Differentially Private Data Sharing in Cloud Federation with Blockchain

Mu Yang
University of Greenwich, UK

Andrea Margheri
Runshan Hu
Vladimiro Sassone
University of Southampton, UK

Cloud federation is an emergent Cloud-computing paradigm that allows services from different Cloud systems to be aggregated in a single pool. To support secure data-sharing in a Cloud federation, anonymisation services that obfuscate sensitive datasets under differential privacy have been recently proposed. However, by outsourcing data protection to the Cloud, data owners lose control over their data, raising privacy concerns. This is even more compelling in multi-query scenarios where maintaining privacy amounts to controlling the allocation of so-called privacy budget. In this paper we propose a blockchain-based approach that enables data owners to control the anonymisation process, and enhances the security of the services. Our approach relies on blockchain to validate the usage of privacy budget and adaptively change its allocation via smart contracts, depending on the privacy requirements provided by data owners. Prototype implementation with the Hyperledger permissioned blockchain validates our approach with respect to privacy guarantee and practicality.

Cloud federation builds up interconnectivity and cooperation among already deployed clouds, enabling organisations to achieve various business goals, such as controlled sharing of data, services and optimisation of computing resources usage[1-3].

To support secure sharing of federated data, anonymisation services have been proposed as a building component of Federation-as-a-Service (FaaS),[3-4] a recent Cloud federation solution. This component implements differential privacy in order to obfuscate the result of statistical queries towards sensitive datasets,[5] enabling its privacy-preserving sharing. Offering this service in the context of a Cloud federation has benefits—access to multiple data sources and different service providers—but raises significant challenges for privacy management: sensitive datasets from multiple owners each of which has different privacy requirements, loose control on data and untrusted anonymisation services.

Traditional solutions for privacy management cannot be applied as-is in the context of Cloud federation. Firstly, typical management of privacy and data utility requirements must be extended to support multiple datasets and data owners.[6] Secondly, to protect anonymised data from degradation of privacy protection, recent approaches proposed to verify privacy budget which determines the amount of noise produced in the obfuscation process, and stop data sharing as soon as the budget is used up.[7] However, stopping sharing data must be avoided as much as possible to not hamper the business goals of the federation. Most importantly, due to the lack of trust among Cloud federation members, anonymisation services themselves cannot offer adequate guarantees for controlling and tracing privacy budget, e.g. they will end up being single point of attack to make multi-query de-anonymisation attacks possible.

Realising a trustworthy decentralised management of the privacy budgets is then of paramount importance to ensure privacy protection of sensitive datasets and, most of all, to enhance assurances on the anonymisation services. To this aim, we introduce here a new solution based on blockchain, an innovative technology that among other fascinating properties on data integrity ensures full decentralised control on data and code execution.

In the following, we first introduce a motivating example to better illustrate the limitations of existing approaches, then the research challenges addressed in the paper.

Motivating Example

Assume that a dataset containing employees' absence information is available (from multiple-owners) for the sharing process. A data requestor sends a statistical data request (e.g., a Mean query) and receives obfuscated query result by the anonymisation services deployed in the cloud federation. The controls on how datasets are integrated and accessed are here out of scope, our focus is on the privacy protection mechanism employed to anonymise the datasets.

The dataset, shown in Table 1, contains privacy-sensitive information, such as salary and number of absence, and the data owner wants to prevent the leakage of the sensitive information. To this aim, state-of-the-art anonymisation service based on Differential Privacy[5] is used. Intuitively, it relies on an ε parameter setting the privacy budget of a given dataset. Based on the ε, it generates randomised noise to the query result.

Table 1: Employee dataset

Employee ID	Date of Birth	Absence	Salary
1	08/11/1955	16	23112.30
2	22/07/1953	3	25388.43
3	01/01/1966	1	17303.11
…	…	…	…

Privacy requirements. Let us assume data queries about the mean of employees' salary. Data owners can set ε according to their privacy requirements, e.g. 0.1 or 0.5 for, respectively, stronger and weaker privacy protection that means different noise levels on obfuscated results. Note that the value setting of privacy budget ε can be presented to user using a Likert scale that ranges from "strong protection" (e.g., ε=0.1) to "weak protection" (e.g., ε=0.5) with "medium protection" (e.g., ε=0.25) as the neutral (default) option. Therefore, users select the desired protection strength rather than setting numeric values. The corresponding data loss due to the differential obfuscation can be visualised to users under each protection option in order to help users trade-off privacy protection and data utility. Notably, developing a rigorous method for choosing an optimal ε in practice, as well as the design of effective user interface for the privacy and data utility trade-offs are open research areas, and are out of the scope of the paper. Figure 1 graphically shows the noise of obfuscated results over 20 queries (results correspond to the critical points of the lines in the figure). With ε set to 0.5 results (dotted line) are closer to the actual ones (solid line), while with ε set to 0.1 results (dashed line) differ more offering stronger protection, but less utility.

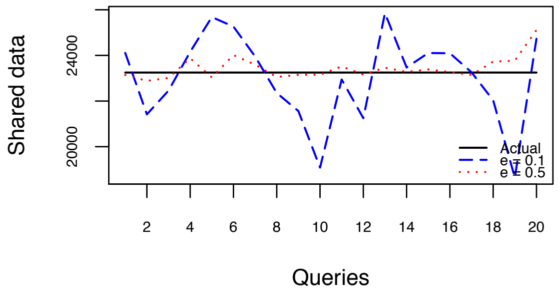


Figure 1: Obfuscated query results

Privacy degradation. Differential privacy suffers from privacy degradation as the number of queries increases. Sharing a single query result guarantees chosen privacy level (e.g., 0.1 or 0.5 in this example), making difficult to determine what the actual average salary is. However, if we combine multiple obfuscated results (i.e., each of the lines in the figure), the privacy level degrades by cumulating ε over queries.[5] For instance, executing 20 queries indicates 20ε-differential privacy. This can be visualised in Figure 1 as the sum distance between the points on the dotted line with the actual one tends to equalise, hence revealing the actual value. Furthermore, if more query types are allowed, that is, data requesters send not only Mean queries but also Max, Min, Quartile queries, then the data requesters can learn much more information by sending queries continuously. A few studies have been investigating adaptive budget allocation strategies,[8] which inspire us on the design of our mechanism to save budget consumption. Additionally, as budget may be consumed by multiple privacy services and refer to multiple data owners, the budget management cannot be entrusted to a single anonymisation service. Instead, it requires adequate integrity and accountability guarantees such that all involved parties can rely on it.

Proposed Approach

In this paper, we propose a blockchain-based approach for privacy-preserving data sharing in Cloud federation. It allows data owners to control the anonymisation process, such as defining their own privacy requirements, tracing the data-sharing activities, and enjoying secure services supported by the outsourced anonymisation services. The main contributions of this paper are the following:

- A blockchain-based data sharing approach to store, verify and adaptively allocate privacy budget consumptions via autonomous smart contracts according to data owner privacy and data utility requirements.
- A high-level system architecture enabling the integration of any data anonymisation service with any smart contract blockchain solution.
- Implementation and evaluation by means of the Hyperledger Fabric blockchain, and discussion on privacy and data utility enhancements.

PRELIMINARIES

Differential Privacy

Differential Privacy is proposed as a privacy technique for protecting individual records of statistical databases.[5] This is usually achieved by designing a mechanism that adds randomised noise to the query output, so that an adversary is not able to determine whether a targeted record is included in the database or not, no matter what side information the adversary might have.

To present our approach, we first present the key concept and implementation mechanism of differential privacy, more details are available in the differential privacy work[5].

Definition 1. (ε-Differential Privacy[5]). A randomised mechanism M with domain $\mathbb{N}^{|\mathcal{D}|}$ is ε-differentially private if for every set of outputs $S \subseteq \text{Range}(M)$ and all databases $D, D' \in \mathbb{N}^{|\mathcal{D}|}$ that differ in one record,

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in \mathcal{S}].$$

A popular technique that satisfies Definition 1 is the Laplace mechanism.[5]

Definition 2. (Implementing ε-Differential Privacy: The Laplace Mechanism). Given any query q, the Laplace mechanism is $q(D) + y$ where y is a random variable drawn from the Laplace distribution with scale parameter $b = S_q/\varepsilon$ where S_q represents the l_1-norm sensitivity[5] of the query q, and location parameter $\mu=0$.

$$f(y) = \frac{1}{2b} \exp\left(-\frac{|y|}{b}\right).$$

The variable y expresses how much noise should be added to the query outcome. The smaller the ε or the greater S_q, the greater noise generated for achieving ε-differential privacy. We use $\text{Lap}(\varepsilon)$ to denote the randomised noise generated by the Laplace mechanism.

An important property of differential privacy is the composition property, which shows how privacy degrades linearly when the number of queries on the same database increases.

Lemma 1. (Composition[5]). If M_1 is ε_1-differentially private, and M_2 is ε_2-differentially private, then let M be another mechanism that executes M_1 and M_2 independently on a database, M is $(\varepsilon_1 + \varepsilon_2)$-differentially private.

Blockchain and Smart Contracts

Blockchain is a novel technology that recently came to prominence when used as a public ledger for the Bitcoin cryptocurrency. It consists of consecutive chained blocks, replicated and stored by the nodes of a peer-to-peer network. Blocks are created in a decentralised fashion by means of a consensus algorithm, which can range from expensive proof-of-work mechanism (e.g., Bitcoin's) to lightweight Byzantine consensus algorithm (e.g., Hyperledger's). The use of consensus algorithms enables several data integrity related properties in blockchain, such as distributed control of the data on the chain, non-repudiation and persistency of transactions.

Differently from Bitcoin, new types of blockchains have recently appeared featuring smart contracts, that is, programs deployed and autonomously executed on the blockchain. Being part of blockchain makes contracts and their executions immutable and irreversible. The state-of-the-art smart contract blockchains are Ethereum and Hyperledger. Our implementation relies on the latter due to its performance and flexible architecture.

BLOCKCHAIN-BASED DATA SHARING

The objective of our blockchain-based data sharing approach is to allow data owners to control anonymisation processes, and to guarantee chosen privacy levels when using third-party anonymisation services especially to protect against multi-query attacks. Secure management of privacy budget is indeed the key to ensure privacy.

The Approach

Our approach utilises blockchain smart contracts to store, verify and adaptively allocate privacy budget consumptions depending on data owner's privacy and data utility requirements. It relies on the two phases below.

At the *Setup* phase, data owners provide their privacy and data utility requirements which are then stored in the smart contract. The privacy requirement is represented by the privacy budget ε_0, which represents the maximum amount of budget allowed on sharing data. Data utility requirement is represented by a numerical variable, denoted by $u \in R \geq 0$, representing the maximum amount of noise allowed on the actual query result, thus to maintain adequate data utility.

At the *Runtime* phase, data queries are managed returning, when allowed by the privacy budget and requirements, anonymised results. Indeed, our approach M consists of an unbounded sequence of mechanisms M_1, M_2, $\cdots$, where M_i operates when the i-th query is received. Figure 2 illustrates the activities involved in each mechanism M_i. Logically, it can be decomposed into three main test activities (i.e. the diamond boxes in the figure): *Query matching*, *Utility-based approximation* and *Budget verification*. The activity flow is reported in the figure and relies on data and computation offered by smart contract.

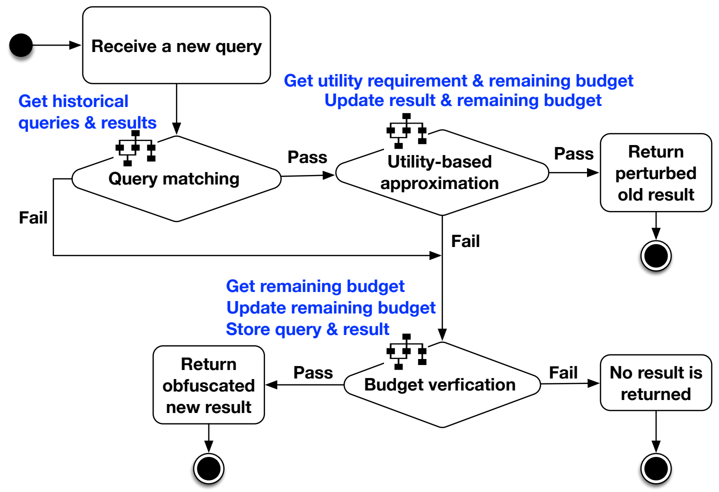

Receive a new query
Get historical queries & results
Get utility requirement & remaining budget
Update result & remaining budget
Query matching
Pass
Utility-based approximation
Pass
Return perturbed old result
Fail
Fail
Get remaining budget
Update remaining budget
Store query & result
Return obfuscated new result
Pass
Budget verfication
Fail
No result is returned

Figure 2: Overview of the runtime mechanism M_i (diamond boxes relies on smart contracts)

Query Matching

This activity aims at determining whether a newly received query has been executed before. Smart contract checks the sharing history stored on the blockchain. Formally, the sharing history amounts to the following tuple

$$(DsetId, \epsilon_r, [(qry_1, res_1), \cdots])$$

where DsetId is a reference to a dataset (or a column of it), while εr is the remaining associated privacy budget. The following list of tuples forms the sharing history. Each tuple (qry_1, res_1) has as first element the function type of the query—e.g. sum, average, max, min—and as second the corresponding previously released result. Thus, res_i are just the latest released results for each query type *i*, hence lightweight information whose limited size makes them suitable for blockchain storage.

The query matching compares a newly received query qry on the dataset referred by DsetId with the corresponding history. The query is denoted by the tuple (DsetId, qry, ε), where the parameter ε denotes the requested budget for executing the query. The value of ε can be provided by data consumer, or pre-defined as a fixed value by anonymisation services. Without loss of generality, we assume function types fixed and comparable by names; additional comparison parameters can be set as well. Namely, given a DsetId, the test is passed when qry is equal to a qry_i part of the history. Notably, to keep queries private to all the members part of the blockchain, the history data can be stored hashed. The comparison will be then on hash texts.

Utility-based Approximation

This test aims at checking whether a previous released result can approximate the result to return for the current query. The test pseudocode is reported in Algorithm 1. Firstly, it checks whether the remaining budget is enough for executing the test (Line 1), If yes, it produces an obfuscated version of the old result resold using a very small amount σ of the privacy budget (Line 2). Otherwise, it returns false (Line 11) stating the approximation test failed. The computed obfuscated result is compared by a smart contract with the actual one with respect to the threshold u (Line 3). If the approximation test passes, the new obfuscated result is set as the last returned result of such query (Line 4), the budget is updated accordingly (Line 5) and the approximated result is returned (Line 6). Otherwise, only the budget is updated (Line 7, 8) to keep tracking that σ was consumed by the approximation test.

When this approximation test succeeds and res_{old} is used, the consumed budget σ is significantly less than that (i.e., requested budget ε) used for returning the actual res. The obfuscation added to res_{old} aims at adding randomness to the utility test. This permits dealing with the fact that adversaries may know how the test works and attempt to gain knowledge about the actual result res from the test result.

Algorithm 1 Utility-based approximation Pseudocode

```
Input: res, $res_{old}$: actual and previous query results
u: data utility requirement; $\varepsilon_r$: remaining budget
$\sigma$: a small budget for performing approximation test.
Output: res’ if passes; boolean value false otherwise.
1: if $\varepsilon_r \geq \sigma$ then     ◂Checking budget for the test.
2:    $res' = res_{old} + Lap(\sigma)$;    ◂Obfuscating old result.
3:    if $|res' - res| \leq u$ then
4:        $res_{old} = res'$    ◂Updating history in blockchain.
5:        $\varepsilon_r = \varepsilon_r - \sigma$    ◂Updating budget in blockchain.
6:        return res'.
7:     else
8:        $\varepsilon_r = \varepsilon_r - \sigma$    ◂Updating budget in blockchain.
9:        return false.
10: else
11:    return false.
```

Budget Verification

The budget verification test is triggered if there has been no same query executed (i.e., the query matching test failed), or the query result cannot be approximated (i.e., the approximation test failed). Thus, a new result has to be computed, as long as the remaining privacy budget is enough.

This test is carried out on a smart contract that, given a query tuple (DsetId, qry, ε), compares the remaining budget εr of the dataset DsetId with the requested budget ε. If the test succeeds, the anonymisation service generates randomised noise under differential privacy to add to the actual query result consuming the requested budget. Otherwise, the query is rejected because it would violate the defined privacy requirement.

This test ensures the satisfaction of pre-defined privacy requirement ε_0 as it makes sure the consumed budget does not exceed ε_0. The mechanism updates the remaining budget, query function type (if it was not been stored before) and the generated new result in the blockchain.

To summarise our approach, the privacy budget allocation strategy depends on the three tests proposed in the data sharing mechanism: query matching, approximation and budget verification. More specifically, these tests aim to save privacy budget consumption by checking whether previously released results can approximate the new query result. If passes, then an approximation is returned avoiding the calculation of the new perturbed result which consumes more privacy budget.

Adversary Model

The ultimate goal of adversaries is to degrade privacy, i.e., to re-identify data subjects in a targeted dataset. To this aim, we assume that two types of adversarial activities can be carried out.

Firstly, we assume that adversaries can access to all perturbed query results, and are able to launch different types of privacy attacks based on their observations of those query results, in order to re-identify data subjects. Being our approach satisfying the ε_0-differential privacy, the privacy is guaranteed.

Secondly, we assume that adversaries are interested in tampering privacy budget, such as altering, deleting privacy budget and making it unavailable so that perturbation will not be properly applied on protecting dataset. Being the privacy budget managed exclusively on blockchain, it is guaranteed that the attacker is not able to compromise such decentralised structure to tamper with the anonymisation process.

System Architecture

To implement the proposed approach, we propose a generic system architecture for blockchain-based data sharing. Specifically, an Anonymisation Interface (AI) is realised to integrate pluggable differential privacy component with blockchain smart contracts.

As illustrated in Figure 3, federated datasets and anonymisation services (denoted by ANM) are integrated via AIs, which act as mediator with blockchain smart contracts realising the control flow in Figure 2 previously described. Data consumers interact with any AI to query datasets. Then blockchain smart contracts execute the test activities to ensure privacy protection.

Data owners federating their sensitive datasets to a Cloud federation can then trust third-party anonymisation services due to the principled exploitation of blockchain smart contracts. They store and evaluate sharing history, while enforcing utility and data privacy requirements. Non-repudiable evidences of privacy budget consumption and released query results enhance the security guarantees on privacy-preserving data sharing processes. In particular, blockchain smart contracts carry out the secure management of privacy budget and carry out the test activities. The third-party anonymisation services only execute when there is no previously released result that can be used. This prevents attacks of altering, deleting budget consumptions, and improves the availability of anonymisation services.

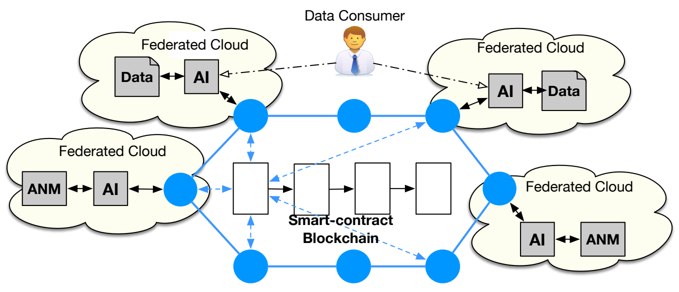


Figure 3: Blockchain data sharing System in a Federated Cloud (AI stands for Anonymisation interface, while ANM stands for Anonymisation service)

EXPERIMENTAL EVALUATION

We prototyped our blockchain-based data sharing approach by the Hyperledger Fabric smart contract blockchain and a traditional implementation of differential privacy using Laplace mechanism. A real-world dataset from the Italian Ministry of Economy and Finance is used, which contains employees' salary information as exemplified in Table 1. Our implementation is in Hyperledger Fabric V0.6 on a 2.6 GHz 4 core Intel Xeon laptop running Ubuntu 14.04.5.

The experiments aim at evaluating, on the one hand, privacy and data-utility guarantee and, on the other hand, blockchain practicality. Specifically, the query function types are four—i.e. sum, average, max and min—the privacy requirement ε_0 is set to 10, and the data-utility requirement u to 1500. The requested budget ε for each query is fixed at 0.5. Queries are simulated continuously and randomly by uniformly choosing a query type from those four. The compared baseline approach is the standard differential privacy mechanism that generates randomised noise independently for each query.

Privacy

As proved in the description of our approach, our mechanism always provides ε_0-differential privacy. That is, the consumed privacy budget does not exceed ε_0 that the pre-defined privacy requirement is satisfied. In this experiment, we focus on how the budget is consumed over queries. Figure 4 shows the budget consumption when our and the baseline approach are implemented. It is clear that the remaining budget decreases linearly in the baseline approach, so that the budget is used up after 20 queries. The remaining budget in our approach decreases slower than in the baseline approach. Specifically, for the first query, it drops the same in both approaches as there has been no sharing history and no result can be approximated. From the second query, the decrease slows down as historical sharing results become available for approximation at some queries. More specifically, after receiving 6 queries, the historical sharing tuple stored in the blockchain becomes

$$(DsetId = EmployeeDset, \epsilon_r = 7.93,$$
$$[(qry_1 = \max, res_1 = 28643.57),$$
$$(qry_2 = average, res_2 = 23147.29),$$
$$(qry_3 = \min, res_3 = 16127.25),$$
$$(qry_4 = sum, res_4 = 578106.25)])$$

where all four query types have been received and stored for future approximation.

We now change the number of query types from four to two (i.e., max and average) representing the situation where two query types are allowed. The consumption of privacy budget is plotted together with the situation of four query types. Indeed, the fewer query types, the less the budget is consumed: it is more likely that historical sharing results can be used to approximate new results. Therefore, our approach is able to allow more queries executed and is more effective when there are fewer query types.

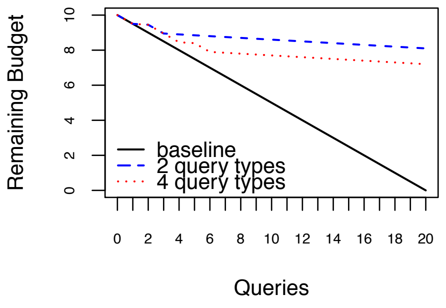


Figure 4: Budget consumption as the number of queries increases, where "2 query types" means that just max and average queries are allowed, while "4 query types" also includes min and sum queries.

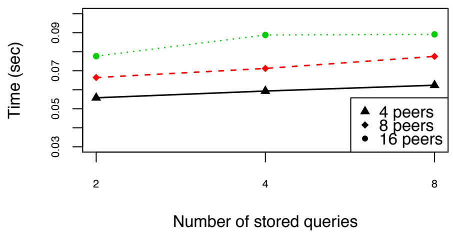
4.2 Data Utility

Our approach introduces less noise compared with the baseline approach after receiving more queries, as the approximation test takes into account the utility requirement, guaranteeing the amount of generated noise is bounded by u = 1500. We compute the mean of the absolute noise over 20 queries, and have 43331.823 for the baseline approach, 3864.97 and 3806.47 for our approach with respectively four and two query types. Therefore, our approach provides slightly better data utility, and the number of query types does not affect the data utility.

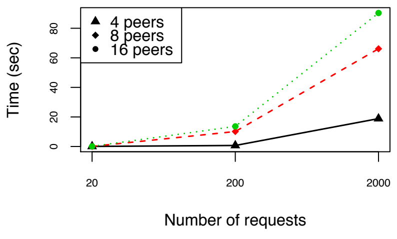
Blockchain Practicality

Storage capacity. As the data sharing history stored on each block only the latest released result, rather than a full list of release results, the size of the tuple that gathers such sharing history is suitably small and can be optimised grouping by query types. Tuples can be illustrated, e.g., as (DsetId, ε_r, [(qry_1, res_1), (qry_2, res_2)]) if two query types are allowed. Therefore, the design of history tuple is light and suitable for blockchain storage as testified by the extensive tests.

Blockchain performance. The performance of smart contract computation is shown in Figure 5a when the number of stored query types changes from 2 to 4 and 8. The computation on Hyperledger Fabric is very efficient as the maximum time is 0.09 second. There is a slightly increase in time when the size of the sharing history tuple increases. As the number of peers deployed in the Hyperledger blockchain increases, the computation time increases. This is because it takes more time to allow all peers to confirm the computation result (i.e., adding a new block). We now simulate more query requests at a single timestamp, particularly from 20 to 200 and 2000 requests. As shown in Figure 5b, the time increases and the maximum time becomes 90 seconds when there are 2000 requests received at the same time. Therefore, implementing our approach in Hyperledger Fabric offers good performance, and is able to handle great number of requests.



(a) Number of allowed query types



(b) Workload

Figure 5: Smart contracts performance.

Permissioned blockchain. Our prototype implementation relies on Hyperledger Fabric, which enables us to deploy a private blockchain with control on operating users. Data owners are the default users who are allowed to access the blockchain but only manage the anonymissation process of their own datasets. Additional access rules can be negotiated with data owners supported by the function of smart contract.

RELATED WORK

A considerable body of research has been devoted to address the data privacy issues in cloud computing. Because of the openness and multi-tenant characteristic of the cloud, traditional privacy-preserving approaches (such as anonymisation techniques[9]) by their own cannot ensure the protection of personal data. Cryptographic approaches have been proposed to encrypt data before uploading to the cloud,[10,11] and data can only be decrypted by authorised data consumers. These approaches rely on novel access control models to support various access request from federated clouds.[12]

In order to equip data owners with more control and accountability over data protection, blockchain-based proposals utilise blockchain to store data and control data sharing as a data management platform.[13,14] More specifically, Enigma,[13] a peer-to-peer network supports different parties to jointly store and run computations on data while guaranteeing the privacy of data. This proposal combines blockchain with multi-party computation techniques and examines a mobile application data sharing scenario. The other proposals, such as the work by Ekblaw et al.,[14] aim to protect patient health records, and ensures the immutable, quick access, confidential properties of such data storage and access. While these approaches focus on storing sensitive data directly on blockchain, our solution stores the process of anonymisation services which provides stronger data privacy guarantee and requires only light configuration for implementing our solution in cloud federation.

CONCLUSION

Our blockchain-based data sharing approach allows data owners to control the privacy protection of their datasets while enjoying the anonymisation services provided in a cloud federation. Future work includes examining practical deployment issues in a cloud federation, integrating with security components (e.g., access control) and developing an effective user interface to support the control of the anonymisation services.

REFERENCES

1. A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in CLOUD. IEEE, 2010, pp. 337-345.
2. T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud Federation," in Cloud Computing, GRIDs, and Virtualization, 2011, pp. 32-38.
3. F. P. Schiavo, V. Sassone, L. Nicoletti, and A. Margheri (Eds.), "FaaS: Federation-as-a-service," CoRR, vol. abs/1612.03937, 2016.
4. A. Margheri, M. S. Ferdous, M. Yang, and V. Sassone, "A distributed infrastructure for democratic cloud federations," in Cloud Computing, 2017 IEEE 10th International Conference on, 2017, pp. 688-691.
5. C. Dwork, "Differential privacy," in Proceedings of the 33rd International Conference on Automata, Languages and Programming, 2006, pp. 1-12.
6. B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," ACM Comput. Surv., vol. 42, no. 4, pp. 1-53, 2010.

7. F. D. McSherry, “Privacy integrated queries: An extensible platform for privacy-preserving data analysis,” in Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, 2009, pp. 19-30.
8. G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias, “Differentially private event sequences over infinite streams,” Proc. VLDB Endow., vol. 7, no. 12, pp. 1155-1166, 2014.
9. M. Yang, V. Sassone, and K. O’Hara, “Anonymisation: managing data protection risk code of practice,” UK Information Commissioner’s Office, 2012.
10. C. Esposito, A. Castiglione, and K. K. R. Choo, “Encryption-based solution for data sovereignty in federated clouds,” IEEE Cloud Computing, vol. 3, no. 1, pp. 12-17, 2016.
11. C.Wang, Q.Wang,K.Ren,andW.Lou,“Privacy-preservingpublic auditing for data storage security in cloud computing,” in Infocom, 2010 proceedings ieee, 2010, pp. 1-9.
12. D. Chen and H. Zhao, “Data security and privacy protection issues in cloud computing,” in Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering Volume 01, 2012, pp. 647–651.
13. G. Zyskind, O. Nathan, and A. S. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in Proceedings of the IEEE Security and Privacy Workshops, 2015, pp. 180-184.
14. A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, “A case study for blockchain in healthcare: “medrec” prototype for electronic health records and medical research data,” in Proceedings of IEEE Open & Big Data Conference, vol. 13, 2016, pp. 13.

ABOUT THE AUTHORS

Mu Yang is a Lecturer in the Department of System Management and Strategy at the University of Greenwich, UK. Her research interests include cloud computing, data privacy and security, blockchain technology. She has a PhD in Computer Science from the University of Southampton, UK. Contact her at m.yang@greenwich.ac.uk.

Andrea Margheri is a Senior Research Fellow in the Department of Electronics and Computer Science at the University of Southampton, UK. His research interests are in the area of cyber security of modern computing systems. He has a PhD in Computer Science from the University of Pisa, Italy. Contact him at a.margheri@soton.ac.uk.

Runshan Hu is a PhD student in the Department of Electronics and Computer Science at the University of Southampton, UK. His research interests include data privacy, anonymisation, and machine learning. Contact him at rs.hu@soton.ac.uk.

Vladimiro Sassone has worked at the University of Southampton since 2006, where he is a Professor in Cyber Security, the Roke/Royal Academy of Engineering Research Chair in Cyber Security, the Head of the Cyber Security Group, the Director of the GCHQ/EPSRC Academic Centre of Excellence for Cyber Security Research (ACE-CSR), the Director of the Cyber Security Academy (CSA), His recent research include resource access control over untrusted networks, trust management systems, predictive trust-and-reputation models, anonymity and privacy in the presence of trust and attackers’ belief systems. Contact him at vsassone@soton.ac.uk.