Received February 5, 2021, accepted February 22, 2021, date of publication March 9, 2021, date of current version March 18, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3064896

PETchain: A Blockchain-Based Privacy Enhancing Technology

IBRAHIM TARIQ JAVED[1], FARES ALHARBI[2], TIZIANA MARGARIA[1], NOEL CRESPI[3], AND KASHIF NASEER QURESHI[4]
[1]Lero—Science Foundation Ireland Research Centre for Software, University of Limerick, V94 T9PX Limerick, Ireland
[2]Computer Science Department, Shaqra University, Riyadh 11961, Saudi Arabia
[3]Institut Polytechnique de Paris, Telecom SudParis, 91011 Evry, France
[4]Department of Computer Science, Bahria University, Islamabad 44000, Pakistan

Corresponding author: Ibrahim Tariq Javed (ibrahimtariq.javed@lero.ie)

This work is partly supported with the financial support of the Science Foundation Ireland grant 13/RC/2094_P2 and partly funded from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754489.

ABSTRACT With the increasing use of smart devices and sensors, enormous amounts of data are being generated continuously. The data is commonly stored in centralized cloud platforms and consumed by different services. The data is indeed a valuable resource for many service providers who provide advanced features and utilities to their subscribers. However, user data include personal and sensitive information which can be misused in many ways. There is no way for a subscriber to confirm that their service provider is compliant with data privacy regulations. The existing privacy enhancing techniques such as anonymization and differential privacy substantially reduce data usability while ensuring privacy. Therefore, it remains essential to provide a feasible solution that allows service providers to take advantage of user data while guaranteeing their privacy. In this paper, we present PETchain: a novel privacy enhancing technology using blockchain and smartcontract. In PETchain, data is stored securely in a distributed manner and processed in a user-selected trusted execution environment. Users deploy the smartcontract that allows them to decide whether and how their data can be exploited by service providers. The feasibility and performance of PETchain are presented by implementing PETchain over a consortium Ethereum blockchain.

INDEX TERMS Blockchain, Ethereum, privacy enhancing technology, privacy preservation, smartcontract.

I. INTRODUCTION

The rapid development in the fields of IoT, social networks, and cloud computing has led to the increased generation, storage, and processing of personalized data. In this new era of big data, both public and private service providers are collecting large amounts of data from their users to provide them with enhanced services and features. However, most of the collected contains private and confidential information that can be easily abused. Service providers focus on providing strong authentication, integrity, and confidentiality solutions but generally under-look user's privacy while collecting, storing, and processing their personal data [1]. Once a service provider acquires the data it can easily misuse or distribute it without user consent. Thus, users must completely trust their service providers and have little or no control over their data. Furthermore, the users are unable to define and implement access control for their data, to decide who and when can access their data, and how can they process it.

The General Data Protection Regulation (GDPR)[1] came into act on 25 May 2018 in the European Union. The GDPR contains provisions and requirements to protect users' rights related to their data. This includes the user's right to remain informed about their data being gathered, processed, and distributed by their service provider. It also provides the user with the right to have their data updated or even deleted forever [2]. However, users, for now, do not have any ability to check whether their service providers are GDPR compliant or not. Users have to completely trust their service providers with their data if they wish to use their services. Furthermore, if any investigation on a privacy breach is conducted, the supervisory authority does not have any reliable or auditable logs to inspect. Authorities must rely on the logs maintained by the service providers themselves.

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran Tariq.

[1]https://gdpr-info.eu/

To protect and improve user privacy several privacy-enhancing technologies (PET) have been introduced in the literature. The most widely adopted include homomorphic encryption [3], anonymization [4], and differential privacy [3]. Homomorphic encryption has a significant computational overhead for data processing which makes it incompatible with most of the current client-server applications. On the other hand, anonymization and differential privacy reduce the originality of the data, thus limiting the service providers' ability to derive value from user data. Recently, the utilization of blockchain technology is considered to be a promising solution to preserve user privacy due to its widely recognized accuracy and integrity features [1]. DIN SPEC 4997 [5] describes technical and organizational measures for data protection as well as requirements based on GDPR. It further presents architectural blueprints to illustrate how to use blockchain to improve data privacy. Therefore, in this paper we propose PETchain: a user-centric PET solution that utilizes blockchain technology along with smartcontracts, InterPlanetary File System (IPFS), and Trusted Execution Environment (TEE). PETchain allows users to securely store their data in a distributed manner and implement their data access policy in an accountable and auditable manner. Thus, PETchain is suitable to ensure privacy in data management and sharing applications. It further allows service providers to provide services to their users without keeping a copy of their data.

The major contributions of this paper are as follows:

1) A novel privacy enhancing technology, "PETchain", that allows service providers to utilize user data while guaranteeing their privacy. PETchain allows users to securely store and maintain their data in IPFS. Data owners are given complete control over their data by deciding who can access, process, and utilize their data. The data is processed in a user-selected isolated secure environment, assuring the confidentiality and integrity of the data. Logs of each data access are maintained in an immutable and auditable manner in the blockchain.
2) A smartcontract that allows users to define their access control policy. The PETchain smartcontract consists of several functions. With the set_identifier function the user stores data identifiers of the data files uploaded over IPFS, whereas with the set_authorization function users authorize service providers to utilize their data. Get_identifier is the only function that can be called by the service provider to request data access. Moreover, with the destroy_smartcontract and pause_smartcontract functions the users can pause and remove their smartcontract at any time.
3) An implementation of PETchain over a consortium blockchain. We choose to use a consortium blockchain as it maintains user control and privacy while having reduced cost and high throughput when compared to the public blockchain. Furthermore, unlike a private blockchain network, it does not consolidate power to a single party and distributes it across different organizations. We analyze the performance of PETchain on a consortium blockchain by using four performance metrics: transaction GAS cost, transactions per second, number of lost blocks, and propagation delay. Different parameters such as sealers, block-time, and block-gas-limit were analyzed to achieve the best possible results for our consortium blockchain. We choose to implement our solution using the Ethereum platform as it facilitates developers to deploy their smart contracts in a secure and simple manner. However, PETchain can be implemented over any consortium blockchain platform that supports the deployment of smart contacts.

The rest of the paper is structured as follows: the related work is described in Section II highlighting existing PET solutions. In section III, we provide an overview of the blockchain technology, describing types and advantages of blockchains, consensus agreement, and smartcontracts. In section IV, we introduce and detail the PETchain solution to ensure the privacy of users in a holistic manner. In Section V, we describe how we implemented our solution on Ethereum and provide its performance analysis. Lastly, we offer our conclusions in Section VI.

II. RELATED WORK

In this section, we detail the related work by presenting different PET solutions from the literature. As illustrated in Figure 1, this includes anonymization, pseudonymization, homomorphic encryption, differential privacy, data summarization, and decentralized learning. We describe the benefits and limitations of each PET solution. We further present a few prototypes that utilize blockchain technology to improve user privacy.

Anonymization and pseudonymization techniques are used to remove personal information from the dataset allowing data owners to remain unidentifiable [6]. Anonymization is completely irreversible whereas pseudonymization allows the data owner to be re-identified by using some additional information. The three most common techniques are K-anonymization [4], L-diversity [7] and T-closeness [8]. K-anonymity is achieved using suppression and generalization methods until each row of the dataset becomes identical to at least $k - 1$ other rows. L-diversity is based on k-anonymity to address homogeneity and background knowledge attacks [9], whereas T-closeness [8] reduces attribute disclosure by decreasing the granularity of the data. These techniques are being widely adopted to achieve user privacy. However, they consistently reduce the originality of the information by decreasing the accuracy and precision. This highly affects the utility of the data. Higher levels of anonymization enhance privacy but make data unproductive and ineffective.

Homomorphic encryption allows procedures to be performed on encrypted information, guaranteeing the same output as if the operations were performed on the clear data. This property enhances privacy by enabling applications

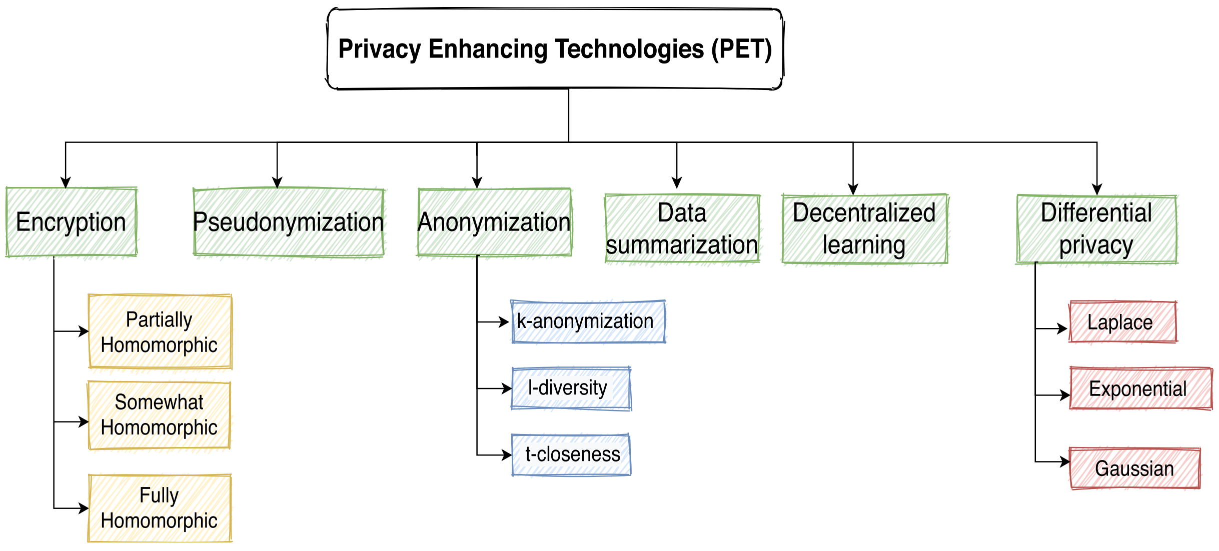


FIGURE 1. **Taxonomy of privacy enhancing technologies.**

to process, store, and share encrypted data rather than the original data. Homomorphic encryption can be further categorized into partial, somewhat, and total homomorphic encryption. Partial homomorphic encryption can only support one operation, whereas somewhat homomorphic encryption can support two types of operations together. However, both partial and somewhat homomorphic encryption techniques can only support basic operations types such as addition and multiplication. For this reason, they don't apply to scenarios that require advanced analytical computations to process the data. On the other hand, total homomorphic encryption supports a vast variety of analytical operations on encrypted data. However, this comes at the expense of computational latency: homomorphic encryption is impractically slow as it has a large computation overhead when applying operation on encrypted data. It usually takes 2-5 seconds per operation, which is incredibly slow for most practical uses [3]. Therefore, unless the computational overhead is substantially decreased, homomorphic encryption algorithms remain counterproductive for data-hungry applications.

Decentralized learning [10] allows sensitive data to be offloaded to end-user devices where the processing can be done. This enhances privacy by eliminating the risk of exposure of sensitive data, but it overburdens the user devices. Therefore, decentralized learning also remains impractical for applications that require immense computations similar to homomorphic encryption. Moreover, decentralized learning is also prone to attacks that expose intermediate results and leak information about user data.

Differential privacy techniques allow to collect and share aggregate information about users while maintaining the privacy of individual users. This is done by adding statistical noise to each user's data before it is shared [11]. The aggregate information from the dataset can then be extracted by deducting the noise from it. However, information regarding a particular individual cannot be derived. Three major strategies are used in differential privacy: Laplace, Exponential, and Gaussian [12]. The Laplace and Gaussian strategies are typically utilized for numerical datasets, while exponential techniques are used for non-numerical datasets. Differential privacy is considered to be an effective method to gather aggregate user information while preserving the privacy of individuals. However, privacy is achieved at the expense of data accuracy. The amount of noise added highly impacts the precision and accuracy of data. The trade-off between data precision and privacy is controlled using an operator referred to as epsilon [13]. Concealing sensitive information requires a high value of epsilon, which affects the usefulness of the information. Furthermore, differential privacy is only applicable to scenarios where aggregate information is useful and cannot be used for personalized services. Similarly, data summarization techniques [14] are also not applicable to personalized services as they create summarized variants of data sets that hide the actual information of individuals.

After the success of Bitcoin [15], researchers started paying more attention to the underlying blockchain technology. Due to its cryptographic security, immutability, and accountability features blockchain is useful in many non-financial fields. Over the last few years, researchers started using blockchain technology as a decentralized access control moderator in various areas where access control policies are defined by using smartcontracts. FairAcces [16] provides a secure access control mechanism for IoT applications. A proof of concept is presented using Raspberry PI and local blockchain to grant, delegate, and revoke access. In [17] smartcontracts are used to provide attribute-based access control for cloud storage systems. A role-based access control is presented in [18] in which user relationships are publicly visible on the blockchain. [19] presents a user-driven access control framework for the decentralized online social network which allows user to define their privacy policies. The main advantage of these access control mechanisms is auditability, as blockchain handles each transaction in a decentralized, transparent, and immutable manner. However, these mechanisms are designed to provide access control in a particular scenario where privacy is not considered their primary objective.

Researchers have also used blockchain technology to decentralize traditional data management systems. For instance, in [20] a blockchain-based data management solution ensures GDPR compliance and avoids security violations. An architectural blueprint for personal data management using blockchain is presented in [5]. [21] presents an IoT storage system that allows to manage and trade data generated by IoT devices securely and efficiently. [22] provides a decentralized personal data management system where user-defined access control and regulations are embedded into the blockchain using smartcontracts. For industry 4.0 applications, [23] introduces a distributed resource management framework based on smartcontract technology. Whereas a privacy preservation solution for Medical IoT applications is proposed in [24] to protect individuals' rights related to personal data. Most of these works are considered to be prototypes and lack technical analysis and implementation. The conceptual and architectural groundwork of theoretical nature has been presented but a complete implementation of a privacy preservation solution is still missing. However, the major drawback of decentralized data management and storage systems is that whenever a service provider gains access to user data they retain a copy of the user's data. This leads to privacy concerns as service providers can use the data to extract critical information or distribute it to a third party without user consent. Therefore, in the current decentralized data management systems users still have to trust their service providers to properly utilize their data when providing them with services.

We observed that a comprehensive approach to resolve privacy using blockchain technology with proper implementation and performance analysis is still missing in the literature. Therefore, in this paper we utilize existing concepts to propose a new PET solution that holistically addresses privacy by using blockchain and smartcontract technologies. It aims to cover three aspects of privacy protection: i) control over data, ii) access control and iii) auditability. A user-centric approach is adopted to ensure user control over data, allowing the user to be in charge of what data is collected, where it is stored, and how it is processed. Access control refers to the policies that the user can implement to specify by whom and when they want their data to be accessed or shared. Auditability is ensured by maintaining immutable logs for each data access so that a supervisory authority has accurate and authentic data records to investigate upon.

III. BLOCKCHAIN OVERVIEW

Blockchain technology has recently attained a lot of attention as it is used in various intuitive applications. Blockchain is a distributed database maintaining a scalable list of data records. Each block of information carries transaction data, a timestamp and an encrypted hash value of the previously linked block [25]. Linking these blocks using the hash function makes the chain of data grow in the database. The concept came up in 2008 by Satoshi Nakamoto, who devised the first blockchain database by solving the double-spending problem using the Hashcash method. Blockchain was further adopted as a technique to implement the cryptocurrency bitcoin [15]. In bitcoin, blocks serve as a public ledger that keeps all the financial transactions. Recently, blockchain has been integrated with other business domains and fields of application to enhance their security and privacy. Besides, smartcontracts deployed over blockchain promise to improve agreements between parties without third-party intervention. The following subsections will present a further overview of some terminology of blockchain.

A. WORKING OF BLOCKCHAIN

A blockchain consists of many blocks that themselves consist of hashed batches of transactions. A transaction contains information that needs to be shared securely. Transactions are grouped inside a block based on the size of the block. The block is sent over the network in a synchronized manner that allows each node to store a copy of the blockchain. The synchronization requires an agreement between nodes which is called consensus. The working of a blockchain can be summarized in the following phases: transactions creation, transaction submission and broadcast phase, block validation, block inclusion into the blockchain. Figure 2 shows stepwise how blockchain works. In the first step, an entity requests a transaction that contains useful information. The block representing the transaction is then broadcast to all nodes in a network using a Peer-to-peer protocol. The network of nodes validates the transaction with the help of a known algorithm. Once the transaction is validated a new block containing the transaction is added to the blockchain.

Once the transaction is submitted it needs to be authenticated. This is done using a pair of cryptographic public and private keys. The public key is used to create a digital identity whereas the private key is used to provide a digital signature. The digital signature provided with the transaction is used to authenticate the user [26]. However, before the transaction is added to a block it also needs to be approved or validated. This decision is made using a consensus protocol [27]. The validators investigate the blocks and check that they meet all the requirements. Once this is completed, the transaction is marked as valid, grouped, and published. The validation process is the key that allows a new block to be added to the blockchain. Once the validation is agreed by all nodes, the agreement has been reached for the block following the consensus protocol.

B. TYPES OF BLOCKCHAIN

There are four types of blockchain architecture: public, private, consortium, and hybrid. All these types have a peer-to-peer network consisting of nodes that can create and validate transactions. However, there are differences in terms

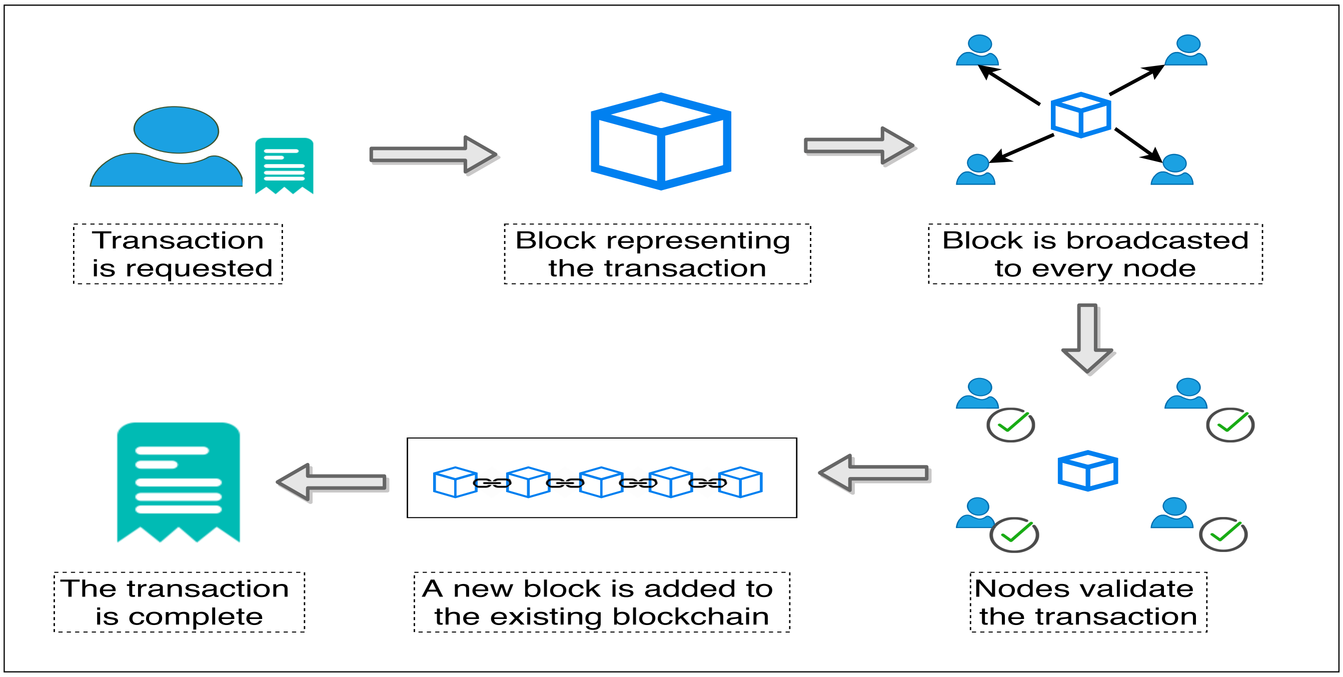

Transaction is requested
Block representing the transaction
Block is broadcasted to every node
Nodes validate the transaction
A new block is added to the existing blockchain
The transaction is complete

FIGURE 2. Working of blockchain.

of accessibility and availability between different types of blockchain.

1) PUBLIC BLOCKCHAIN

A public blockchain is an open-source platform that allows anyone to sign-in and access without any permission. All its participating nodes are given the authority to verify transactions and validate blocks. Nodes conduct a mining process to reach a consensus agreement for including a block to the blockchain. The permissionless and non-restrictive nature of public blockchain makes it suitable for cryptocurrencies such as Bitcoin [15].

2) PRIVATE BLOCKCHAIN

A private blockchain consists of a closed network owned by an entity or organization and is restricted to specific users. A new user can be added only if the owner of the private blockchain allows it. The creation of new blocks and adding transactions are restricted, and permission is only given to controlling nodes set up by the owner of the blockchain. The restrictions due to inaccessibility and authorization increase the level of security and privacy. Therefore, this characteristic of private blockchains makes them favorable for use in financial institutions and banks.

3) CONSORTIUM BLOCKCHAIN

A consortium blockchain is a community blockchain that allows more than one organization to be involved in managing a private blockchain [28]. In a consortium blockchain, the authority is shared among a group of entities, so that consortium blockchains are considered semi-decentralized. Transactions can only be verified by a set of consortium nodes. Thus, the consortium blockchain awaits has a low transaction cost associate with it and does not face scalability problems as compared to public blockchains.

4) HYBRID BLOCKCHAIN

Hybrid blockchain is a mixture of public and private blockchain and takes advantage of both characteristics simultaneously [29]. The hybrid blockchain is customizable as it allows users to decide who can participate in the blockchain and which transactions can be made public. Thus, its hybrid architecture ensures privacy by taking advantage of the restricted access of private blockchain while maintaining integrity, transparency, and security as in a public blockchain.

C. CONSENSUS AGREEMENT

Consensus is defined as a general common interest allowing participants of a multi-agent system to agree on a specific target to be accomplished. In the blockchain, consensus agreement is used to define rules and regulations that allow different nodes in the network to reach a settlement on which transaction is valid and can be included in a new block of the blockchain [30]. Various consensus protocols exist, and three have gained popularity: Proof of Work (PoW), Proof of Stake (PoS), and Proof of Authority (PoA).

1) PROOF OF WORK

The PoW protocol is used during mining to provide validation for new blocks. In this protocol, the nodes compete with each other to receive a reward by confirming that the transaction received is accurate. The process includes a hard mathematical puzzle to be solved by mining nodes. The nodes try to solve a puzzle using a trial and error method. Once a mining node finds a solution it communicates it with the rest of the nodes. A consensus is achieved when the majority of the nodes agree that the miner has solved the problem. Once consensus is achieved the transaction is added to the blockchain and the miner receives a reward in the form of transaction fees. It is impossible to fake a consensus as it requires hacking at least 50% of nodes in the network. The PoW is a very efficient protocol, however, it consumes a lot of power and time due to its high computation cost to solve a mathematical puzzle.

2) PROOF-OF-STAKE

The PoS has a different methodology to generate consensus: it uses a pseudo-random function to select a miner that is allowed to create the new block based on the node's wealth. The stake or wealth of the node determines the chances for a node to become a validator for the next block. The higher the stake of the node, the higher its chances to become a validator. PoS resolves the high computational cost of the PoW protocol. However, a node's wealth is not always appropriate for the selection since the creation of new blocks may be taken over by a single wealthy node as done in centralized systems [31]. This may result in the blocks being managed unfairly. Different methods such as randomized block selection and coin age selection are introduced so that PoS does not favor only the wealthiest nodes.

3) PROOF-OF-AUTHORITY

In PoA the validators stake their identity instead of their wealth. The person who wants to validate a transaction needs to first confirm its identity. The validation performed is linked to the identity and stored over the blockchain. This means that the validators are staking their reputation. To become an authorized validator, the validator needs to confirm its real identity. When a transaction has been validated the identity of the validator is confirmed on-chain in a certain way by an approved protocol. The identity is only staked by a small group of validators, improving the efficiency and security of the consensus agreement. PoA does not require high computational cost as PoW or a huge amount of wealth to stake as PoS. However, the PoA is only applicable to private and consortium blockchain networks.

D. WHY BLOCKCHAIN

The revolution of blockchain technology has recently reached efficient and successful integration with different applications. This success comes from fundamental features that we summarise here as follows:

- Decentralization: Blockchain distributes information to all nodes available on its network instead of keeping the information in a central entity. This makes it more secure, as no single entity can take control of the network and tamper with information.
- Immutability: Data inside a blockchain cannot be altered since there are many copies of data on different nodes, which makes tampering almost impossible. Furthermore, the use of cryptography does not allow anyone to intrude and alter the data saved on the blockchain. Any modification of data needs to be agreed upon by a majority of nodes.
- Integrity, Authenticity, and Non-Repudiation: When a transaction is made the data is guaranteed against any modifications by verifying the content against its transmitted hash. The user's private key is used to digitally sign the transaction, which assures the authenticity and non-repudiation of each message.
- Auditability and Traceability: Each transaction is recorded in a verifiable and permanent manner. In a blockchain, all operations are traceable and available to each participant. These features facilitate accountability and audit.

E. SMARTCONTRACT

We conclude the section of blockchain overview discussing smartcontracts since they are relevant to our work. A smartcontract is a method used to make, negotiate, and verify agreements electronically. The method is built by coding a computer program to allow participants to transfer anything of value under predefined conditions. Smartcontracts are considered as a replacement for the traditional contracts facilitated by third parties. It simply removes the third party by translating an agreement to a computer code that is executed over the blockchain. The code is automated and keeps track of the agreement's terms and conditions. Thus, smartcontacts are self-verifying, self-enforcing, and tamper-proof. The smartcontract is considered to be the third generation of the blockchain revolution. Blockchain has drastically gained in popularity from the smartcontract capability. Businesses that aim to simplify, speed up, and reduce their costs are leveraging smartcontracts. Fields that are currently benefiting from smartcontacts include healthcare, merchandising, real-estate, e-governance, IoT, etc.

IV. PETchain FRAMEWORK

In this section, we present PETchain: a privacy enhancing technology that utilizes blockchain and smartcontracts to ensure the privacy of users in a holistic manner. We start by describing the system actors and presenting the system model of PETchain. Next, we detail how PETchain utilizes distributed storage and a trusted executor to preserve user privacy. Finally, we describe PETchain smartcontracts and their various functions.

A. SYSTEM MODEL

The PETchain framework consists of five major entities as described below:

- **User**: An individual or organization that owns data and has the right to allow who can access and process it.
- **Service Provider**: An online public or private organization that is established to deliver various applications to their consumers. They provide services and features to their subscribers by acquiring and processing their data.
- **Distributed Storage**: A peer-to-peer distributed open-source file-sharing system that allows data to be

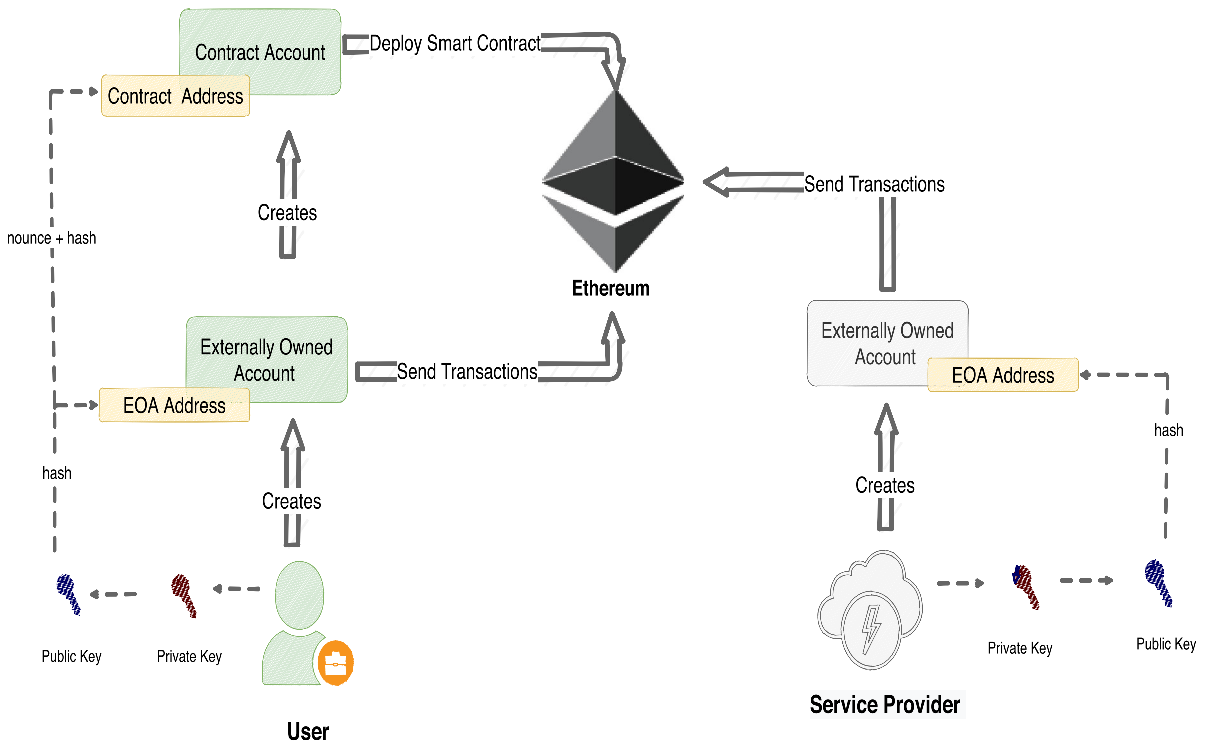


FIGURE 3. Ethereum accounts and addresses.

stored, maintained, and distributed in a fast and secure manner.

- **Trusted Executor**: A trusted individual or organization that provides a secure isolated environment that allows code to be executed over some data while guaranteeing the confidentiality and integrity of the data as well as the code there loaded.
- **Blockchain**: A peer-to-peer distributed platform that supports decentralized applications. It should supports smartcontract code execution and storage over the distributed network such as Ethereum.

In the PETchain framework, all users and service providers are obliged to register themselves to the Ethereum network. Figure 3 shows two types of accounts and addresses over the Ethereum blockchain. As shown in the figure, each entity registers an externally owned account (EOA) represented by an EOA address. Users also create a contract account to deploy their smartcontract having a unique contract address. PETchain exploits several encryption keys presented in Table 1. Each entity operating over Ethereum generates a pair of public key P_u^{eth} and private key P_r^{eth}. The P_u^{eth} is used to generate a unique 20-byte EOA account address that identifies it over Ethereum blockchain. Each message sent by the user over the PETchain platform is digitally signed by their P_r^{eth} for maintaining the authenticity and integrity of each message. K_D is a symmetric key generated by the user to encrypt data before uploading it over the distributed storage. The uploaded data is uniquely identified by its data identifier. The user also shares a secret key K_{TE} with the trusted executor. Users further encrypt K_D using K_{TE} to produce K_D^E and store it over the smartcontract along with the data identifier.

TABLE 1.

Symbol	Representation
P_u^{eth}	Private key for ethereum
P_r^{eth}	Public key for ethereum
K_D	Symmetric key to encrypt data
K_{TE}	Symmetric key shared between user and trusted executor
K_D^E	Encrypted K_D using K_{TE}

The system model of PETchain is presented in Figure 4. There are two kinds of messages over PETchain: API calls, and contract transactions. The API messages are used by entities to interact with each other, whereas contract transactions are sent to the smartcontract over the Ethereum network. To completely understand how the PETchain system model works we describe the several steps involved:

1) *Publish smartcontract:* The user defines its access control over its smartcontract. The smartcontract contains the list of service providers authorized to utilize user data.
2) *Store data:* The user encrypts its data using E_D and uploads it over the distributed storage system.
3) *Update data identifier:* The user updates the data identifier of the uploaded data over its smartcontract.
4) *Request data:* The service provider requests data by calling a function of smartcontract. If the service provider belongs to the user's authorized list it will be returned with the data identifier and K_D^E. It also receives the URL of the user's trusted executor.

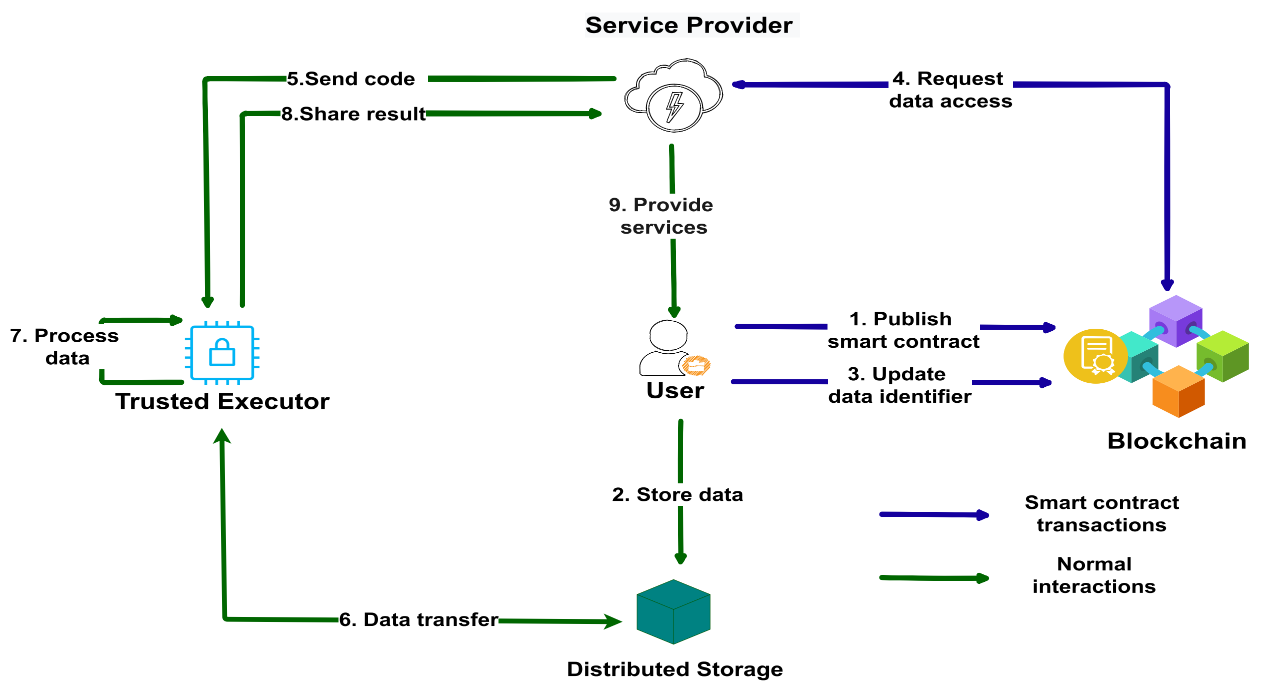


FIGURE 4. PETchain system model.

5) *Send code:* The service provider sends the code to be executed over user data and the data identifier to the trusted executor.
6) *Data transfer:* The trusted executor uses the data identifier to download the data from the distributed storage.
7) *Process data:* The data is processed in an isolated secure environment using the code received from the service provider. Trusted executor decrypts K_D^E to get K_D, which is then used to decrypt the downloaded data before executing the code.
8) *Share data:* The results of the processed data are shared with the service provider.
9) *Provide services:* The service provider can use the results to provide its services to the user.

PETchain supports a user-centric architecture that aims to give complete control to the user who owns the data. Users can maintain a list of authorized service providers (EOA addresses) over its smartcontract that are allowed to utilize their data. PETchain ensures the security of the data by allowing users to store the encrypted versions of their data on the distributed storage. Only the trusted executors will be able to decrypt and process user data. This is done in a secure isolated environment to ensure user privacy. Thus, PETchain allows service providers to provide users with services without obtaining their data. Moreover, with the use of blockchain PETchain keeps the logs of all data access in an auditable and immutable manner.

B. InterPlanetary FILE SYSTEM

PETchain uses a distributed data storage system instead of centralized cloud storage. In distributed data storage there is no single point of failure as data is scattered and stored across different peers. Moreover, faster access to data is achieved as peers can directly share content, unlike centralized storage which has high bandwidth requirements. PETchain uses InterPlanetary File System (IPFS)[2] as its distributed data storage system. IPFS defines a peer-to-peer protocol that allows data to be accessible through distributed peers [32]. IPFS is considered to be faster, secure, and reliable compared to cloud storage services. It also aims to resolve the data redundancy issue of the web. This is mainly because IPFS uses content addressing rather than location addressing to identify data stored over the network. The IPFS uses the hash of the data file as its identifier, which always remains unique to the data. A newer version of data is uploaded in case the data is updated or changed. If a particular data file is requested then the data identifier is used to locate the file across different peers. A node on IPFS whose stored hash matches with the content identifier will return the data file to the requester.

In PETchain, the user encrypts and digitally signs the data before uploading it over the IPFS network. Figure 5 illustrates the different steps a data file goes through before being uploaded over the IPFS. The user starts by selecting the data file required to be uploaded. The data file is then encrypted using a symmetric encryption algorithm such as AES and symmetric key K_d. Thus, K_d is required by anyone who wants to access and process the data. In the following step, the data file is digitally signed by the user. The user uses the private key P_r^{eth} of thee Ethereum's EOA to sign the data file. The public key P_u^{eth} can then be used to verify that the data file belongs to the same user that is registered on Ethereum and owns the smartcontract. The hash of the digitally signed data

[2]https://ipfs.io/

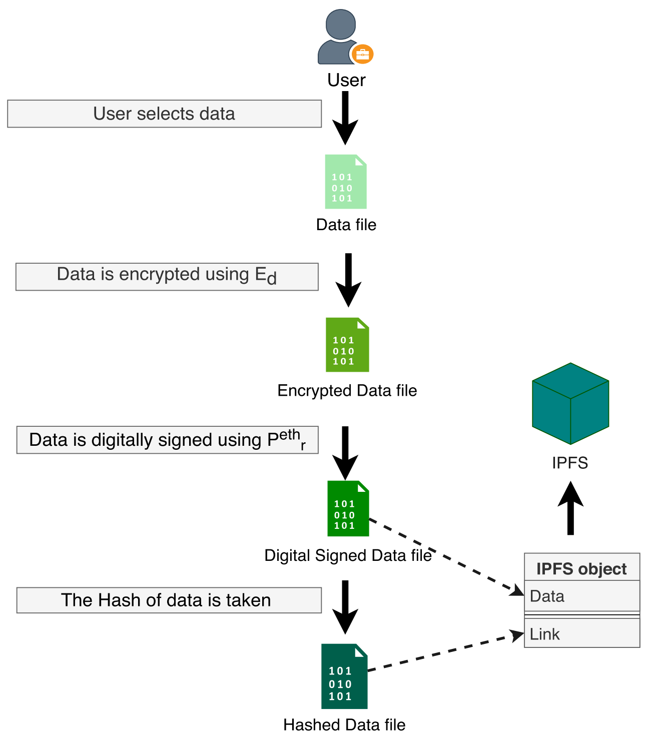


FIGURE 5. Data storage on IPFS.

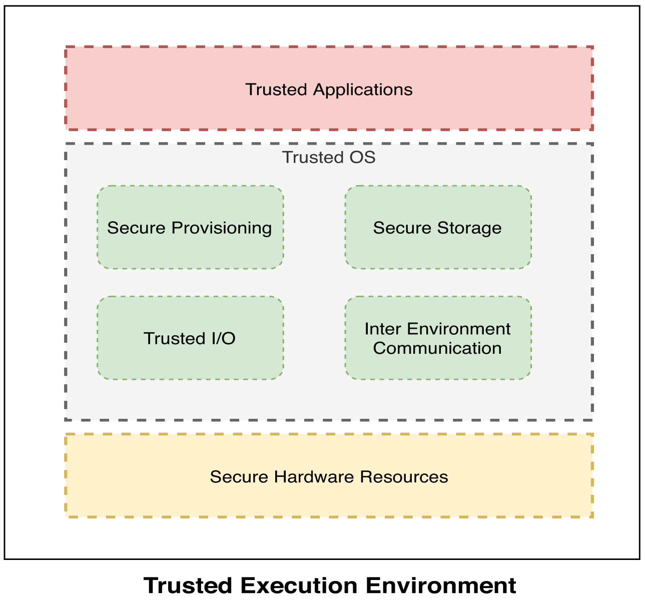

Trusted Applications
Trusted OS
Secure Provisioning
Secure Storage
Trusted I/O
Inter Environment Communication
Secure Hardware Resources
Trusted Execution Environment

FIGURE 6. Basic building blocks of TEE.

Algorithm 1: set_identifier(*data_id*, *SP_address*, *data_key*, *TE_url*)

```
if $msg.sender == owner$ then
    Map $SP\_address$ to $data\_id$; Map $SP\_address$ to $data\_key$; Map $SP\_address$ to $TE\_url$; return true;
else
    return false;
end
```

file is taken using an SHA 256 algorithm. The hash acts as the data identifier. The user then uploads the data over IPFS in the form of an IPFS object that consists of the data field and link field. The data field consists of a data file whereas the link consists of the data identifier. IPFS data field can only accommodate 256 Kb of data. Therefore, if the data file is larger than 256 KB, it is broken down into several IPFS objects and stored over different peers in the network. The main object consists of the links to each IPFS object which can be used to locate other objects and recover the data file.

C. TRUSTED EXECUTION ENVIRONMENT

In PETchain, a trusted execution environment (TEE) is used to process user's data. Users can implement their TEE or use a third party trusted executor. A TEE is a processing environment that allows the authenticity, integrity, and confidentiality of its data and code. The basic building blocks of TEE are presented in Figure 6. TEE allows service providers to process the user's encrypted data in a secure environment without having access to the data. The trusted executor has access to both user's data and the service provider's code. However, the architecture of TEE makes it unfeasible for anyone to have a copy of the data. Once the code is executed over the data the service provider is informed about the results of the execution. In PETchain, we choose GrapheneOS [3] which is an open-source, security-hardened TEE. GrapheneOS implemented over Intel SGX hardware provides adequate performance. The latency on Graphene-SGX for opening a 64 KB file is 383μs, reading a 64KB file takes 0.5μs whereas forking a 16MB process takes 0.8s [33].

[3]https://grapheneos.org/

D. PETchain SMARTCONTRACT

We describe now the PETchain smartcontract and its functions. The PETchain smartcontract is written and compiled in solidity. The contract is deployed and owned by the user. When the contract is deployed the user's EOA address is recorded as the owner of the smartcontract. The smartcontract is then used to store and retrieve data identifiers. The variables used in the smartcontract to store information include i) *data_id*: contains data identifier, ii) *data_key*: contains encrypted key for the data, iii) *SP_address*: contains the address of the service provider, iv) *TE_url*: contains the URL of the trusted executor, and v) *Status*: indicates whether the service provider is trusted or not. We use mapping value type in solidity to relate and store information due to its lower computational cost compared to arrays [34]. The mapping stores information in a virtual hash table having each potential key mapped to a value.

The PETchain consists of five functions. The algorithms of these functions are presented in Algorithms 1-5 and explained as follows:

1) **set_identifier**: This function allows the user to upload the data identifiers securely over the smartcontract.

Algorithm 2: set_authorization(*SP_address*, *status)*

```
if msg.sender == owner then
    map status to SP_address;
    return true;
else
    return false;
end
```

Algorithm 3: get_identifier*()*

```
if Status of msg.sender is trusted then
    get data_identifier, get data_key and TE_url using
      msg.sender address
    returndata_id, data_key, TE_url;
else
    return false;
end
```

Algorithm 4: destroy_smartcontract()

```
if msg.sender == owner then
    destroy owner's smartcontract
    return true;
else
    return false;
end
```

Algorithm 5: pause_smartcontract()

```
if msg.sender == owner then
    Pause owner's smartcontract
    return true;
else
    return false;
end
```

The EOA address of the service provider is sent as an argument of the function along with the data identifier, encrypted key, and URL of the trusted executor. The address is mapped to the data identifier, encrypted key, and URL of the trusted executor to be stored over the smartcontract. The function allows only the owner of smartcontract to upload the data identifier. The same function can be used to change the data identifier if the user's data is updated.

2) **set_authorization**: This function allows the user to maintain a list of trusted service providers. Only the owner of the smartcontract can call this function. Its arguments require the EOA address of the service provider and the status, indicating whether it is trusted or untrusted. This function can also be used to change the status of the service provider.
3) **get_identifier**: This function is used by the service provider to retrieve data identifier. When the service provider calls it, its EOA address is recorded. Using the address, a check is performed whether the service provider belongs to the list of trusted service providers. If the service provider is listed as trusted, the function will return the corresponding data identifier, encrypted key, and URL of the trusted executor. If the address is indicated as untrusted or not listed, the requester will be notified that it does not have the authority to access the user's data identifier.
4) **destroy_smartcontract**: the owner is able to destroy the smartcontract. Once the smartcontract is destroyed its functions can never be called again.
5) **pause_smartcontract**: the owner is able to pause and unpause the contract.

A sample code of PETchain **set_identifier** and **get_identifier** function are provided as follows.

```
function set_identifier(bytes32 _dataid, address
   _address,  bytes32 _key, string memory _url)
  public
      {
      require (msg.sender == owner,"You are not
  the owner");
        keymapping[_address]=_key;
       urlmapping[_address]=_url;
       addressmapping[_address]=_dataid;
      }

```

```
      function get_identifier() public view
  returns(bytes32, bytes32, string memory)
      {
      require(paused == false,"contract is
  paused by owner");
      require (msg.sender == owner || keccak256(
  bytes(authmapping[msg.sender])) == keccak256("
  trusted"),"policy doesnot allow to access");
      return(addressmapping[msg.sender],
  keymapping[msg.sender],urlmapping[msg.sender])
  ;
      }
```

V. IMPLEMENTATION

In this section, we describe how we implemented PETchain on a consortium blockchain to analyze its feasibility and performance. A consortium blockchain is more suitable in terms of ensuring privacy bcause the information contained in the blocks is only visible to the members of the consortium and not to the entire public. To implement our solution we deployed an Ethereum PoA blockchain consisting of five nodes and executed PETchain smartcontract over it. Currently, Ethereum allows two consensus protocols: PoW and PoA. As we choose to implement PETchain over a consortium blockchain we selected the PoA consensus. We performed various experiments to select the appropriate configuration to achieve the best possible performance. The three critical parameters for this PoA blockchain are:

1) Sealers: Sealers are nodes that have the authority to validate a transaction and include it on a block. A block can only be added to the blockchain if it is validated by at least 51% of the sealers present in the network. Initially, sealers are configured in the blockchain genesis block. New sealers can be added anytime if 51% of the sealers present in the network allow it.

2) Block-time: Block-time is the time between two consecutive blocks. It is configured inside the blockchain genesis block. After each interval of block-time, a new block is added to the blockchain. However, the actual block-time always varies from the configured block time due to various delays caused by the network during synchronization.
3) Block-gas-limit: Block-gas-limit is the maximum amount of gas that can be collected from transactions in each block. It is set in *wei* where one *wei* is equivalent to $10^{-}18$ ETH. The gas limit determines the size of the block and the number of transactions that can fit inside a block, which is dependent on the GAS cost of each transaction.

A. EXPERIMENTAL SETUP

In our experimental setup, we used the AWS cloud service to deploy five EC2 virtual machines. We ran Ubuntu Server 18.04 LTS (HVM) having 1 GB RAM and 20 GB storage in each virtual machine. The virtual machines were configured to run a Go Ethereum (geth[4]) client to initialize Ethereum nodes. For each node, at least one externally owned account was created using a public-private key pair. The public key is further used to generate a 20-byte public address. For example Alice creates an externally owned account and has a public address of “$0 \times 1f4993692CDaD87DEc7227650B7aA557EcE6E5b1$”. She can further use this address to deploy her smartcontract and sends transactions.

The complete code of PETchain smartcontract is available in the github repository.[5]

To generate a genesis block for our blockchain we used Puppeth.[6] The genesis consists of a simple JSON file that contains the configuration parameters and thresholds. Listing 1 shows a sample genesis file that consists of all necessary parameters required to set up a PoA blockchain. The algorithm used for PoA consensus in Ethereum is Clique. The *chainid* for a Clique network can be any number other than 1,2,3,4,42 and 62 which are associated with the main and test networks. The *difficulty* field is set to zero in clique as no mining is required. The *gasLimit* field is used to set the block-gas-limit whereas the *Epoch Period* is for configuring the block-time. The *extraData* field is used to add the addresses of accounts that are selected as sealers at blockchain initialization.

We used geth to initialize a node in each virtual machine. Each node is configured with the same genesis block. To initialize the blockchain the enode address of nodes are provided. Enode address is a URL that consists of the 512 bit public key and the IP address of nodes. This allows nodes to discover their peers. After initializing the blockchain we

```
{
  "config": {
    "chainId": 2006,
    "clique": {
      "period": 5,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x5f195492",
  "extraData": "da84a025a783dd6c97
  df6208f4e33ece781847fb",
  "gasLimit": "0x47b760",
  "difficulty": "0x0",
  "coinbase": "0x0000000000000000
  0000000000000000000000",
  "alloc": {
    "da84a025a783dd6c97df6208f4e33
    ece781847fb": {
      "balance": "0x200000000000000
      00000000000000000000000000000
      000000000000000000000"
    }
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x000000000000000
  00000000000000000000000000000000
  0000000000000000"
}
```

Listing 1. JSON genesis.

used Remix [7] and Metamask [8] wallet to deploy PETchain contracts. The EOA generated using geth were imported to the Metamask to deploy smartcontract. The remix is linked to Metamask using injected web3 provider whereas the Metamask wallet is connected to our clique network using RPC URL.

B. PERFORMANCE EVALUATION

The performance of PETchain over Ethereum blockchain is analyzed using the following metrics:

1) **Transaction GAS Cost (*TGS*):** The GAS required to execute a smartcontract transaction in the network.
2) **Transaction Per Second (*TPS*):** The number of transactions that can be executed in the network in each second.
3) **Lost blocks:** The number of blocks lost due to the delay caused in broadcasting the signed block by the sealer.

[4]https://github.com/ethereum/go-ethereum/wiki/geth
[5]https://github.com/ibrahimtariqjaved/PETchain
[6]https://github.com/puppeth?q=&type=&language

[7]https://remix.ethereum.org/
[8]https://metamask.io/ wallet

TABLE 2. Transaction GAS cost for PETchain.

No	Contract Transaction	TGS
1	**PETchain deployment**	1296004
2	**set_identifier**	90819
3	**set_authorization**	46655
4	**get_identifier**	30299
5	**pause_smartcontract**	28508
6	**destroy_smartcontract**	14134

4) **Propagation Delay:** The amount of time it takes to propagate a block to the entire network.

In the PoA network, users do not have to pay miners the cost of executing their transactions. However, *TGS* remains a good metric to check the computational efficiency of a smartcontract. Lower *TGS* implies that less computational power is required for execution which enhances the overall throughput of the network. We computed *TGS* required for the completion of different transactions over PETchain. This includes the deployment of smartcontract and executing functions such as set_identifier, set_authorization, get_identifier, pause_smartcontract, destroy_smartcontract. We made efforts to keep the GAS cost as low as possible. We were able to decrease the GAS cost substantially by using mapping instead of arrays to store information [34]. Table 2 enumerates the amount of GAS required to complete different transactions over the blockchain. We can observe that a user requires approximately 1300, 000 GAS to deploy a PETchain smartcontract over the Ethereum PoA network. This cost is required only once when the smartcontract is deployed. On the other hand, the functions require very little GAS. This shows the computational efficiency of our smartcontract. The set_identifier function requires the highest GAS, around 90, 000, as it maps three different values including data_key, data_id, and TE_URL to SP_address as presented in Algorithm 1.

The public Ethereum network supports $12-15$ *TPS* [35]. However, with the consortium network we were able to achieve a much higher *TPS*. We ran our blockchain for one hour to compute *TPS* using the following equation:

$$TPS = \frac{Gas\ Limit}{TGS * Block - time}$$

where block-gas-limit and block-time are configured in the genesis file. However, the actual block-time varies due to network delays and synchronization issues. Therefore, we measured the average block-time by running the blockchain for one hour. The measured vs. configured block-time is presented in Figure 7. The measured block-time is then used to compute the *TPS* for three different block-gas-limits (400000, 1000000and 2000000) as shown in Figure 8. It can be observed that high *TPS* is achieved with increased block-gas-limit. This is because more transactions can be accommodated into a block by having a high block-gas-limit. On the other hand, the *TPS* decreases rapidly by increasing the block-time. To achieve a high *TPS* a low block-time and high block-gas-limit can be selected. However, selecting an appropriate block-time is crucial as it causes delays in the network as seen in further experiments.

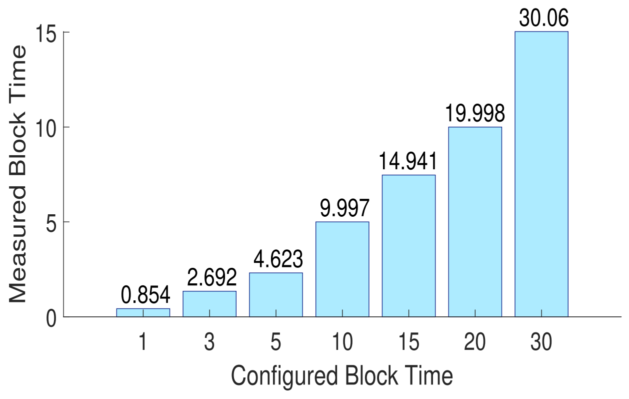
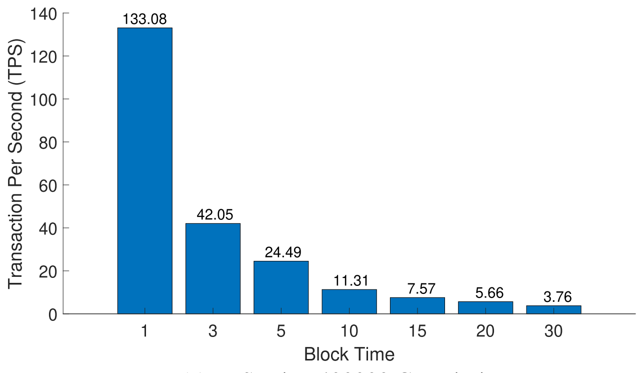
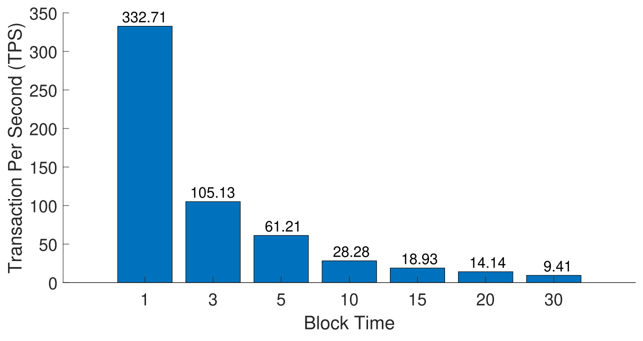


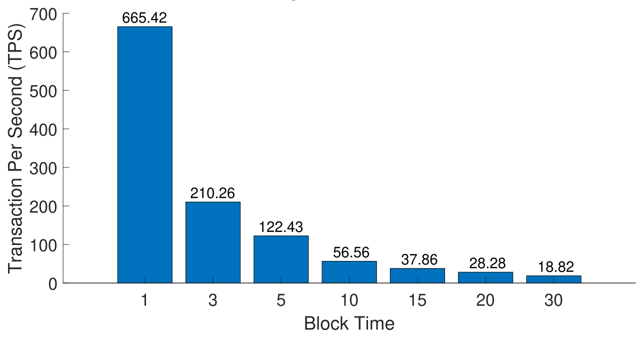
Figure 7. Measured vs. Configured block time.



(a) TPS using 400000 Gas Limit



(b) TPS using 10000000 Gas Limit



(c) TPS using 20000000 Gas Limit

Figure 8. Transaction Per Second (TPS) as a function of block-time.

When a sealer delays its signed block in broadcasting, the backup sealers propose a new block. This causes the block to be lost. The number of lost blocks causes a delay in adding

TABLE 3. Lost blocks in terms of a) Blocktime and b) Sealers.

Block-Time	Lost Blocks
1	845
3	269
5	45
10	17
15	23
20s	3
30s	10

(a) Block-time

Sealers	Lost Blocks
1	0
2	0
3	45
4	32
5	67

(b) Sealers

TABLE 4. Propagation time in terms of a) Blocktime and b) Sealers.

Block-Time	Propagation Time (μ sec)
1	327.84
3	352.54
5	392.74
10	254.32
15	321.1
20	271.2
30	302.28

(a) Effect of Block-time on Propagation Time

Sealers	Propagation Time (μ sec)
1	250.86
2	280.64
3	392.74
4	515.98
5	632.74

(b) Lost Blocks

new blocks to the blockchain, that affects the overall latency of the network. Therefore, we tried to observe the amount of lost blocks in a one hour period concerning block-time and sealers in the network. Table 3a shows the effect of block-time on lost blocks. We observed that lost blocks have a direct relationship with the block-time. A sharp decrease in lost blocks is observed when the block-time is increased. This is because the sealer gets more time to receive, validate, and sign the transactions inside a block. Therefore, for PETchain it is appropriate to select a block-time between $5-10$ to reduce the number of lost blocks. Moreover, Table 3b shows the effect of the number of sealers over lost blocks. We observed 0 lost blocks for 1 and 2 sealers. However, as the number of sealers increases the number of lost blocks also increases. This is due to the synchronization issues between sealers, as at-least 51% sealers need to verify the block before it can be part of the blockchain. Therefore we recommend not to use more than half of the nodes as sealers in the network.

We further calculated the propagation delay in the network. This represents the time required for a block to be available to each node in the network. We present the effect of block-time and sealer to propagation delay in Figure 4. From Table 4a we observe that the propagation delay is strongly dependent on the number of sealers. More synchronization issues were observed when the number of sealers were added to the network, and this was the reason behind higher propagation delay. The propagation delay in the 5 node network having 1 sealer was observed to be 250.86. By increasing the sealers to 5 the propagation delay increased to 632.74. However, we could not observe any relationship between block-time and propagation delay as seen in 4b.

The performance analysis allowed us to characterize PETchain at different configuration settings. We used different amounts of sealers, block-time, and block-gas-limit to see the effect on our performance metrics. We observed that high *TPS* can be simply be achieved by lowering the block-time. However, block-time was strongly related to the number of lost blocks, which affects the overall performance of the network. Therefore, we propose to use a block-time of around 10 seconds. This allows to achieve a high *TPS* and minimize the delays in the network caused by lost blocks. We were able to achieve a *TPS* of around 60 for a gas limit of 20000000. This is almost four times higher than that of the public Ethereum network [35]. To minimize the propagation delay we propose to use only half of the nodes as sealers. Thus, in a 5 node network no more than $2-3$ sealers should be used.

At last we present a comparison of existing Privacy enhancing techniques with PETchain as shown in Table 5. The comparison is conducted concerning features including, data utility, accuracy, complexity, and overheads. Anonymization, differential privacy, and data summarization ensure privacy by reducing the originality of the data. Therefore, these techniques damage the utility and usefulness of the data. Moreover, they cannot guarantee the accuracy of results of data analytic applied. On the other hand, homomorphic encryption, decentralized learning, and PETchain maintain data utility and accuracy of results. However, homomorphic encryption techniques are fairly complex to implement and present a very high computational overhead since operations

TABLE 5. Comparison of privacy enhancing technologies.

Privacy Enhancing Technologies (PET)	Damage to Data Utility	Accuracy of Results	Very complex to apply	High computational overhead	Overburdens user device
Anonymization	Yes	No	No	No	No
Differential Privacy	Yes	No	No	No	No
Data summarization	Yes	No	No	No	No
Homomorphic Encryption	No	Yes	Yes	Yes	No
Decentralized Learning	No	No	No	Yes	Yes
PETchain	No	Yes	No	No	No

are conducted over encrypted data. Decentralized learning also presents high computational overhead as machine learning algorithms are applied on individual devices in a distributed manner before the results can be used by the system model. Decentralized learning also overburdens user devices as computations are offloaded and conducted over the user device. On the contrary, PETchain neither overburdens user device nor has a high computational overhead. However, proper experimental analysis is required to compare each PET in terms of actual latency observed when a particular operation is applied over data.

VI. CONCLUSION

In this paper, we proposed a novel privacy enhancing technology based on blockchain to manage and exploit user data. The proposed technique aims to address user privacy holistically. In our approach, users can define their access control policy by deploying their smartcontract. Users upload encrypted data to IPFS and store its hash to their smartcontract. The authorized service providers can access the hash but can only decrypt and process the data in an isolated execution environment. This allows service providers to process user data without acquiring nor misusing the data. To the best of our knowledge, this is the first-ever implementation of PET using blockchain technology that holistically addresses privacy. In our approach, the users select a trusted execution environment where their data can be processed in a privacy enabled manner. Moreover, they can store their data independently in a distributed manner using IPFS. The proposed solution has been implemented on a consortium blockchain due to its privacy, cost, and performance advantages over the public blockchain. The performance of PETchain is analyzed using the Ethereum platform. It was observed that a high TPS can be achieved by having low block-time and a high gas-limit. However, lower block-time was observed to have a high amount of lost blocks, decreasing the performance of the blockchain. Therefore, a bock-time of 10 is proposed that allows a TPS of around 60. Moreover, a lower number of sealers is recommended, to reduce propagation delay in the network. For our future work, we aim to analyze and improve PETchain by checking its compatibility with GDPR.

ACKNOWLEDGMENT

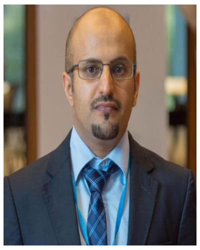
The authors would like to acknowledge Shaqra University, Saudi Arabia, for their support in this work.

REFERENCES

[1] S.-C. Cha, T.-Y. Hsu, Y. Xiang, and K.-H. Yeh, "Privacy enhancing technologies in the Internet of Things: Perspectives and challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2159–2187, Apr. 2019.
[2] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR)," in *A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.
[3] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2014, pp. 1–9.
[4] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, Oct. 2002, doi: 10.1142/S0218488502001648.
[5] *Privacy by Blockchain Design: A Standardised Model for Processing Personal Data Using Blockchain Technology*, Standard DIN ISO 4997:2020-04, 2020. [Online]. Available: https://www.beuth.de/de/technische-regel/din-spec-4997/321277504
[6] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.
[7] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, Mar. 2006, p. 24.
[8] N. Li, T. Li, and S. Venkatasubramanian, "T-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 106–115.
[9] A.-E.-E.-A. Hussien, N. Hamza, and H. A. Hefny, "Attacks on anonymization-based privacy-preserving: A survey for data mining and data publishing," *J. Inf. Secur.*, vol. 4, no. 2, pp. 101–112, 2013.
[10] B. Jeon, S. M. Ferdous, M. R. Rahman, and A. Walid, "Privacy-preserving decentralized aggregation for federated learning," 2020, *arXiv:2012.07183*. [Online]. Available: https://arxiv.org/abs/2012.07183
[11] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[12] M. U. Hassan, M. H. Rehmani, and J. Chen, "Differential privacy in blockchain technology: A futuristic approach," *J. Parallel Distrib. Comput.*, vol. 145, pp. 50–74, Nov. 2019.
[13] E. ElSalamouny and S. Gambs, "Differential privacy models for location-based services," *Trans. Data Privacy*, vol. 9, no. 1, pp. 15–48, Apr. 2016.
[14] B. Mirzasoleiman, M. Zadimoghaddam, and A. Karbasi, "Fast distributed submodular cover: Public-private data summarization," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, in Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates, 2016, pp. 3601–3609.
[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper, 2008, vol. 4. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[16] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "FairAccess: A new blockchain-based access control framework for the Internet of Things: FairAccess: A new access control framework for IoT," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5943–5964, Dec. 2016.
[17] S. Wang, X. Wang, and Y. Zhang, "A secure cloud storage framework with access control based on blockchain," *IEEE Access*, vol. 7, pp. 112713–112725, 2019.
[18] M. U. Rahman, B. Guidi, F. Baiardi, and L. Ricci, "Context-aware and dynamic role-based access control using blockchain," in *Advanced Information Networking and Applications*, L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds. Cham, Switzerland: Springer, 2020, pp. 1449–1460.
[19] M. U. Rahman, F. Baiardi, B. Guidi, and L. Ricci, "Protecting personal data using smart contracts," in *Internet and Distributed Computing Systems*, R. Montella, A. Ciaramella, G. Fortino, A. Guerrieri, and A. Liotta, Eds. Cham, Switzerland: Springer, 2019, pp. 21–32.
[20] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, "GDPR-compliant personal data management: A blockchain-based solution," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1746–1761, 2020.
[21] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep. 2019.
[22] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, May 2015, pp. 180–184.
[23] A. Lahbib, K. Toumi, A. Laouiti, and S. Martin, "DRMF: A distributed resource management framework for industry 4.0 environments," in *Proc. IEEE 18th Int. Symp. Netw. Comput. Appl. (NCA)*, Sep. 2019, pp. 1–9.
[24] B. Alamri, I. T. Javed, and T. Margaria, "Preserving patients' privacy in medical IoT using blockchain," in *Edge Computing*. New York, NY, USA: Springer, 2020, pp. 103–110.
[25] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3796–3838, Jul. 2019.
[26] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 858–880, 1st Quart., 2019.
[27] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2017, pp. 243–252.

[28] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, and B. Hamida, "Consortium blockchains: Overview, applications and challenges," *Int. J. Adv. Telecommun.*, vol. 11, nos. 1–2, pp. 51–64, 2018.
[29] D Team. *Blockchains Tutorials*. Accessed: Aug. 10, 2020. [Online]. Available: https://data-flair.training/blogs/types-of-blockchain/
[30] A. Ellervee, R. Matulevi, and N. Mayer, "A comprehensive reference model for blockchain-based distributed ledger technology," in *Proc. CEUR Workshop*, 1979, pp. 320–333.
[31] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, in Computer and Communications Security. New York, NY, USA: Association for Computing Machinery, 2016, pp. 3–16, doi: 10.1145/2976749.2978341.
[32] J. Benet, "IPFS-content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*. [Online]. Available: https://arXiv.org/abs/1407.3561
[33] C.-C. Tsai, D. E. Porter, and M. Vij, "Graphene-SGX: A practical library OS for unmodified applications on SGX," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, Santa Clara, CA, USA, Jul. 2017, pp. 645–658. [Online]. Available: https://www.usenix.org/conference/atc17/technical-sessions/presentation
[34] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, Apr. 2014.
[35] *The Ethereum Blockchain Explorer*. Accessed: Feb. 1, 2021. [Online]. Available: https://etherscan.io/

IBRAHIM TARIQ JAVED received the Ph.D. degree in computer science and telecommunication from the Institut Mines-Telecom, Telecom SudParis, France. He is currently working as a Postdoctoral Researcher with the Lero—Science Foundation Ireland Research Centre for Software, University of Limerick, Ireland. He is also an Assistant Professor with the Department of Computer Science, Bahria University, Pakistan. He has several years of research and teaching experience at various academic and research institutes. His research interests include blockchain, privacy, identity and trust management, peer-to-peer communication, and nano networks.



FARES ALHARBI received the Ph.D. degree in computer systems engineering from the Queensland University of Technology, Brisbane, QLD, Australia, in 2019. He is currently an Assistant Professor with Shaqra University, Riyadh, Saudi Arabia. His research interests include network management, cloud computing, data center optimization, data management, and resource scheduling of distributed systems, and identity management and privacy.

TIZIANA MARGARIA is currently the Chair of Software Engineering and the Head of the Department of Computer Science and Information Systems, University of Limerick. She also heads the Lero Committee on International Relations Development. In Lero, she heads research projects on scientific workflows, in particular for data analytics, on model-driven service-oriented software design for evolving systems, and holistic HW/SW cyber-security. She is a Fellow of the Irish Computer Society and the Society for Design and Process Science (SDPS). She is a Vice President of the European Association of Software Science and Technology (EASST), a Past President of the ERCIM Working Group on Formal Methods for Industrial Critical Systems (FMICS), a Steering Committee Member of ETAPS and the European Joint Conferences on Theory and Practice of Software, a Managing Editor of STTT, the *International Journal on Software Tools for Technology Transfer* (Springer), and a Co-Founder of TACAS and ISoLA series of conferences. In the European Society for Emergency Medicine (EuSEM), she co-chairs the Special Interest Group on Technology and Processes of Care in the Emergency Care (SIG-TPCEC).

NOEL CRESPI received the master's degree from the University of Orsay (Paris 11) and the University of Kent, U.K., the Diplome d'ingenieur degree from Telecom ParisTech, and the Ph.D. and Habilitation degrees from Paris VI University (Paris-Sorbonne). Since 1993, he has been with CLIP, Bouygues Telecom, and then at Orange Labs, in 1995. He took leading roles in the creation of new services with the successful conception and launch of Orange prepaid service, and in standardization (from rapporteur ship of IN standard to coordination of all mobile standards activities for Orange). In 1999, he joined Nortel Networks as a Telephony Program Manager, architecting core network products for EMEA region. In 2002, he joined the Institut Mines-Telecom, where he is currently a Professor and the Program Director leading the Service Architecture Laboratory. He coordinates the standardization activities for Institut Mines-Telecom at ITU-T, ETSI, and 3GPP. He is also an Adjunct Professor with KAIST, an Affiliate Professor with Concordia University, and a Guest Researcher with the University of Goettingen. He is the Scientific Director of the French-Korean Laboratory ILLUMINE. His current research interests include service architectures, softwarization, social networks, and the Internet of Things/services.

KASHIF NASEER QURESHI received the Ph.D. degree in computer communications (networks) from Universiti Teknologi Malaysia, in 2016. He is currently working as an Associate Professor with the Department of Computer Science, Bahria University, Islamabad. He is an Experienced Cybersecurity and Information Technology Researcher with more than ten years of experience at leading academic institutes in Malaysia and Pakistan. He has worked as a Key Researcher on several projects sponsored by the Ministry of Education Malaysia (MOE) and the Higher Education Commission Pakistan. His current research interests include information security, network, and wireless communication, and the Internet of Things.