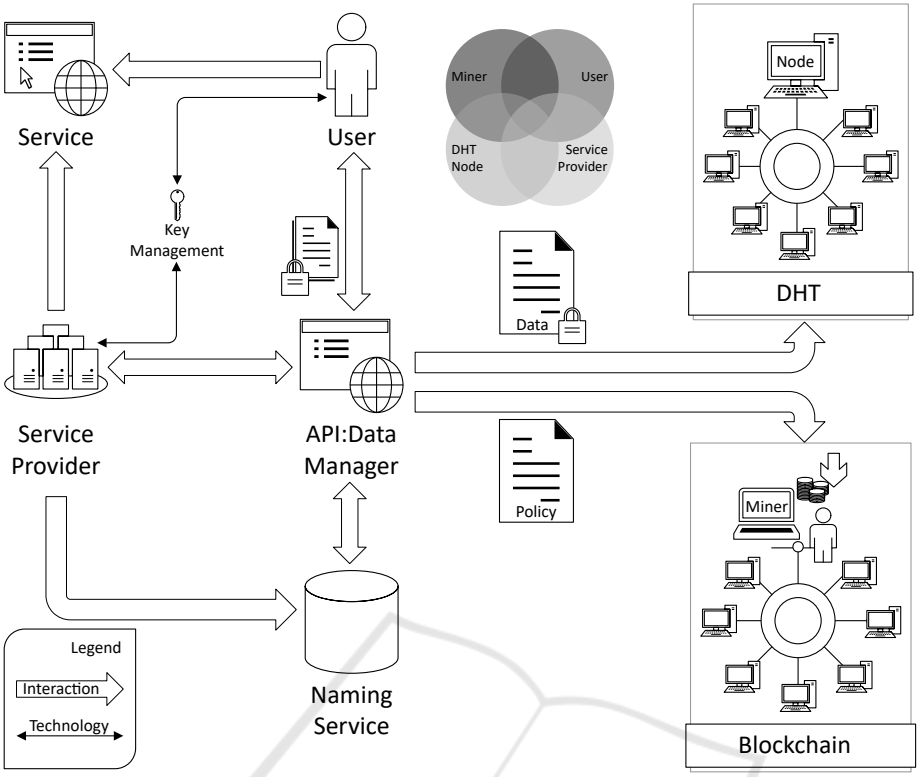

Figure 1: Architecture of the DPMS.

However, the data manager also has to be provided as a standalone web service or application to provide users a service-provider-independent access to the stored data and policies. This is necessary, especially if users want to withdraw access rights from a service provider, who stopped providing the data manager to the users. The application and web service can be hosted on the DHT to avoid additional costs for hosting services.

3.2 Blockchain

The blockchain, visualised as computers interacting in a network, is powered by miners. A possible way of powering the blockchain is to offer incentives, like a crypto-currency, to miners that are in no other way concerned with the DPMS. This however induces costs that need to be covered by users or service providers, which may lead to bad acceptance of the overall system.

Therefore, we suggest powering the blockchain from computing power provided by users and service providers. Both parties do not need additional incentives to be willing to provide some computational power, as they are benefiting from the running system.

Additionally to storing access policies, the ledger is used to log access to the data, and even attempted access can be logged. This enables users to see who accessed and who tried to access their data.

Instead of the proof-of-work method used in the bitcoin blockchain, we propose to use the proof-of-stake method. This is due to the massive energy consumption required for proof-of-work mining and the speed at which new blocks can be created.

3.3 Naming Service

Due to its nature the blockchain does not reveal who is behind a public key. So, users can only see which public key tried to access their data, when looking at a logging transaction. However, a DNS (dynamic name service) like system can be deployed, which reveals the service provider that relates to a given public key. The naming service should only be used for service providers, in order to preserve users' anonymity and thus improving their privacy. Only service providers that have access to users' data can identify who is related to a public key, any outside party cannot assert the relation between public key in the blockchain and user.

The naming system is enforced by only allowing access requests from the owner of the data and registered service providers. Any public key, that is not registered in the naming service, is prohibited to

access any data. By expanding the functionality of the DHT, the naming service could be hosted and enforced by the DHT nodes.

Even service providers' privacy can be protected, by only allowing name lookups by users, whose data has been accessed/attempted to be accessed by a given service provider.

3.4 Distributed Hash Table

Parties that are already involved in our system, e.g. by using it to access or store data, can become participants in the DHT. These parties do not need any further incentive to participate, because they already benefit from using the system.

Our proposed system is not yet implemented, but we provide some ideas on how it could be implemented. The DHT part of our system could be implemented using the chord protocol (Stoica et al., 2001). An alternative could be Kademlia (Maymounkov and Mazières, 2002). Both protocols provide some basic features for a DHT.

However, the DHT implementation needs to be adjusted so that it provides redundancy over multiple nodes. The access control also needs to be implemented into the DHT system. The DHT needs to make sure that a service provider is registered in the naming service and has a current access granted transaction in the blockchain.

Another important thing to mention is the fact that a DHT node should not have access to the data it is storing. Although the data itself is encrypted, a DHT node could possibly have access to the encryption key. By restricting the node's access to the stored data, a higher level of security can be provided. This could be achieved by storing the data in an encrypted storage, not providing the key to the node host.

The DHT implementation should have small computing needs, as this would allow a deployment on devices like routers at the users' homes. These devices are already running all day and are capable to participate in a DHT. BitTorrent[3] clients like *transmission*[4] already make use of these always online devices. *Transmission* is available for various router models.

3.5 Key Management

One of the most important parts of our system is the key management. Users create a set of information and encrypt it using *symmetric* encryption. *Symmetric* encryption is used, to enable multiple service providers to decrypt the same data. If an asymmetric algorithm was used, only one service provider, the one providing the public key, would be able to decrypt the data. This would stop our system from reducing the recurring entering of the same information, because the data would have to be encrypted for every service provider separately.

For service providers to be able to access the stored data, they must be able to decrypt the stored information.

We suggest that the user creates a secured connection to the service provider and then transmits the encryption key through the encrypted connection to the service provider. Once the service provider gains access rights for the data, it can be downloaded from the storage system and decrypted to access the information.

When users withdraw access rights or get informed about an encryption key disclosure, they need to be able to withdraw the data, that was encrypted with the key, from the system. We propose to add a functionality to the DHT that allows users to revoke data from the system. However, this functionality should not just delete the data, since a user may not have a local copy of the data at hand, which is needed for replacing the data, after using a new encryption key. We suggest that the DHT nodes, storing the encrypted data, encrypt it with a node specific key. Only when the owner of the data wants to access it, the DHT decrypts the data and returns the original encrypted data. This stops anybody other than the owner to access the data. After downloading the data, the user can request a deletion of the secured data and upload the newly encrypted data, afterwards.

In cases where the re-encryption of the data is not needed instantly (no key breach occurred) the user can download the original data, delete it and then re-upload a newly encrypted version.

The DHT re-encryption approach mentioned above allows the system to automatically react to data breach notifications from service providers, by automatically re-encrypting all data that was shared to that service provider. This ensures privacy even when data breaches at service providers occur.

To further protect data integrity, the encrypted data can be enhanced with a signature of the user, to make sure that the stored data originates from the user itself.

3.6 Participants

The Venn diagram in Figure 1 shows the participants of our system and illustrates that all four

[3]A peer-to-peer file sharing service using a DHT (http://www.bittorrent.org/)

[4]https://transmissionbt.com/

groups can overlap. Users sharing their data via the DPMS can voluntarily provide computing and network power to host a DHT node or perform mining on the blockchain, however they are also free to just use the service, relying on other participants hosting it. As mentioned earlier, always on-line devices like routers could be utilised by voluntary participants to strengthen the network.

3.7 Operation

Personal data is handled only in an encrypted state, protecting the user's privacy as neither the data manager nor participants of the DHT can access the data.

The policy containing access rights is encoded into blockchain transactions and committed to the chain (cf. Section 4). Users can provide access to a piece of their data by adding a service provider to the access list of that data and sharing the encryption key with that service provider, instead of providing the same data repeatedly to every service they want to use. Example procedures for data creation and access are explained in Section 5.

Both users and service providers interact with the data manager API, users to store data and manage access rights, and the service providers to retrieve data, that has been shared by users. Service providers also have to register at a naming service.

In order to enhance the privacy of the users, they can create multiple key pairs when using the system. This allows users to share data with different services without service providers knowing what other services the users are employing.

4 TRANSACTION ENCODING

The access policy is encoded into transactions on the blockchain. As these transactions can transfer crypto-currency from multiple addresses to others, the crypto-currency is used to encode the policy. The currency consists of coins (1c) and these coins can be divided into 100 million smallest units (100.000.000s). First, we describe a policy transaction, which encodes access rights, followed by an access transaction, logging a service provider accessing the data. As our system uses a custom blockchain the currency has no actual value and no established name.

4.1 Policy Transaction

Table 1 shows the structure of a transaction, encoding the access rights of two service providers to the user's data. The general idea is based on work by (Zyskind et al., 2015a). However, our system does not require shared identities and instead addresses each entity with its own public key.

A transaction is indexed with a transaction id (*txid*), which is calculated as the double hash of the transaction itself. It is possible that two transactions have the same double hash, which makes the index unusable. To circumvent collisions the output at position six can optionally be added to the outputs, changing the hash of the transaction. In a non-colliding case, this output is not present. The symbolic value of this output is one smallest unit and it is sent to a sink address, which can output coins through a faucet. The faucet provides users with free coins in order to allow them to use the system.

The input of the transaction is the previous transaction that granted the user coins, for example from the faucet to the user, or a previous policy transaction. The input transaction is signed by the user issuing the policy transaction. The value of x is provided by this previous transaction.

The first two outputs are symbolic and encode the SHAKE256 hash (Dworkin, 2015), with a length of 320 bits, of the encrypted data, which is used to address the data in the storage system and to check the integrity of the data. Both outputs are provided with a symbolic value of one smallest unit each.

Table 1: Policy Transaction: Structure and contents of an example blockchain transaction, encoding an access policy for two service providers.

txid: h(h(transaction))			
Input	Value	Output	Value
user signed previous transaction	x	0: bytes 1-20 of shake256(data,320)	1s
		1: bytes 21-40 of shake256(data,320)	1s
		2: service provider 1's address	1c
		3: service provider 2's address	1c
		4: user's address	x-*n*c-1.5c-2s/[x-*n*c-1.5c-3s]
		5: logger address	1.5c
		[6: sink address]	1s

The table shows an example structure for two service providers, but it can be easily extended by just adding more outputs for more service providers. Each service provider is granted one coin, which allows for 50 million access transactions to be created before the user needs to refresh the policy transaction. This number is less than the 100 million smallest units because of the access transaction encoding described in the next section. This value could also be adjusted to create temporary access to the data by using smaller values. If two smallest units were granted, a one-time only access could be encoded.

The output to the user's address (4) is used to preserve the user's remaining coins for later use in other policy transactions. The value transferred to this address is calculated based on the input value x, the number of service providers in the policy (n) and whether the sink output was needed.

Output 5 transfers one and a half coins to a logger address, that is used by the DHT to create logging transactions in cases where a service provider tried to access the data without permission. The amount transferred to the logger makes 50 million logging transactions possible.

4.2 Access Transaction

Table 2 shows the structure of an access transaction, encoding the access of the data by a service provider. The input is either the original policy transaction, in case the transaction is the first access transaction after creation of the policy transaction, or the last access transaction by the service provider for this data. The value of x depends on the amount received/remaining from the previous transaction.

The first two outputs contain the *txid* of the original policy transaction with some additional padding. Both outputs have symbolic values of one smallest unit. The last output (2) returns the residual value to the service provider. This is used in the next access transaction that the service provider needs to create for the data.

Additionally to the outputs shown in Table 2, a sink output can be added in the case of a transaction id collision, like output six from Table 1.

5 EXAMPLE PROCEDURES

This section describes different procedures of our system. First, we explain how the user can add data and corresponding access rights to the system. Afterwards, we explain how a service provider can access the data that a user shared with the service and how the system protects user data from unwanted access.

5.1 Sharing Data

In this section we describe two methods for users to provide access to their data. First, we explain the initial storage of the data in our system, followed by an update of an existing access policy to share existing data with a new service provider.

5.1.1 Data Storage

Figure 2 shows the messages that are passed, when a user tries to access a service that needs some information and the user decides to store the data using the DPMS.

First, the user requests a service from a service provider. The service provider asks the user to supply all needed data and provides the user with the public key (*pubKey_sp*) of the service provider, which is used in the access policy on the blockchain.

As users are using the DPMS, they create a symmetric encryption key and share it with the service provider via an encrypted connection. Users also provide their public key (*pubKey_u*), from the blockchain, to the service provider, which is used as the identifier of the user.

Using the encryption key, the user encrypts the data that needs to be provided and creates an access policy that states that the service provider is granted access. To specify this policy the user provides the public key of the service provider to the data manager. The user also provides the transaction that sent some blockchain currency to the user's public key,

Table 2: Access Transaction: Structure and contents of an example blockchain transaction, encoding the access of data by a service provider.

txid: h(h(transaction))			
Input	Value	Output	Value
signed previous transaction	x	0: bytes 1-20 of txid of permission transaction	1s
		1: bytes 21-32 of txid of permission transaction + 8 bytes padding	1s
		2: service provider's address	x-2s

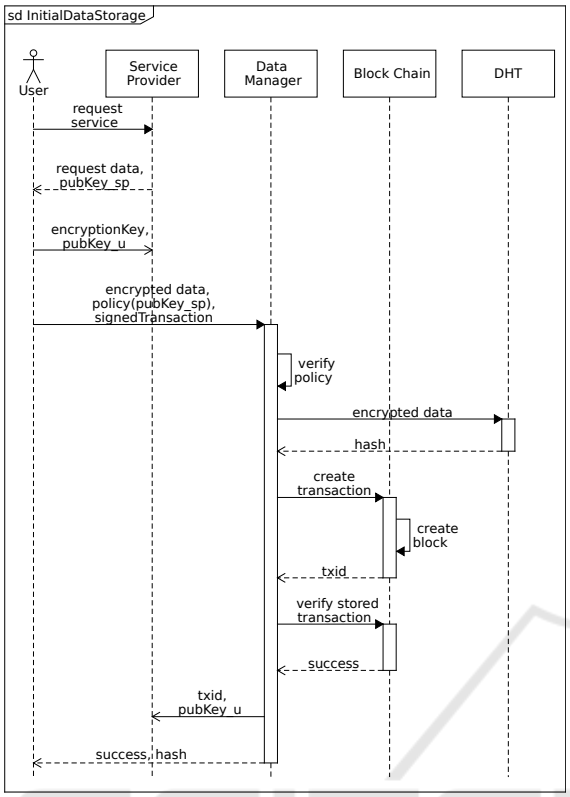

Figure 2: Sequence diagram of the initial storage of a user's data and corresponding access policy.

signed with the user's private key. This transaction can be another access policy or a transaction from the blockchain faucet.

The data manager first verifies that the policy is formatted correctly and checks that the service provider's public key supplied is registered in the naming service. Both steps are part of the policy verification in Figure 2.

After successful verification, the data manger uploads the encrypted data to the DHT and receives the hash of the data, which is also the address needed to access the data in the system. Using this hash and the verified policy, a transaction is created in the blockchain (cf. Section 4). This transaction is then processed by miners and stored in a block.

Once the transaction is stored, the data manager verifies the transaction by waiting for enough new blocks to be created, thus checking that the transaction is persistent in the blockchain. The time spent waiting depends on the block creation time of the blockchain, which can be several seconds per block, and the number of blocks needed for a transaction to be considered persistent. The smallest block creation time possible still needs to be evaluated.

After the persistence is verified, the data manager informs the service provider about the *txid* for the user with the supplied *pubKey_u*. Finally, the user is informed about the success in sharing the data and is supplied with the hash of the stored data.

5.1.2 Access Permission

Similar to the initial storage of the data, users can share previously uploaded data with new service providers. The first three steps of the process are the same as for the initial storage.

Users request a service from a service provider that they did not use before. The service provider requests some data and provides its public key (*pubKey_sp*) to the user. Users share the encryption key for the data, that was created when initially uploading the data, and their public key with the service provider.

Instead of handing the encrypted data to the data manager, just the hash of the encrypted data is needed together with the updated access policy. The updated policy contains all previously permitted service providers and the one that should be allowed access. Additionally, the user provides the signed transaction that sent some blockchain currency to the user's public key, similarly to the initial process from the previous section.

The data manager verifies the policy and that the service provider is registered in the naming service, before creating the new transaction on the blockchain. Miners then create a block containing the transaction. The data manager verifies that the transaction is persistent on the chain and then informs the service providers from the original access policy, as well as the newly added service provider, about the *txid* for the policy transaction in combination with the user's public key (*pubKey_u*). Finally, the user is informed about the successful adaptation of the access policy.

5.2 Data Access

This section describes two cases of the data access communication. First, a service provider that was granted access requests the data. Afterwards, a malicious service provider trying to access data without permission is shown.

5.2.1 Access Granted

The sequence diagram in Figure 3 shows the messages passed when a service provider, that has access rights on the data, tries to access a user's data.

The service provider requests the data via the data manager supplying the *txid* of the policy transaction and a signed version of the policy transaction. The

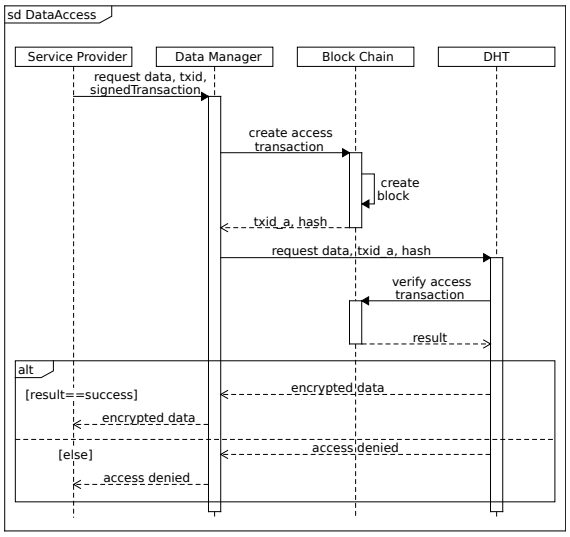

Figure 3: Sequence diagram of a service provider accessing a user's data. The service provider has access rights.

transaction is signed using the service provider's private key from the blockchain key pair. The data manager tries to create a transaction that logs the access to the data. This transaction requires the policy transaction from the previous section in order to be created on the blockchain. Each access costs two smallest units of the blockchain currency (2s) from the original policy transaction or a previous access transaction (cf. Section 4).

After the blockchain created the transaction, the *txid_a* of the logging transaction and the hash of the data is returned. The data manager requests the data from the DHT supplying the *txid_a* and the hash of the data. The DHT verifies that a current access transaction (*txid_a*) is persistent in the blockchain.

If a persistent transaction is found, the encrypted data is returned to the data manager. If the transaction was not found, for example because it was created in a forked chain, the access is denied.

The data manager finally either returns the encrypted data to the service provider or informs the service provider about the failed access.

Using the *encryptionKey*, that was shared by the user, the service provider can now decrypt the data.

5.2.2 Access Denied

When no access policy was granted to a service provider trying to access some data, the following procedure kicks in.

The service provider requests the data from the data manager. The data manager tries to create the necessary transaction on the blockchain. The blockchain will not allow the creation of the logging transaction, because the service provider is not part of the policy transaction.

This is the first access denial, which is returned to the service provider. However, even if the service provider manipulates the data manager and tries to access the data on the DHT directly, the DHT will stop the process, because of the missing transaction on the blockchain. The DHT then performs the logging action, that would normally be performed by the data manager and creates a transaction on the blockchain that informs the user about the tried access.

And, even if attackers gain access to the data on the DHT, they still need to obtain the shared encryption key from either service providers that got access rights or from the users themselves.

6 DISCUSSION

In this section, we list the benefits and limitations of our proposed system.

6.1 Benefits

Our proposed system provides a flexible privacy protecting data management system. The DPMS can be configured using different back-end systems, like a DHT or specialised data hosts.

It does not require trust in a single third party, when a DHT is used. Users do not put their data in the hands of a single entity that might use the data for anything it wants to. Instead users that are interested in using the system for their own data are collaborating in the operation of the system.

Users can manage their entered data and do not have to re-enter the same information on every service they want to use. This simplifies users' experience with Internet services.

Data leakage on a service provider side does not endanger the privacy of the user, as the DHT can react quickly and re-encrypt all linked data. Thus, data stored using our system is protected against data leakage and does not require users to react to leakage news.

Due to the use of proof-of-stake instead of proof-of-work for the mining process of the blockchain, less energy is consumed when maintaining the blockchain, compared to bitcoin's blockchain.

6.2 Limitations

When users revoke access to their data for one service provider, they must re-encrypt the stored data. This is necessary because the service provider, whose access

got withdrawn, still knows the encryption key of the data and thus might gain access to the data.

Another problem is the fact that service providers could be downloading and decrypting the data and storing it on internal servers instead of requesting the data from our system.

Without evaluation, performance issues cannot be excluded from the limitations of this system. On the one hand the time needed to create new blocks adds up, when waiting for transactions to be considered persistent. On the other hand, searching for transactions can also be considered time consuming, as searches directly on the blockchain are inefficient, due to linear search complexity.

7 RELATED WORK

In previous work Zyskind et al. proposed a blockchain-based system to protect personal data (Zyskind et al., 2015a). The system uses the immutable ledger of the blockchain for the storage of policies that define access rights. Their system has some limitations that we try to resolve with our proposed system. One limitation of the system is the shared identity that binds a user to several services, requiring a new identity when adding or removing a service from the group. Our system removes this limitation by using the standard public and private key pair identities from the blockchain domain. A user just adds new or removes services from the system by placing transactions on the blockchain. Services and users are identified by public key identities.

In the system, Zyskind et al. proposed, users must upload the same data multiple times to share it with different services, as an alternative to creating new compound identities. Our system allows users to share their data even after the first initial sharing, simply by creating a new transaction in the blockchain.

In our system, the blockchain and DHT structures are not connected directly. Instead the DHT implementation will verify the existence of a specific transaction on the chain. After an access transaction has been created in the blockchain, the DHT will grant access to the data.

Zhang et al. propose another privacy related system based on blockchain technology (Zhang et al., 2018). Their system is specialized for Internet of Things (IoT) devices and the data these devices collect. The approach combines the use of a blockchain with the use of trusted execution environments to preserve the privacy of the user, whose data was collected by the IoT device.

Another related work is called ”data-exchange wallet” by Norta et al., which, like the system presented in this paper, tries to provide a data management system for Internet users (Norta et al., 2018). The wallet, however, provides incentives to users to sell their data to companies. Norta et al. argue that this brings the profit, that service providers make selling their users’ data, back to the owner of the data. The data-exchange wallet has been implemented[5] and is currently in a beta phase. Our proposed system differs from this approach by not trying to sell users’ data but protecting it from misuse and further processing by the service providers.

Another block-chain-based system, which aims to protect users’ privacy is Enigma (Zyskind et al., 2015b). This system has a limited number of use cases due to the usage of calculations on encrypted data. It preserves privacy, in cases where these calculations are applicable, by using a decentralised calculation approach based on smart contracts. These contracts are executed by a blockchain.

Yli-Huumo et al. conducted an extensive systematic literature review on the research of blockchain technologies and revealed that over 80% of the reviewed papers were focusing on bitcoin and only less than 20% dealt with other use cases for the blockchain technology (Yli-Huumo et al., 2016).

Healthcare Data Gateways were proposed to provide privacy protected electronic health records utilising blockchain technology (Yue et al., 2016). However, in their paper Yue et al. do not specify how the blockchain is integrated into their system. Instead, they just use the term blockchain cloud whenever they talk about the secure and private storage of the health records.

Another system concerned with medical records was proposed by Xia et al., again using blockchain technology managing the access of researchers to medical data. They provide some estimated evaluation showing the growth of the blockchain depending on the number of transactions, concluding that their system is more scalable than the bitcoin blockchain system (Xia et al., 2017).

8 CONCLUSION AND FUTURE WORK

Overall our proposed DPMS seems to be a promising system that can make users’ lives easier. It reduces the effort needed, when using multiple services requiring the same data to be provided by the users.

[5]https://datawallet.com/

Additionally, the DPMS improves the users' privacy and protects them from data leakages, as it takes the data away from the service providers and places the data in a secured peer-to-peer system. However, a few limitations still must be addressed before our system can be deployed for public use.

Trusted computing modules and a verified client software might be used to stop service providers from locally storing the user data.

Time spent searching transactions could be reduced by using an external index, that allows to find blocks and transactions with reasonable efficiency.

The system could be extended to allow services to write data into users' data sets. Allowing the user to share data between different services, e.g. shared images or computed interests could be shared between multiple social media platforms.

In the future the suggested distributed hash table could be specified in more detail. Once the DHT is specified, an implementation of the system would be possible. The implemented system can then be used to evaluate the proposed approach of managing personal data.

Future work could also investigate other storage systems like the inter planetary file system (IPFS)[6] to replace the DHT.

Instead of sharing the encryption key directly via an encrypted connection a group key management method could be used. Examples for such methods are the certificateless public key cryptography (Al-Riyami and Paterson, 2003) or attribute-based group key management (Nabeel and Bertino, 2014).

REFERENCES

Al-Riyami, S. S. and Paterson, K. G. (2003). Certificateless public key cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 452–473. Springer.

Dworkin, M. J. (2015). SHA-3 standard: Permutation-based hash and extendable-output functions. Standard, Federal Information Processing Standards.

Kiayias, A., Russell, A., David, B., and Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology – CRYPTO 2017*, pages 357–388. Springer International Publishing.

Maymounkov, P. and Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. *Lecture Notes in Computer Science*, 2429:53–65.

Nabeel, M. and Bertino, E. (2014). Attribute based group key management. *Transactions on Data Privacy*, 7(3).

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *https://bitcoin.org/bitcoin.pdf*.

Norta, A., Hawthorne, D., and Engel, S. L. (2018). A privacy-protecting data-exchange wallet with ownership-and monetization capabilities. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *Acm Sigcomm Computer Communication Review*, 31(4):149–160.

Xia, Q., Sifah, E. B., Smahi, A., Amofa, S., and Zhang, X. S. (2017). Bbds: Blockchain-based data sharing for electronic medical records in cloud environments. *Information*, 8(2):44.

Yli-Huumo, J., Ko, D., Choi, S., Park, S., and Smolander, K. (2016). Where is current research on blockchain technology?-a systematic review. *PLoS One*, 11(10):e0163477.

Yue, X., Wang, H. J., Jin, D. W., Li, M. Q., and Jiang, W. (2016). Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. *Journal of Medical Systems*, 40(10):218.

Zhang, N., Li, J., Lou, W., and Hou, Y. T. (2018). Privacyguard: Enforcing private data usage with blockchain and attested execution. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 345–353. Springer International Publishing.

Zyskind, G., Nathan, O., and Pentland, A. (2015a). Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184.

Zyskind, G., Nathan, O., and Pentland, A. (2015b). Enigma: Decentralized computation platform with guaranteed privacy. *https://arxiv.org/pdf/1506.03471.pdf*.

[6]https://ipfs.io/