Computers and Electrical Engineering 109 (2023) 108747

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

GDPR compliance verification through a user-centric blockchain approach in multi-cloud environment☆

Haris Ahmad, Gagangeet Singh Aujla *

Department of Computer Science, Durham University, Durham, UK

ARTICLE INFO

Keywords:
Compliance verification
Blockchain
GDPR
Privacy protection
Security

ABSTRACT

With cloud-hosted web applications becoming ubiquitous, the security risks presented for user personal data that is migrated to the cloud are at an all-time high. When using a cloud-hosted web application, users only ever interact with web interfaces of the web applications and are usually completely unaware of how their data is distributed amongst the multiple cloud service providers that the web application uses, making it difficult to verify the lawful use and ownership of personal data. The General Data Protection Regulation (GDPR) seeks to empower users to gain better control over their personal data. Blockchain-based approaches have risen in popularity over the recent years to tackle the challenge of verifying GDPR compliance in multi-cloud environments. By deploying smart contracts on the blockchain, we can create transparent and immutable logs of data processes in the hopes of automating GDPR compliance verification. However, the existing works are still limited to provide a user-centric compliance verification. To this end, we propose a user-centric, blockchain-based framework for data management in a cloud environment where all GDPR-relevant data operations take place on the blockchain through well-defined smart contracts.

1. Introduction

Cloud computing is the all-encompassing term used to refer to any form of computing service that is delivered over the Internet (or "the cloud"). Cloud-hosted applications are generally built upon an ecosystem of interconnected cloud service providers, i.e. user applications such as e-commerce websites will be connected to other software providers such as payment gateways or shipping services. The user data collected to personalise user experience on cloud-hosted applications (e.g., online purchases) must also be migrated on to the cloud [1]. With user data being collected by and shared amongst multiple cloud providers, it becomes a challenge for the application provider to verify the security and lawful use of the user's personal data. Further, the user only ever interacts with the web interface of the application service provider and, after trusting the application provider with their personal data, it is difficult for the user to be certain who has had access to their data or the operations that have been carried out on it. To combat this, the General Data Protection Regulation (GDPR) was published in 2016 (and enforced in 2018 to give companies time to revamp their architecture for GDPR compliance) to set a legal standard on how a user's personal information can be handled by businesses and organisations. The primary goals of the GDPR are to enhance the control and rights of European individuals on their personal data and also to simplify data protection regulation by providing organisations a single set of rules with which they may streamline compliance. Some of the key GDPR terms are provided below.[1]

☆ This paper was submitted for special section VSI-aips, but should be included in regular issues of CAEE. Reviews were processed by Guest Editor Prof. Ali Kashif Bashir and recommended for publication.

* Corresponding author.
E-mail address: gagangeet.s.aujla@durham.ac.uk (G.S. Aujla).

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

https://doi.org/10.1016/j.compeleceng.2023.108747
Received 5 September 2022; Received in revised form 27 April 2023; Accepted 1 May 2023
Available online 1 June 2023

- **Personal Data** is any piece of information related to an identified or identifiable (e.g., can be referenced using some identifier such as name, IP address, location, etc.) natural person. This is a very inclusive definition as it means "personal data" can consist of any "objective" data such as name and address, "subjective" data such as performance evaluations and political views, metadata and even inaccurate data as long as it relates in some way to an identifiable individual. This means service providers must be aware of any type of data that they collect/process as any of it can be under GDPR jurisdiction.
- **Data Subject** is any living European individual about whom a company or organisation collects, holds or processes personal data and can be identified by reference, directly or indirectly, to that personal data. The data subject is the individual who's personal data is regulated by GDPR and is entitled to the GDPR's eight privacy rights. In the cloud application scenario, this will be the user/customer using the web application.
- **Data Controller** is any natural/legal person or company that dictates the purposes and means for the collection, storage and processing of the data subject's personal data. The data controller is ultimately responsible for implementing data protection policies and ensuring compliance with GDPR. In the cloud scenario, this will be the web application whose interface the data subject interacts with to provide their personal data.
- **Data Processor** refers to any natural/legal person or company that processes data on behalf and under the authority of the data controller thus have a more limited compliance responsibility. However, if the processor decides to act in a way that is against the instructions of the controller, i.e. the processor defines its own purpose and means of data processing, then it becomes a data controller and is equally liable for GDPR compliance. In the case of a cloud application, the data processor can be the payment gateway software or the shipping services a web application such as an e-commerce website might utilise.

Despite the many benefits that an on-demand computing service provides, cloud computing also comes with great risks when considering the privacy and security of personal data [2] — including data leaks, unauthorised access, and data interception during transfers. One difficulty during the collection and storage phase is dealing with the data subject's consent and opt-out rights. Despite the data subject having consented to certain operations, they reserve the right to revoke their consent. This leads to issues when, in a multi-tenanted cloud environment, data processing is being outsourced to multiple cloud service providers. This means that careful tracking of each piece of collected personal data at all distributed locations is vital. Tracking personal data gives the data controller the capacity to not only verify that processing has ceased but also to verify data erasure, to fulfil the data subject's "right to be informed" and "right to rectification". Tracking personal data on the cloud, however, is easier said than done. Cloud services are generally not tailored to the client's specific needs and so lack the detailed, data-centric logging functionality necessary to gain insight on how personal data is processed. Further, current monitoring tools for clouds are only designed from a system-centric perspective to measure the performance and costs of operations.

To tackle the above challenges, approaches using blockchain technology and smart contracts have seen a rise in popularity. A "blockchain" is a distributed database with an append-only data structure that consists of a sequence of "blocks", with each block containing a list of transactions, a timestamp of when the block was created and a cryptographic hash of the previous block. Since each block contains information about the previous block, they form a "chain" with each block reinforcing the one before it. Each block is shared amongst all members (or nodes) of the blockchain in a peer-to-peer network, thereby providing a common consensus on the contents of the blockchain. Adding valid blocks to the blockchain is crucial since a longer chain results in the network being more certain of the correctness of its current state because each valid block added to the blockchain reinforces the validity of the entire chain. The Proof-Of-Work (PoW) is essentially a way for nodes on the network to prove that they have expended computational power (e.g., work) to achieve a consensus in a decentralised way and to prevent malicious nodes from overtaking the network [3]. Therefore, by its nature, the blockchain is resistant to data modification (or immutable) as, once recorded, changes in any block would invalidate hashes of all previous blocks and break the consensus of the nodes on the network. Furthermore, the fact that all transactions are validated to be true by a consensus mechanism and the cryptographic design of the blockchain makes it secure and difficult to tamper with. Finally, due to its distributed nature, i.e. every node possesses a copy of the blockchain, the blockchain network is completely transparent [3]. These qualities make blockchain technology very appealing for the GDPR compliance problem as there would be no way to manipulate the logs created when tracking personal data crossing between multiple cloud boundaries, with the recorded logs being transparent to all. Smart contracts [4] are computer programs deployed onto the blockchain network to digitally facilitate, verify and enforce contracts between two or more parties in an automated manner without any time loss or the need for an intermediary party. Smart contract programs usually consist of triggering statements such as "if-then" statements that allow parties to set agreed upon conditions that need to be met before certain actions (or "transactions") can be carried out. When smart contracts are deployed onto the blockchain network, they are recorded and verified by the blockchain and become tamper-resistant. After deployment, smart contracts stay running on the blockchain and can be queried to retrieve the state of the blockchain or to record new transaction data onto the blockchain. Using smart contracts on a blockchain network, we have the potential for creating automated, transparent and tamper-resistant logs of data processes in a multi-cloud environment.

1.1. Research problem and contributions

The GDPR and the Data Protection Act 2018,[2] the UK's implementation of the European GDPR, seek to enable non-expert users to make knowledgeable decisions on how their data is handled, i.e. providing "informed consent" for the data operations carried out on

[2] https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted

their personal data. GDPR compliance provides greater trust for users in their service providers and protects service providers from the costly fines (the higher of 4% of annual global turnover or 20 million Euros) and loss of reputation that can result from GDPR infringements. A famous example of GDPR infringement is the misuse of data of 87 million Facebook users in the 2018 Cambridge Analytica scandal resulting in Facebook being fined $5 billion.[3] Having stressed the importance of GDPR compliance, realising these privacy regulations in a cloud environment remains a challenge. An approach that has seen a rise in popularity is the utilisation of blockchain technology and smart contracts due to the potential of their features (such as transparency, security, and immutability) in tackling GDPR. Thus, we have considered the following research question (RQ) in this paper.

RQ: *How to achieve GDPR compliance through a user-centric, blockchain-based personal data management system in a multi-cloud environment?*

To address the above RQ, we believe that GDPR compliance is achieved when the data subject has full control and visibility over his/her personal data and all GDPR related operations take place through well-defined methods on smart contracts deployed on the blockchain for complete transparency. hence, we design and implement a blockchain-based framework, using a cloud-based e-commerce application as a case study. The major contributions of this article are listed below.

1. A cloud-based e-commerce application scenario is designed to demonstrate the importance of GDPR and motivate our own solution.
2. An analysis of the GDPR requirements is performed for the deployed e-commerce scenario.
3. Encode GDPR-relevant data operations onto Ethereum smart contracts deployed and use them on an Ethereum network to automate the monitoring and logging of GDPR-relevant data operations and detect compliance violations.
4. A web application is designed to enable the actors in our GDPR framework to interact with our smart contracts in order to carry out data operations and receive GDPR violation reports.
5. Thoroughly test the efficacy of our proposed solution using the Ganache and Ropsten testnets.

1.2. Organisation

The paper is organised as follows. The related work is outlined in Section 2. While proposed approach presents the proposed method, the results are evaluated in Section 3. Before drawing a conclusion in Section 5, we discuss the results in Section 4.

2. Related work

In this section, we provide a comprehensive overview of the research that attempted to ensure compliance with GDPR legislation without impeding the performance of the hosted service. In [5], the authors develop an integrated Knowledge Graph that represents GDPR data regulations which can be used by Big Data handlers to automate GDPR compliance. However the proposed system does not automatically flag compliance violations and its effectiveness in a cloud environment is not evaluated. In [6], business processes of a cloud-based purchase/order system are converted into timed automatons implemented in UPAAL where states represent GDPR relevant data operations such as access, storage, transfer, and profiling. GDPR verification is carried out by checking whether certain factors are met for each data operation (i.e. if the user has provided consent, the data is encrypted, etc.). The paper does not however integrate the model and evaluate its performance in a cloud-based service. The aforementioned approaches do not leverage the immutability and transparency of the Blockchain, two features that are key in achieving GDPR compliance as the entire essence of the data protection act is to provide the user with a clear and tamper-proof view of the operations carried out on their user data.

In [7], the authors provide an overview of Blockchain technology, the requirements of the GDPR regarding the processing of personal data and present two real-world GDPR problems and discuss whether they could be solved using the Blockchain. The paper does not provide an automated solution for verifying GDPR compliance however the authors also express concerns that processing personal data in a Blockchain may violate GDPR. A conceptual model called ProvChain was presented in [8] to gather cloud data provenance and logging this onto a Blockchain network to provide data operation assurance. Authors of [9] present a data provenance framework that uses a layered architecture of Ethereum smart contracts to fulfil functional and non-functional requirements of heterogeneous IoT. This is extended in [10] by considering the framework in additional use case scenarios and with more rigorous evaluation of transaction throughput and latency of the solution. Authors of [11] present LineageChain, a novel provenance system implemented on Hyperledger and ForkBase that enables querying of historical states. One of the earliest works on personal data management is [12], which proposes a protocol that uses a Blockchain network as an automated access-control manager for a decentralised personal data management system. In [13], the authors propose a Blockchain-based data usage audit architecture for availability and accountability such that user privacy is protected. The approach involves deploying auditable smart contracts on Blockchain networks for transparent data access and processing/sharing of data by data owners. Further, authors of [14] propose a Blockchain approach for storing cloud attestation. Smart contracts are used to record the migration of data in cloud environments and can be queried by the user to find the location of their data. This is extended in [15] to give users more control over the migration of their data by enabling them to whitelist cloud providers. The approaches detailed above leverage blockchain and smart contracts in an effort to improve the cloud user's privacy and trust. However, none of the papers above use the GDPR guidelines to provide standard regulations for cloud providers to follow. This results in the data privacy and transparency achieved

[3] https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook

Table 1
Feature-based comparison of proposed approach with the existing proposals considering GDPR.

Paper	Web application	Consent/ compliance verification	User centric	Use case	Monitor	Encryption	Violation	Obligations	Data minimisation
[1]	×	×	✓	×	✓	×	×	×	×
[2]	×	×	✓	×	✓	×	×	×	×
[16]	×	Manual	×	×	×	×	×	×	×
[17]	×	×	×	×	×	✓	×	×	×
[18]	×	×	×	×	×	×	×	×	×
[19]	×	×	×	✓	✓	×	✓	×	×
[21]	×	Automated	✓	pharmacy	✓	×	Automated	×	×
[22]	×	Automated	✓	pharmacy	×	×	Automated	×	×
[23]	×	Automated	✓	healthcare	✓	×	Automated	✓	✓
Current	✓	Automated	✓	e-commerce	✓	AES	Automated	✓	✓

by the above methods being difficult to evaluate as well as not being able to match the standards currently expected with the introduction of the GDPR.

To overcome the above challenges, an approach, meeting GDPR requirements, that uses the blockchain for data accountability and provenance tracking is proposed in [16]. Two models of a Blockchain-based approach for data accountability and provenance tracking are implemented. One model involves users creating tailored smart contracts for each controller (i.e. cloud provider) and this contract keeps track of the shared data, how the data is used and the policies the user has set in regards to the usage of that data. The other model involves controllers creating a generic smart contract specifying how data from all users will be used and users join this smart contract by accepting the data usage policies. The paper finds that user-created contracts are more effective for sensitive data with less frequent exchanges whereas the controller-created contracts are more appropriate for dynamic data with more frequent exchanges. A limitation of the paper is that verification for the compliance of the consent rules must be done manually for both models.

A secure and transparent personal healthcare data sharing system that complies with GDPR is proposed in [17]. The method uses blockchain supplemented by cloud storage to share healthcare data and machine learning approaches to monitor the quality of the personal data. A limitation of the approach is that it does not consider any methods to verify the data stored on the blockchain network. The availability and immutability of Blockchain records is leveraged in [18] to create tamper-resistant audit trails for forensics of IoT Systems using cloud resources for checking GDPR compliance. However, the paper does not translate GDPR obligations into smart contracts nor does it consider an automated method to verify the GDPR compliance of cloud providers. An approach for creating audit trails for IoT devices under GDPR is discussed in [19]. GDPR obligations are translated into smart contracts for automated privacy-protection of IoT data. The paper does not, however, consider compliance verification of the IoT devices like in [20]. Formal representations for the verification of GDPR obligations such as user consent, data protection, data minimisation, and data transfer are presented in [21]. The paper then encodes these obligations onto smart contracts to carry out GDPR verification in IoT environments. One limitation of the paper is that it does not consider user preferences for verifying GDPR obligations, another limitation is that it does not utilise containers or virtual machine-based environments. Similarly, GDPR obligations were encoded into smart contracts in [22] to automate GDPR compliance verification of cloud providers however the paper does not give much consideration to the data protection and data minimisation obligations. A cloud-based architecture for verifying GDPR compliance during access, transfer and profiling operations is presented in [2]. Voters on a blockchain network are responsible for flagging GDPR violations via voting smart contracts, with the users being enabled to trade-off GDPR compliance for performance by varying the degree of compliance required of the provider. The paper also illustrates how quickly the costs of using a blockchain can rise as the number of data operations (carried out by service providers) and the number of voters increase.

A Blockchain-based approach that uses smart contracts and a container framework is proposed in [23]. The authors introduce a GDPR-agent to capture data operation events carried out by cloud service providers (inside containers) and these are passed through a filtration mechanism to filter out operations relevant to GDPR obligations (data protection, minimisation, transfer, and storage). The filtered operations are checked by smart contracts deployed on an Ethereum Blockchain to verify GDPR compliance and violations are sent to a GDPR manager, a Restlet server that handles internal communication and communication with the user. The paper also allows the user to prioritise certain GDPR obligations to reduce overhead.

In conclusion, a limited research has been carried out recently into using blockchain to automate GDPR compliance verification. The novelty of our approach is that we provide a lot more functionality to the data subject in our GDPR compliance framework to improve the trust of the data subject in the data controller. Also, all data storage and data operations take place on the blockchain for complete transparency. The comparison of the proposed approach and some of the above discussed existing proposals based on various features is provided in Table 1.

3. The proposed approach

Here, we describe the blockchain-based GDPR compliant platform that data subjects can use to share their personal data with controllers and gain complete transparency over its processing. To automatically enforce GDPR regulations, the proposed platform must have the capability to:

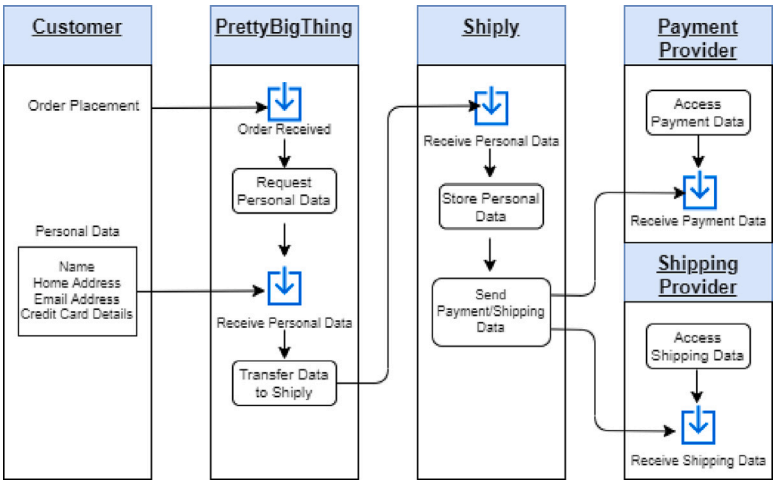

Fig. 1. Distribution of personal data in a typical cloud-based e-commerce scenario.

1. allow the data subject to grant/revoke consent for the data operations permissible on their personal data.
2. ensure that both the data subject and data controller are aware of the location of user data.
3. monitor the operations executed on user data.
4. transparently record the GDPR relevant data operations (i.e. access, storage, transfer, profiling, etc.) carried out on personal data. This enables the data subjects, data controllers or any third party to verify that only authorised operations are being carried out.
5. flag GDPR violations and alert the data subject.

3.1. E-commerce scenario — "PrettyBigThing"

We consider a cloud-based e-commerce scenario to demonstrate the need for GDPR compliance verification in a multi-cloud environment. Consider a customer that visits an online fashion retailer, PrettyBigThing, to purchase clothes, make a payment and have them delivered to their home address. During the entire order placement process, the online retailer, PrettyBigThing, will require access to several items of personal data from the customer — including name, home address, email address, card details. PrettyBigThing makes use of a Software-as-a-Service (SaaS) cloud provider (Shopify) to create and host their e-commerce store. When customers use PrettyBigThing to place orders, the information collected by the e-commerce store is stored on the databases of Shopify. This can be to accelerate future purposes, i.e., by saving the address of the customer, or to feed to into third-party Shopify plugins that PrettyBigThing makes use of to sell products. PrettyBigThing may also make use of some third party providers to facilitate payment and shipping. The service provider for payment would need PrettyBigThing to provide access to the customer's name and bank account details. The third party for shipping and delivery would need access to the customer's name and home address to be able to deliver the purchased clothes. The flow of personal data in this multi-cloud framework is illustrated in Fig. 1. The customer in the above scenario only ever interacts with the web interface of PrettyBigThing and will most likely be unaware of the data flow that occurs in cloud ecosystems which illustrates why transparency is such a stressed issue in GDPR. Further, each piece of information regarding the customer that is collected and shared in the above ecosystem is classified as "personal data" as per GDPR, thereby warranting a need for GDPR compliance verification. The customer in this situation is the "data subject". PrettyBigThing plays the role of "data controller" as it is the e-commerce site that is collecting the data subject's personal data and dictates the purposes for the collection, storage and processing of the data. Shopify, the payment provider, and the shipping provider are all "data processors" because they only process the data subject's personal data on behalf of PrettyBigThing.

3.2. GDPR requirements

We now describe the GDPR requirements that require verification based on data flow in the aforementioned multi-cloud framework.

1. **Consent.** PrettyBigThing, as the data controller, is required to inform the data subjects of the operations that are to be carried out on their personal data and enable the data subjects to provide/withdraw their consent for each operation.
2. **Data storage.** The customer's data should be stored in a form such that it can identify the data subject (thereby classified as "personal data" and as such requiring regulation) for no longer than is needed for the processing purposes set by the data controller.

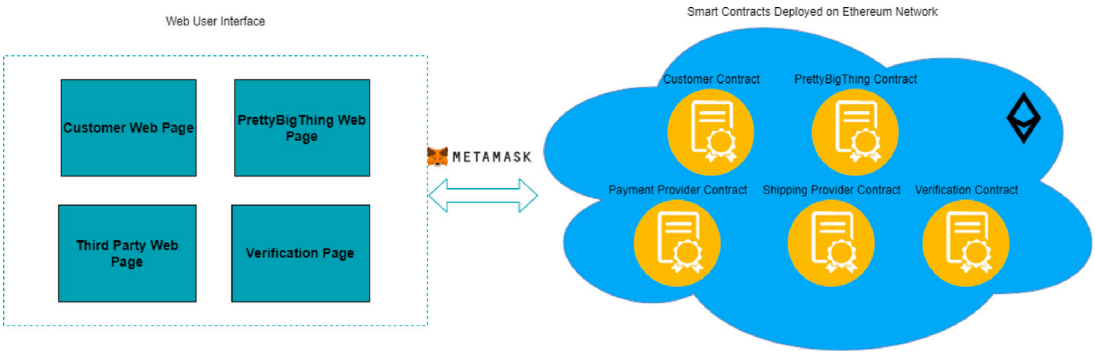

Fig. 2. A high-level overview of the proposed architecture.

3. **Data minimisation.** Personal data of the data subject that is collected, held and processed must be specific to the purposes set out by the data controller and such data should be limited to only what is necessary for the stated purposes.
4. **Data correctness.** The personal data held by PrettyBigThing must be accurate and kept up-to-date, where appropriate.
5. **Data protection.** It is the responsibility of the data controller (PrettyBigThing) to ensure that sufficient measures are taken to protect the data subject's collected personal data from unauthorised processing, accidental damage or loss throughout its life-cycle in the cloud.
6. **Recording Data Operations.** For PrettyBigThing to demonstrate GDPR compliance, and also to fulfil the data subject's "right to be informed", the data controller should record all data processing activities that occur on the customer's personal data.
7. **Reporting Data Processing Violations.** PrettyBigThing is required to report GDPR violations to the data subject (customer). Data processing violations will be conducting data operations for which the data subject has not provided informed consent.
8. **Data transfer.** PrettyBigThing is required to transfer personal data to data processors, such as shipping providers, given that the data subject (customer) has provided informed consent of the transfer operation and has been made aware of the purpose of the transfer. Further, only personal data that is relevant to the purpose is permitted to be transferred.

3.3. Architecture

The approach we take in this paper to achieve GDPR compliance in a multi-cloud environment scenario, such as the one described, leverages the transparency, security and automation capabilities of the Ethereum blockchain to create a user-centric data management framework where all personal data is stored on the blockchain and all GDPR relevant data operations take place through well-defined smart contracts deployed on the Ethereum network. The scenario we use to design our architecture is very generic and commonplace in multi-cloud ecosystems, making our architecture applicable for any cloud-based application that seeks to improve its users' privacy and trust. A high-level overview is illustrated in Fig. 2. For better understanding of our platform, we split it into two levels: the front-end and the back-end. The front-end is a web application consisting of a customer page, a page for PrettyBigThing, a third-party provider page and a page for GDPR verification. The back-end of our platform constitutes the smart contracts (as seen in Table 2) deployed on the Ethereum blockchain — these include a Customer Smart Contract, a PrettyBigThing Smart Contract, a Payment Provider smart contract, a Shipping Provider smart contract, and a GDPR verification smart contract. The actors of our GDPR framework (i.e. the customer, service providers, etc.) only ever interact with the web user interface. The web application, in turn, connects to the smart contracts and enables the actors to query and make transactions on the blockchain.

3.3.1. Web application

The web application consists of four webpages, one for each actor in our GDPR framework. Customers interact with the customer webpage, PrettyBigThing interacts with the PrettyBigThing webpage, the payment and shipping providers use the third-party webpage, and any authority checking for GDPR compliance can make use of the verification webpage. The webpages allow our actors to query and make transactions on the blockchain. The webpages connect to our deployed smart contracts using MetaMask,[4] a third-party web extension for enabling access to the Ethereum blockchain from the web browser. The customer webpage allows the customers to input their personal data, view the table of data processes proposed by PrettyBigThing, provide or revoke consent for these data processes, and also to view the detected GDPR compliance violations. The PrettyBigThing webpage is used to submit new data processes and allows PrettyBigThing to select and execute data operations. The verification page is used to view the logs of data operations that are carried out on our platform. The customer, PrettyBigThing, and third-party providers can also view the personal data stored on their smart contracts using their respective webpages. However, no personal data is stored by the web application.

[4] https://metamask.io

Table 2
All data operations, storage and GDPR verification smart contract.

	Customer	PrettyBigThing	Payment provider	Shipping provider	Verification
Stores	Data controller address	Encrypted user personal data	Encrypted user personal data	Encrypted user personal data	Table of data operations
	Encrypted customer personal data	Table of operations			Table of violations
	Table of operations and consent				
Methods	Revoke access	set new operation	Receive personal data	Receive personal data	Receive data operation logs
	Access operation	Execute operation	Hash function	Hash function	Verify data operations
	Edit operation	Hash function			Verify data correctness
	Transfer operation				
	Register new operation				
	Hash function				

3.3.2. Customer Smart Contract

The Customer Smart Contract is used to store and manage the customer's personal data. The customer inputs their personal data onto the customer webpage which, in turn, calls a method on the customer contract and stores the personal data in a table in the customer smart contract. To store the individual personal data fields (e.g., name, birth date, home address, etc.) we use the bytes32 data type of Solidity since it takes less memory than a string for the same length of the string. This results in the amount of characters for the personal data fields being limited to 32 ASCII characters. This is not an issue on our platform since, by the nature of the personal data fields, the personal data will not be longer than 32 characters. Further, storing personal data on the blockchain means everybody with a copy of the blockchain can potentially view the data, thereby violating GDPR. Therefore, the data on the blockchain is stored in encrypted form, with the encryption taking place off-chain (in the web app before it is fed to the customer contract) and only ciphertext ever being visible on the blockchain. Further, this way we also induce pseudonymisation, an aspect of the "safeguarding confidentiality" obligation, of the customers as no customers are identifiable from the data on the blockchain. Furthermore, the Customer Smart Contract is used to keep a record of the data operations proposed by PrettyBigThing alongside the consent the user has provided for each operation. Whenever PrettyBigThing sets new data operations, the default consent for the customer is set to be negative (as per the "privacy by default" obligation). This table of operations and consent is fetched by the web app and displayed on the customer webpage where customers are able to provide/revoke consent for each data operation. The customer's decisions are then passed back to the smart contract to update its consent records. Moreover, we encode GDPR relevant data operations such as access, edit and transfer onto the Customer Smart Contract. This way, execution of these operations is under the control of the user since, to carry out these operations, PrettyBigThing must call the appropriate methods of the Customer Smart Contract. Before the operations are carried out, they are verified by the consent table, thereby only allowing permissible actions to be carried out on personal data. Further, access to these methods is given only to the PrettyBigThing Smart Contract. If the methods are called by any address not matching that of the PrettyBigThing Smart Contract, then the call fails and the transaction is reverted. A customer can rescind method access permissions at any time (to execute their "right to restrict processing").

3.3.3. PrettyBigThing Smart Contract

The PrettyBigThing Smart Contract is used by the e-commerce provider and is interacted with using the interface on the PrettyBigThing webpage. This smart contract is used to submit new data operations. PrettyBigThing inputs the details of the new data operation — including the customer id, the purpose of the operation, the type of data operation (i.e. read/write/transfer), the specific data required, and the receiver of the data (in case of a transfer operation). The particulars of the operation are then passed to the smart contract. PrettyBigThing maintains a table of data operations that have been set by PrettyBigThing and, upon receiving details of the new operation, updates its own records and also sends the operation to the Customer Smart Contract to add the operation (with negative consent) to its consent table. The list of data operations on the smart contract is fetched and displayed on the PrettyBigThing webpage. The data operation PrettyBigThing selects for execution is sent to the PrettyBigThing smart contract which calls the appropriate methods on the Customer Smart Contract. PrettyBigThing also keeps a record of its own copies of the customers' data (stored as ciphertext) on its smart contract. This is data that is collected through transfer operations. To view the stored data, it needs to be fetched and decrypted by the web application. This means that the data subject needs to trust the integrity of PrettyBigThing and share the key for decryption via off-chain methods.

3.3.4. Third party provider smart contracts

The third party (payment and shipping) provider contracts are designed to receive data transfers from the customer smart contract. The data that is stored on these contracts is, of course, also encrypted and so, similarly to PrettyBigThing, the third party providers need to be completely trusted and a decryption key must be provided off-chain.

3.3.5. GDPR verification smart contract

All GDPR verification takes place through the GDPR Verification Smart Contract that is responsible for recording the data operations that take place. When recording these data operations, they are passed through a filter of GDPR related questions to detect violations. The data violations are then recorded into a "violations table". The verification contract is also used to detect incorrect data held by PrettyBigThing and the third party service providers. Methods in the customer, PrettyBigThing and third party provider contracts are called to calculate hash values for the data stored. These hash values are retrieved by the verification contract and compared to verify data correctness, with mismatching hash values resulting in a GDPR violation that is recorded in the violations table. The table of data operations is displayed on the Verification webpage and the table of GDPR violations is sent to the Customer webpage to inform the customers.

3.4. Data operations in smart contracts

We now go into more detail on the access, edit, transfer and storage operations in relation to the GDPR and how they are carried out on our platform. Each data operation that PrettyBigThing proposes is comprised of a:

1. **Customer ID** which is used throughout the platform to reference the customer about whom the operation is proposed.
2. **Purpose** for the data operation is also provided, especially for transfer operations since the data shared with a third party service provider must be relevant to the purpose fulfilled by the third party.
3. **Data operation** which will be the type of operation that is carried out on the data. The types of data operations we encode are access, edit, transfer and storage.
4. **Personal Data** refers to the particular data field that the operation is about, i.e. name, address, credit card details, etc. Each data operation concerns itself with a singular field of customer data, thereby enforcing "data minimisation".
5. **Data consumer** that receives the data in case of a "transfer" operation.

In the following, we give definitions of the data processes in the context of our framework and detail how they are realised on our platform.

3.4.1. Access operation

The access operation is a read-only data operation. PrettyBigThing is allowed to view the personal data but is not allowed to store the personal data. To carry out an access operation, PrettyBigThing accesses the PrettyBigThing webpage of our web user-interface from where they select an appropriate "access" operation from a list of data processes they have set out. The details of the access operation (e.g., the ID of the customer, the personal data PrettyBigThing would like to access, and the purpose for the operation) are sent to the PrettyBigThing smart contract. The PrettyBigThing smart contract will now make a call to the "Access Operation" method of the customer smart contract and pass the details of the operation. The "Access Operation" method in the customer smart contract first queries the table of operations and consent that is stored in the customer smart contract using the arguments of the operation passed to it by the BigPrettyThing smart contract. If the customer has not consented to the operation, the method ends and returns a message informing the PrettyBigThing contract that consent verification for the operation failed. If consent for the operation is positively verified, the "Access Operation" method finds the requested personal data for the relevant customer in the table of customer data. This personal data is returned to the PrettyBigThing smart contract and sent to the PrettyBigThing webpage where it is decrypted for viewing, however no personal data is stored by either the PrettyBigThing contract or the webpage.

3.4.2. Edit operation

The edit operation allows PrettyBigThing to change the contents of the customer's personal data. The data being edited is the data stored in the table of encrypted customer personal data stored in the customer smart contract. Similarly to the access operation, PrettyBigThing selects the edit operation through the PrettyBigThing webpage and the web application sends details of the edit operation to the PrettyBigThing smart contract (e.g., the ID of the customer, the personal data being edited, the purpose for the editing, and the new value for the personal data in an encrypted form). Once PrettyBigThing receives these operation details, it will call the "Edit Operation" method in the customer smart contract. The "Edit Operation" will query the table of operations and consent to verify that the customer has consented to the operation. If no consent has been provided, the "Edit Operation" method ends and informs the PrettyBigThing contract on the lack of consent. Otherwise, the method finds the relevant personal data of the customer and changes it to the new value provided by the PrettyBigThing. The "Edit Operation" method now ends and returns a message to the PrettyBigThing to inform the data controller of the successful edit.

3.4.3. Transfer operation

In GDPR terms, the transfer operation constitutes the intentional sending of the data subject's personal data to a data controller, data processor or third party. On our platform, a transfer operation constitutes personal data being copied from one smart contract and being sent to a different smart contract where it is then stored. The personal data can be transferred from the customer smart contract to the PrettyBigThing smart contract, transferred from the customer smart contract to the third-party payment or shipping provider smart contracts, or transferred from the PrettyBigThing smart contract to the third-party payment or shipping provider smart contracts. To execute a data transfer, PrettyBigThing selects the data transfer operation from the PrettyBigThing webpage and passes the details (including the customer ID, the personal data to be transferred, the purpose of the data transfer, and the recipient of the data transfer) to the PrettyBigThing smart contract. If the recipient of the data transfer is the PrettyBigThing smart contract,

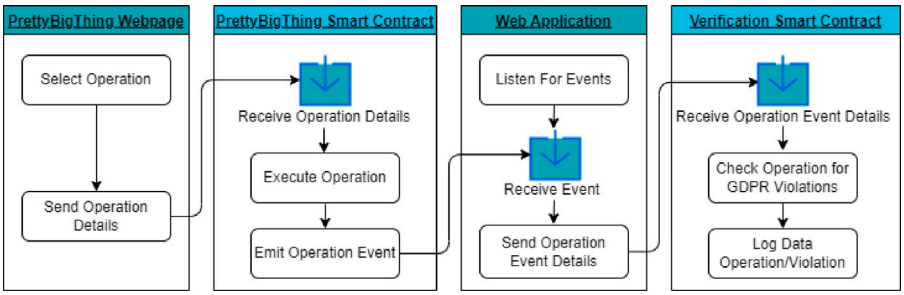

Fig. 3. Workflow to check GDPR compliance.

then it calls the "Transfer Operation" method of the customer smart contract and passes it the details of the transfer operation. The "Transfer Operation" method verifies consent for the operation in the same way as with the access and edit operations. Once consent is confirmed, the "Transfer Operation" fetches the relevant personal data from the customer data table. This personal data is then returned in the same fashion as in the access operation. However, once the PrettyBigThing contract receives the personal data, it stores this personal data in its own table of customer data. If the recipient of the data transfer is, instead, the third-party payment or shipping smart contracts, instead of calling the "Transfer Operation" method of the customer smart contract, the PrettyBigThing smart contract first checks its own customer data table. If the PrettyBigThing smart contract contains the required personal data, it will call the "Receive Personal Data" method of the recipient third-party contract and pass to it the required personal data. The "Receive Personal Data" method then stores the personal data in the table of "Encrypted Customer Payment/Shipping Data" of its own smart contract. IF the customer has not provided consent for this operation, it is detected and flagged as a GDPR violation by the verification smart contract. If the PrettyBigThing smart contract does not contain the required personal data, it must go through the process of calling the "Transfer Operation" of the customer smart contract. After verifying consent for the operation, it is the customer smart contract that calls the "Receive Personal Data" method of the third-party provider smart contract.

3.4.4. Storage operation

After a transfer operation, the transferred data is stored in the recipient smart contract. The data on the blockchain is in encrypted form. The encryption used is Advanced Encryption Standard (AES) for faster encryption, decryption and stronger security.

3.5. Revoking access to the Customer Smart Contract

The customer has the capability to revoke the access rights of the PrettyBigThing contract to the methods of the customer contract. This is to enable the customers to exercise their right to restrict processing. The customer smart contract stores an address variable to store the address of its data controller. This address variable is used to verify that the address of an external blockchain account trying to access the customer methods of the customer smart contract is indeed that of an authorised data controller. The customer smart contract contains methods, callable only by the owner of the customer smart contract, to edit this address variable. To add PrettyBigThing as the data controller for the customers, this address variable is set to the address of the PrettyBigThing contract so that the methods in the customer smart contract can be called from the PrettyBigThing smart contract. PrettyBigThing is removed as the data controller by setting the address variable to 0, the null equivalent of Solidity.

3.6. Monitoring data operations

We use events in Solidity to detect when PrettyBigThing carries out an operation on the customer's personal data and to read the result of the data operation. Events are essentially inheritable members of smart contracts that are triggered or emitted to store the log arguments of a transaction. Events are used to alert an application that calls a smart contract of the changes made to the contract, allowing the application to execute the appropriate actions. Since events are a part of the transaction receipts and transaction receipt hashes are stored inside blocks, events are verified to be correct by the blockchain. An event is emitted every time the PrettyBigThing smart contract carries out a personal data operation. The arguments passed to the event consist of the operation details (e.g., customer ID, purpose, data operation, data type, and data consumer) and the result of the operation. The result is what is returned when the PrettyBigThing contract calls the operation methods of the customer smart contract and is either a confirmation of the success of the operation or a reason for its failure. The web application listens for and receives the triggered events. The result of the data operation attempt is conveyed to PrettyBigThing and then the arguments of the event are fed to the verification smart contract. Inside the verification contract, the operation is filtered through a set of GDPR related questions to detect violations before being logged onto the blockchain. Fig. 3 demonstrates this monitoring procedure from when PrettyBigThing selects an operation to execute to its eventual recording on the blockchain.

3.7. Detecting and reporting violations

The verification contract, after receiving the data operation details, is used to carry out a GDPR compliance check on the personal data operations conducted by PrettyBigThing. In our framework, violations result from data operations being carried out without the customer's consent and also from incorrect personal data being stored by the data controllers and processors.

3.7.1. Consent violations

Any data processing operation that is carried out on the customers' personal data without their informed consent is considered a violation of GDPR. In our framework, informed consent is realised as the customer manually assenting to a data processing task constituting of a combination of: data operation (e.g., access, edit, transfer, and storage), purpose for the data operation, personal data required for the operation (e.g., name, home address, email address, and payment details), and the data consumer (the controller/processor requesting the data operation). Therefore, the only permissible data processing tasks are those where the customer has provided consent for a data processing task consisting of the attributes described, with any deviation of any of the attributes resulting in a new data process and requiring its own consent. Further, verifying that the combination of data operation, purpose, personal data, and consumer is appropriate (e.g. the personal data being transferred to a certain processor must be relevant to the purpose of that data processor) is also required.

Firstly, after receiving the data process details from the web application, the verification contract queries the customer smart contract to check whether the customer has revoked access rights of the PrettyBigThing smart contract, thereby exercising his/her right to restrict processing. If this is the case, then PrettyBigThing is violation of GDPR and the data process is logged as a violation. Otherwise, the verification contract queries the customer contract again to view the consent table and verify that the customer had provided consent for the data process. If consent verification fails, we have detected another GDPR violation and the contract logs this in a violation table. However, even if consent verification is passed, our task is not done. The purpose of the data process needs to be appropriate for the data consumer in the process (e.g., if the purpose is "payment", the data consumer should be the payment provider instead of the shipping provider). In addition, the personal data used in the data process must also be appropriate to the purpose of the process (e.g., if the purpose is "shipping", the personal data should be "home address" instead of "email address"). Finally, the verification contract checks that if the data process is a data transfer operation, then the purpose of the transfer must also be appropriate (e.g., in our scenario the transfer purpose can only be "payment" or "shipping"). If the data process passes through these GDPR questions without flagging any violations, it is logged as a normal data process, otherwise the data process is logged as a violation. The table of data processes in violation of GDPR is forwarded to the web application where they are displayed on the customer webpage in order to alert the data subject.

3.7.2. Incorrect data

The verification contract is also responsible for detecting the storing of incorrect personal data by any of the data controllers and data processors on our platform. The customer smart contract, the PrettyBigThing smart contract and the third party provider contracts all contain methods to compute the hash of the stored personal data. The verification contract confirms that only accurate personal data is stored with a data controller or data processor by calling these methods and comparing the hash to the hash generated by the customer contract. Therefore, the data subject plays a vital role in ensuring that only accurate personal data is used for processing and so the personal data the customer inputs onto the platform must be accurate. The hash function we use for verifying correct data storage is Keccak256, a cryptographic function built in Solidity. Keccak256 is an Ethereum implementation of SHA-3. Keccak256 takes any amount of inputs and converts them into a unique, 32 byte hash.[5]

4. Results

In this section, we detail the tests carried out to analyse the efficacy of our proposed solution. Further, despite the benefits of transparency, security, immutability, and automation of the blockchain, it come with transaction costs and we thoroughly investigate these transaction costs in every element of our framework.

4.1. GDPR compliance analysis

In this subsection, we discuss whether our framework meets the GDPR requirements detailed for our cloud-based e-commerce application.

1. **Consent:** The table of operations and consent in the customer smart contract is updated every time PrettyBigThing proposes a new data process with a record of the data operation (assigned negative consent). The customer is able to provide informed consent by viewing this table on the customer webpage and providing/revoking consent.
2. **Data storage and protection:** The security of the personal data of the customer is maintained by the cryptographic nature of the blockchain and strong AES encryption. Further, encryption of the personal data facilitates pseudonymisation, so the customer cannot be identified by the data stored on the blockchain.

[5] https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions

3. **Data minimisation and correctness:** Data minimisation is implemented in our framework by design as PrettyBigThing can only process a specific personal data type (e.g., name, address) in any single data processing operation. Data correctness is checked by our verification contract by comparing the hashes of the personal data stored with the data controllers/processors with the hash of personal data kept with the data subject. Incorrect data storage is logged as a GDPR violation and informs the data subject and controller of the need to update the stored personal data.
4. **Recording data operations and reporting violations:** Every data operation that takes place passes through a set of GDPR related questions in our verification contract to check compliance. Data operations that pass the GDPR filter are logged as normal data operations, those that do not are flagged as violations, with violation logs being sent to the customer webpage for reporting. Key interest is taken in the data transfer operation in our GDPR compliance filter to ensure that the customer has provided informed consented to all transfers and that all transfer operations are appropriate to the purpose of the recipient service provider.

4.2. Validation of smart contracts

A transaction fee is required from any sender wishing to execute transactions on a blockchain. This transaction fee pays for the computational power required to process the transaction on the blockchain and can also be used to accelerate a transaction, with higher transaction fees giving a transaction higher priority on the blockchain network. Transaction fees can fluctuate widely based on the amount of available block space on the blockchain. Block space is the amount of available space in a block being sent to the blockchain and an absence of block space leads to higher transaction fees and delayed transactions (e.g. higher mining times). In Ethereum, this transaction fee is referred to as the gas price and is paid using ether (ETH), the native cryptocurrency of Ethereum. The gas price is represented in terms of Gwei, a denomination of Ether.[6] Gas is used to purchase the computational resources of the Ethereum network and most importantly, to utilise smart contracts and decentralised applications built on Ethereum. Therefore, we conduct a thorough investigation into the gas fees payable for all elements of our framework that would make a transaction on the blockchain network. For testing purposes, to avoid the volatility and expenses of the main network, we employ the use of two Ethereum test networks (or testnets). An Ethereum testnet is essentially a collection of nodes used to test smart contracts in a production-like environment without incurring the costs of transacting on the main network. The two testnets used in our testing are Ropsten and Ganache. The Ropsten testnet is the most faithful emulation of the Ethereum mainnet because it uses the PoW consensus mechanism, making it a very appealing testing ground for our smart contracts. Ganache, unlike the public testnet Ropsten, is a personal blockchain network. A benefit of using Ganache for our testing purposes are that it greatly simplifies smart contract testing since we are able to quickly launch a completely local blockchain network to deploy smart contracts and execute transactions quasi-immediately since no mining takes place so we do not have to wait for a mining time. Mining time is the amount of time taken for miners on a blockchain network to verify a transaction inside a block and append the block to the blockchain. The host device of the Ganache blockchain network is a Dell Inspiron 5579, 2015 running Microsoft Windows 10 Home with Intel Core i7-8550U CPU and 16-GB memory. The state of the blockchain (May 2022) where these contracts were executed: average block size (87744 B), blockchain size (687.49 GB), average block time (13.44 s), hash rate (1064.27), reward per block (13202)[7].

4.2.1. Contract deployment

To evaluate the gas costs of smart contract deployment, we deploy each of our smart contracts on both the local Ganache network and the public Ropsten network. Due to the instability of public testnets, we deployed the smart contracts on Ropsten five times and calculated the average transaction costs and mining times. The resulting gas prices and mining times for our smart contract deployments are recorded in Table 3. The gas fees recorded for the smart contract executions on the Ganache contract appear to be as expected. The third-party smart contract requires considerably less gas than the other contracts due to it containing the least functionality whereas the customer smart contract and the PrettyBigThing smart contract use up the most gas. On the Ropsten network, not only are the gas prices for the deployment of our contracts considerably higher but, also, seem to be heavily affected by the instability of public testnets as the third-party contract is the second most expensive contract. Further, the table also shows the volatility in the mining times on the Ropsten network since there does not seem to be any relationship between the gas price and mining time, with the most expensive contract (the customer contract) taking significantly less time to mine than the least expensive contract (the PrettyBigThing contract). However, due to the strong likeness of the Ropsten network to the main Ethereum network, these results give useful insight into what we can expect of the performance of our smart contracts on the mainnet. Furthermore, for the remainder of our testing, we only use the Ganache network to test transaction costs for the functions in our smart contracts. This is due to the fast response time and the amount of available test Ether of the Ganache network but also due to the impracticality of running a large number of transactions on the Ropsten network with limited Ether and the added variable of mining time. Additionally, the Ropsten network has served its purpose in demonstrating what we can expect from the Ethereum mainnet in comparison to the Ganache network in Table 3.

[6] 1 Ether is equivalent to 1,000,000,000 Gwei.

[7] https://ycharts.com/indicators/reports/ethereum_statistics

Table 3
A comparison of the deployment costs of our smart contracts.

Smart contract deployment costs			
Contract	Ganache	Ropsten	
	Gas used (gwei)	Average gas used (gwei)	Average mining time (s)
Customer	2321971	6313558	33.8
PrettyBigThing	2450937	4615104	90
Third Party	805630	6009999	99
Verification	1630286	5845564	41

Table 4
The gas costs of adding/removing PrettyBigThing as the data controller.

Set/remove service provider costs (gwei)	
Set service provider	Remove service provider
42842	28549

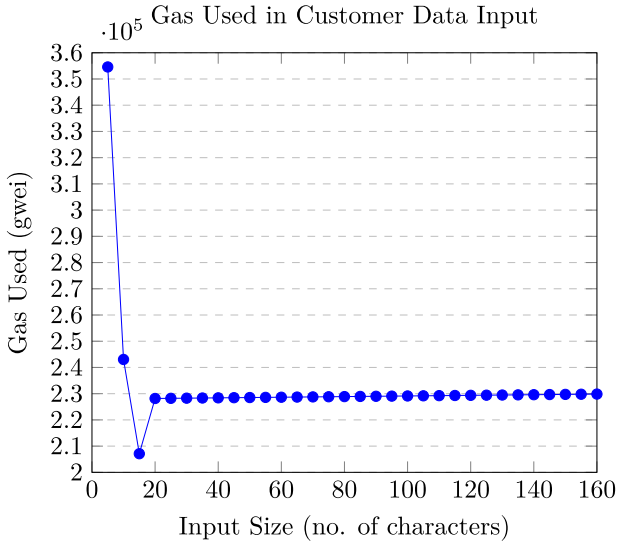

Fig. 4. Increase in gas cost for customer's personal data vs increase in input size.

4.2.2. Data input

In Fig. 4, we illustrate how the gas price of the data input operation changes as the amount of characters in total involved in the personal data input operation increases from 5 to 160 (e.g., we start off with 1 character in each data field and increase this up to 32 characters in each field). Further, we are also testing the scalability of our platform since each individual plot in the graph represents a new customer being added to our platform. The first customer is added with just 1 character for each personal data field and the final customer is added with 32 characters used for each personal data field. So, from left to right on our graph, we have a progression of customers being added to the customer smart contract. The graph shows that the first data input operation costs a considerably amount of gas more than the proceeding customer inputs. This is likely due to the initialising of the table of customer data in each of the customer, PrettyBigThing, and third-party provider contracts. The data input operation cost from the fourth customer onward stabilises and the increase in cost, a linear increase of 60 Gwei per additional customer, is negligible.

4.2.3. Setting/removing service provider

When PrettyBigThing is added to the customer smart contract as the data controller for the customers, the address of the PrettyBigThing smart contract is stored in an address variable for verifying the message sender when external users call the customer smart contract methods. When it is removed as the data controller, that address variable is set to 0. There is no null property in Solidity so explaining why the difference in gas costs of the operations in Table 4 is not greater however it is still interesting that removing PrettyBigThing as the data controller uses less Gas.

4.2.4. Set operation

We also measure the gas costs of PrettyBigThing proposing new data process operations in Table 5. This transaction involves updating a table in the PrettyBigThing and customer contracts therefore requires a transaction on the blockchain. Transfer 1 refers

Table 5
The gas costs of proposing new data processes.

Set operation transaction costs (gwei)				
Access	Edit	Transfer 1	Transfer 2	Average
215614	188493	188565	227288	**188565**

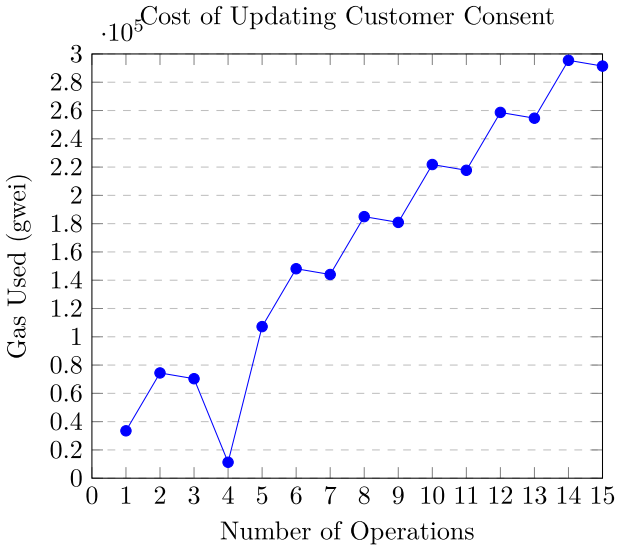

Fig. 5. The amount of gas used to update the consent table increases with the number of data processes that the customer wishes to provide/revoke consent for.

to personal data being transferred to the data controller, PrettyBigThing, and Transfer 2 refers to data being transferred to data processors, third-party payment and shipping providers. Unsurprisingly, the costs of proposing the different data process operations are very similar since the procedure is almost exactly the same. Also, this function is slightly cheaper than inputting customer data, since the process of creating a new customer involves updating tables in the third-party contracts on top of the customer and PrettyBigThing contracts.

4.2.5. Consent updates

The costs of updating the customer's consent for data processing operations is illustrated in Fig. 5. On the customer webpage, the customers are able to select multiple data processing operations and update their consent to the selected operations simultaneously. Therefore, when testing the function to update user consent, we consider the gas costs when the user selects to update consent for multiple data processing operations. This test involves randomly selecting data processes and updating the user's consent for those operations to be the opposite of what it used to be (e.g., revoke consent if it was previously provided or provide consent that was previously refused). From the graph in Fig. 5, it is clear that the cost of updating consent generally increases as the number of updates required increases.

4.2.6. Data processing operations

The data processing operations our framework handles are data access, data edit, and data transfer, described in detail in the solution. Within the data transfer operation, there are three different approaches our solution can take. These include data being transferred from the customer contract to the PrettyBigThing contract, data being transferred from the customer contract to a third-party provider contract, or it could be data being transferred from the PrettyBigThing contract to a third-party provider contract. Further, for each of these data processing operations, the size of personal data being processed can range from just one character to up to 32 characters. In Fig. 6, we show and compare the gas prices each data processing operation incurs as we increase the number of characters being processed in the operation. Furthermore, to gain a more complete picture on the working of the data transfer operations in our framework, for each data transfer operation, we start off with the recipient smart contract not containing any customer data. The test involves carrying out the data transfers of the personal data, from the size of 1 character to 32 characters, in succession such that the recipient contract starts off being empty and then receives a data transfer of size 1 character and so on until it has received 32 data transfers. The gas required for these operations is shown in Fig. 6 and shows that the first transfer, the transfer of just 1 character is considerably more expensive than the other transfers. This is due to the smart contracts initialising their customer data tables upon receiving the first data transfer.

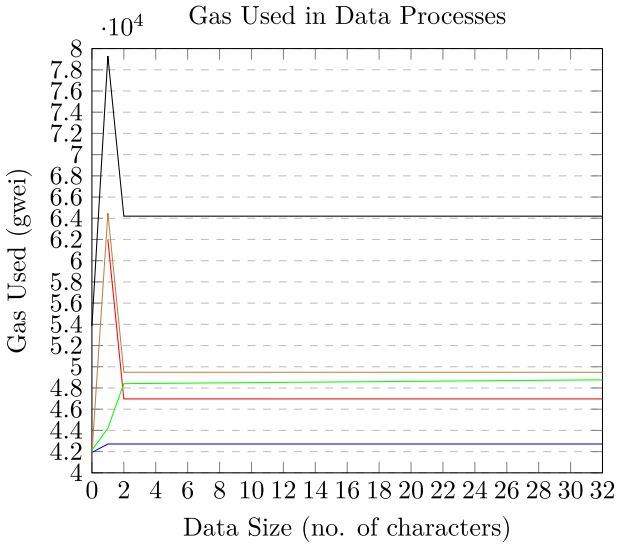

Fig. 6. A comparison of the gas used by each type of data process on our platform as the size (in terms of characters) of the personal data being processed increases.

The most expensive operation is transferring data from the customer contract to a third-party provider contract, shown in black. Compared to transferring data to the PrettyBigThing contract from the customer smart contract or transferring data from the PrettyBigThing contract to a third-party contract, shown in brown and red respectively, transferring data from the customer contract to a third-party contract involves an extra call to an external smart contract. This shows that calling methods in external smart contracts can be expensive and is something to be wary about. The cheapest operation is, as expected, the access operation, shown in blue. Unlike the other options, the access operation does not involve making changes to stored data and its main expense comes from external contract calls from the side of the PrettyBigThing contract. Other than the data edit operation, shown in green, the gas price for the data processes does not rise with the increasing data size. Even for the data edit operation, the rise in Gas cost is negligible. Similarly to the operation of inputting customer data, all of the personal data being processed is stored using the bytes32 data type for which all values have the same size, therefore despite the increase in ASCII characters, the size of the bytes32 variable is fixed and so does not increase the gas price of the data processing operations. Finally, we also consider the gas prices for when each of these data processing operations fail (e.g., being denied access to the customer smart contract or not having consent for an operation). The plot for this is shown at data size = 0, and is, unsurprisingly, cheaper than a successful operation. We do not consider failure for the data transfer from the PrettyBigThing contract to a third-party contract since it bypasses the customer contract and cannot not fail.

4.2.7. GDPR verification

We now discuss the gas costs for verifying GDPR compliance on our framework. Firstly, for the process of logging data in Table 6, we measure the gas required for each scenario of the logging process. The cheapest scenario is one where the data process passes the GDPR verification filter as this means that only the logging table is updated with the new data process. All other scenarios result in adding a new entry into the violations table, thereby raising the gas requirement, and have similar gas costs. Similarly, we also test all scenarios of the operation for verifying data correctness in Table 7. These scenarios are the different results of the data integrity check. These results include all data controllers/processors passing the hash checks, only two passing the hash check, only one data controller/processor passing the hash check, and none of the data controllers/processors passing the hash check and all store incorrect data. As expected, the cheapest scenario is one where all data controllers/processors pass the hash checks and no violations are logged. The second cheapest scenario is where two data controllers/processors pass the check and only one violation is recorded and so on, with the most expensive scenario being where all data controllers/processors fail the hash check and three violations are added to the violation table. A recurring theme in all of our tests is that the most expensive transactions are those that consist of either adding data to the blockchain or involve calls to external smart contracts. Transactions that contain fewer features such as these result in being considerably less expensive.

4.3. Average throughput

The average throughput was computed based on the start and end time concerning the processing of transactions generated within or from outside the designed smart contracts with respect to the time interval. Fig. 7 depicts the variation of throughput with respect to an increase in the number of requests. The results are depicted for 10 transaction requests (plotted as 0 to 9) that represents 10–100 transaction requests. Initially, the throughput was lower adjusting the system as per initial configurations and then it reached to stability and remained almost constant thereafter.

Table 6
The amount of gas used for the different results of a data process logging operation.

Data process logging costs	
Verification result	Gas used (gwei)
Verification passed	218897
Access revoked	384857
Consent violation	350675
Purpose violation	370959
Irrelevant data violation	370784
Inappropriate transfer purpose	370161

Table 7
The amount of gas used for the results of the data correctness verification operation.

Costs of checking data integrity	
Verification result	Gas used (gwei)
Passed by 3 controllers/processors	52401
Passed by 2 controllers/processors	161940
Passed by 1 controllers/processors	267286
Passed by 0 controllers/processors	282286

Fig. 7. Average throughput (in tps).

4.4. Average latency

The average latency for the proposed approach is considered with respect to the time consumed between the submission of request and the completion of transaction logging process. Fig. 8 depicts the variation of latency with respect to an increase in the number of requests. The results are depicted for 10 transaction requests (plotted as 0 to 9) that represents 10–100 transaction requests. Initially, the latency was higher adjusting the system as per initial configurations and then it reached to stability and remained almost constant thereafter.

4.5. Comparison with existing proposals

Although, we cant find an exact match in the existing proposals with which we can compare the results obtained from the proposed work, but have tried to draw some comparative evaluations with respect to different parameters. To validate the proposed work in contrast with the existing similar variants, we have considered two existing works, i.e., [22,23]. First, we have compared our work with [22] to understand the gas cost (in gwei) for deploying different smart contracts. Table 8 shows the comparison of Gas cost (in gwei) for deploying smart contracts. Although, there exists some dissimilarities in the functions designed in these smart contracts but still they can be compared being the closest match. The proposed scheme shows lower gas cost used to deploy different smart contracts in contrast to [22]. It may be interesting to note here, that the number of functionalities in the designed contracts in the proposed work are far more than the existing counterpart, still they showcase lower gas cost, proving them to be efficient. Second, we have compared the average mining time for deploying and validating the designed smart contracts in contrast to [23].

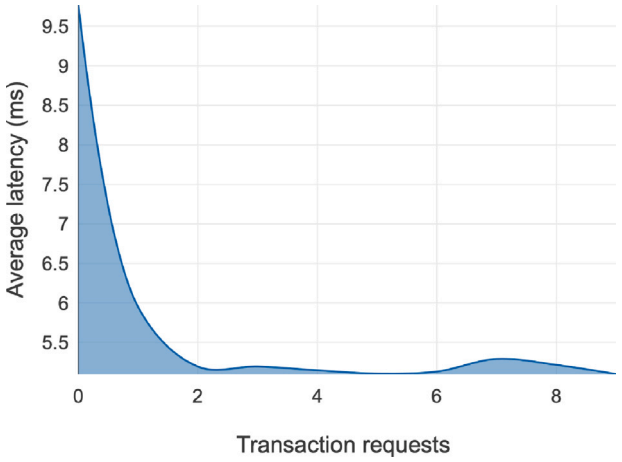

Fig. 8. Average latency (in ms).

Table 8
Comparison of gas cost (in gwei) for deploying smart contracts.

Smart contracts	[22]	Proposed work
Consent	8796531	2321971 (including data storage)
Submission (PrettyBigThing contract)	5008068	2450937
Transfer	–	805630
Verification	1706655	1630286

Table 9
Comparison of mining time (in seconds) for deployment of smart contracts.

Metric	[23]	Proposed work
Average mining time (including all contracts)	257	263.8

Table 9 shows the comparison of Mining time (in seconds) for deployment of smart contracts. It is evident that the proposed scheme shows a slightly higher mining time in contrast to [23]. However, it is just marginal as compared to the additional functionality (as shown in Table 1) the proposed scheme provides in contrast to the existing counterpart.

5. Conclusions

With user data being collected and distributed amongst multiple cloud providers, tracking the data operations carried out on the user's personal data to detect unlawful processing is a challenging problem. To this end, many blockchain-based solutions have been proposed in order to create immutable records of data processes and automate GDPR verification. We present a cloud-based e-commerce application scenario and use it to assess the GDPR requirements a typical cloud-based web application must follow to ensure GDPR compliance. We use this to motivate the design and implementation of our own user-centric, blockchain-based data management framework for sharing personal data with data controllers and data processors. In our framework, all personal data is stored in encrypted form on the blockchain and GDPR-relevant data operations such as access, edit, transfer, and storage take place on the blockchain through well-defined smart contracts deployed on the Ethereum network. The actors in our GDPR framework (the data subject, the data controllers and processors, and the GDPR regulatory authority) interact with our smart contracts through a web user-interface. Personal data is stored in a customer smart contract and all data operations take place through well-defined methods in the customer smart contract. This places the data subject at the centre of our GDPR compliance framework, a feature very few other approaches possess, and increases user trust in the service providers utilising this framework. All data operations that take place on the blockchain are detected and passed through a series of GDPR-related questions to verify compliance inside a verification smart contract. GDPR violations are logged and reported to the user via the web application.

In this paper, we have not evaluated the scalability of the work to a great extent. In future work, we would like to use off-chain storage and test the scalability of our platform by introducing more data controllers and data processors. Ethereum has recently moved on to a new version that considers Proof of Stake as a consensus mechanism instead of PoW, so it would be interesting to see how, the proposed scheme performs looking into the changes witnessed from ethereum 1.0 to ethereum 2.0.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Aujla GS, Barati M, Rana O, Dustdar S, Noor A, Llanos JT, Carr M, Marikyan D, Papagiannidis S, Ranjan R. COM-PACE: compliance-aware cloud application engineering using blockchain. IEEE Internet Comput 2020;24(5):45–53.

[2] Barati M, Rana O. Privacy-aware cloud ecosystems: Architecture and performance. Concurr Comput: Pract Exper 2020;e5852.

[3] Demirbaga U, Aujla GS. MapChain: A blockchain-based verifiable healthcare service management in IoT-based big data ecosystem. IEEE Trans Netw Serv Manag 2022;1. http://dx.doi.org/10.1109/TNSM.2022.3204851.

[4] Wang S, Ouyang L, Yuan Y, Ni X, Han X, Wang F-Y. Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Trans Syst Man Cybern Syst 2019;49(11):2266–77.

[5] Elluri L, Nagar A, Joshi KP. An integrated knowledge graph to automate gdpr and pci dss compliance. In: 2018 IEEE International conference on big data (Big Data). IEEE; 2018, p. 1266–71.

[6] Barati M, Theodorakopoulos G, Rana O. Automating GDPR compliance verification for cloud-hosted services. In: 2020 International symposium on networks, computers and communications. IEEE; 2020, p. 1–6.

[7] Zemler F, Westner M. Blockchain and GDPR: Application scenarios and compliance requirements. In: 2019 Portland international conference on management of engineering and technology. IEEE; 2019, p. 1–8.

[8] Liang X, Shetty S, Tosh D, Kamhoua C, Kwiat K, Njilla L. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: 2017 17th IEEE/ACM International symposium on cluster, cloud and grid computing. IEEE; 2017, p. 468–77.

[9] Sigwart M, Borkowski M, Peise M, Schulte S, Tai S. Blockchain-based data provenance for the internet of things. In: Proceedings of the 9th international conference on the internet of things. 2019, p. 1–8.

[10] Sigwart M, Borkowski M, Peise M, Schulte S, Tai S. A secure and extensible blockchain-based data provenance framework for the internet of things. Pers Ubiquitous Comput 2020;1–15.

[11] Ruan P, Chen G, Dinh TTA, Lin Q, Ooi BC, Zhang M. Fine-grained, secure and efficient data provenance on blockchain systems. Proc VLDB Endowm 2019;12(9):975–88.

[12] Zyskind G, Nathan O, et al. Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and privacy workshops. IEEE; 2015, p. 180–4.

[13] Kaaniche N, Laurent M. A blockchain-based data usage auditing architecture with enhanced privacy and availability. In: 2017 IEEE 16th International symposium on network computing and applications. IEEE; 2017, p. 1–5.

[14] Kirkman S, Newman R. Using smart contracts and blockchains to support consumer trust across distributed clouds. In: Proc. of the 13th International conference on grid, cloud, and cluster computing. 2017, p. 10–6.

[15] Kirkman S, Newman R. A cloud data movement policy architecture based on smart contracts and the ethereum blockchain. In: 2018 IEEE International conference on cloud engineering (IC2E). IEEE; 2018, p. 371–7.

[16] Neisse R, Steri G, Nai-Fovino I. A blockchain-based approach for data accountability and provenance tracking. In: Proceedings of the 12th International conference on availability, reliability and security. 2017, p. 1–10.

[17] Zheng X, Mukkamala RR, Vatrapu R, Ordieres-Mere J. Blockchain-based personal health data sharing system using cloud storage. In: 2018 IEEE 20th International conference on E-health networking, applications and services (Healthcom). IEEE; 2018, p. 1–6.

[18] Westerlund M, Neovius M, Pulkkis G. Providing tamper-resistant audit trails with distributed ledger based solutions for forensics of IoT systems using cloud resources. Int J Adv Secur 2018;11(3):4.

[19] Barati M, Petri I, Rana OF. Developing GDPR compliant user data policies for internet of things. In: Proceedings of the 12th IEEE/ACM International conference on utility and cloud computing. 2019, p. 133–41.

[20] Barati M, Rana O, Theodorakopoulos G, Burnap P. Privacy-aware cloud ecosystems and GDPR compliance. In: 2019 7th International conference on future internet of things and cloud (FiCloud). IEEE; 2019, p. 117–24.

[21] Barati M, Rana O, Petri I, Theodorakopoulos G. GDPR compliance verification in Internet of things. IEEE Access 2020;8:119697–709.

[22] Barati M, Rana O. Tracking GDPR compliance in cloud-based service delivery. IEEE Trans Serv Comput 2020.

[23] Barati M, Aujla GS, Llanos JT, Duodu KA, Rana OF, Carr M, Rajan R. Privacy-Aware cloud auditing for gdpr compliance verification in online healthcare. IEEE Trans Ind Inf 2021.