IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 14, 2022, accepted May 3, 2022, date of publication May 5, 2022, date of current version May 12, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3172973

Enabling Trust and Privacy-Preserving e-KYC System Using Blockchain

SOMCHART FUGKEAW, **(Member, IEEE)**
Sirindhorn International Institute of Technology, Thammasat University, Bangkok 12000, Thailand
e-mail: somchart@siit.tu.ac.th

This work was supported by the Sirindhorn International Institute of Technology (SIIT) Young Researcher Grant under Contract SIIT2019-YRG-SF02.

ABSTRACT The electronic know your customer (e-KYC) is a system for the banking or identity provider to establish a customer identity data verification process between relying parties. Due to the efficient resource consumption and the high degree of accessibility and availability of cloud computing, most banks implement their e-KYC system on the cloud. Essentially, the security and privacy of e-KYC related documents stored in the cloud becomes the crucial issue. Existing e-KYC platforms generally rely on strong authentication and apply traditional encryption to support their security and privacy requirement. In this model, the KYC system owner encrypts the file with their host's key and uploads it to the cloud. This method induces encryption dependency and communication and key management overheads. In this paper, we introduce a novel blockchain-based e-KYC scheme called e-KYC TrustBlock based on the ciphertext policy attribute-based encryption (CP-ABE) method binding with the client consent enforcement to deliver trust, security and privacy compliance. In addition, we introduce attribute-based encryption to enable the privacy preserving and fine-grained access of sensitive transactions stored in the blockchain. Finally, we conduct experiments to show that our system is efficient and scalable in practice.

INDEX TERMS e-KYC, authentication, CP-ABE, key management, access control, blockchain.

I. INTRODUCTION

Electronic-Know Your customer (e-KYC) is a service that banks or financial institutions (FIs) provide virtual banking operation related to authentication and verification of identity electronically to their customers for improving cost efficiency and customer satisfaction. The e-KYC system enables FIs to electronically verify their customer identity and retrieve KYC data for both individual and corporate clients. To implement the e-KYC system, financial institutions either employ off-the-shelf e-KYC software fully equipped with necessary functions or develop their own. Then, they can deploy the system as an on-premise or a cloud-based model. Due to the trend of the outsourcing model, most enterprises have adopted the cloud as the preferred platform for housing their system and data.

A cloud-based e-KYC system provides a more efficient and flexible authentication method compared to the host-based e-KYC authentication method where documents need to be validated via the centralized host. This causes a traffic bottleneck and single point of failure problem. Also, the traceability of the verified transaction is limited since all transactions occurring in the system are entirely managed by the provider. Nevertheless, the security and privacy issue of a cloud-based solution is a concern for many potential enterprises. This is because e-KYC system located on the cloud store customer data documents and it might be viewed by any public cloud tenants or even the cloud service providers (CSPs). To address this concern, most banks and FIs need to implement an encryption mechanism in addition to the strong authentication feature provided by the CSPs. To this end, banks and FIs possessing the e-KYC system need to encrypt the e-KYC data files before they are uploaded to the cloud. When the relying parties request for verification, the host party can either perform the verification by either decrypting the file and sending back the confirmation of the verification result to the requestor or transmitting the copy of encrypted files along with the decryption key to the requestor. This first approach introduces the overheads related to the verification process, communication, and centralized

The associate editor coordinating the review of this manuscript and approving it for publication was Mansoor Ahmed.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

decryption while the latter approach needs to handle key management especially secure key sharing. Specifically, key revocation and key re-generation in the cloud e-KYC environment have not been addressed by any research works. If the client would like to withdraw his consent from any banks or FIs, they have no right to store the client's identity data anymore. Accordingly, the data should be completely deleted and the decryption key needs to be revoked. Any banks or FIs sharing the revoked key need to regenerate a key to fully guarantee that unauthorized banks or FIs cannot access the client's data stored in the cloud.

In addition to the aforementioned problems, exiting cloud e-KYC platforms do not provide shared information for the transaction occurring in the e-KYC verification available for traceability.

Recently, blockchain technology has attracted huge interest by a number of enterprises in many industries including the banking and financial sector. There is a growing interest in using e-KYC platforms that use blockchain and cloud system. Blockchain technology truly promotes the decentralized system enabling transparency, agility, trustworthiness, and cost-effectiveness for transaction processing and management in multi-user and multi-provider environment. In the blockchain system, a smart contract which is a self-executing program that can be implemented on the blockchain enables the automated execution of system logics or functions efficiently. This empowers the usability and programmability of any systems running on the blockchain network.

For years, a number of research works related to blockchain-based KYC have proposed to deliver the decentralized authentication and verification process. However, there are shortcomings that have not been fully solved by existing works. First, there are no works that provide electronic client's consent function with the solid non-repudiation property which is an essential requirement of privacy regulations such as General Data Protection Act (GDPR) [18] in the KYC registration process. Second, most existing works overlook the privacy of transaction stored in the smart contract and blockchain. In addition to the identity or credential documents that are encrypted on the cloud storage, the privacy of all e-KYC processing transactions such as transaction status sharing, data origin authentication, and smart contract that contains personal data stored in the blockchain should be preserved. Finally, most works have a limited feature to allow the customers to access and update their credentials located on the cloud service paid by the FI.

In this paper, we aim to address such research gaps by introducing a secure and efficient blockchain-based e-KYC documents registration and verification process with lightweight key cryptographic protocols run in the cloud Interplanetary File System (IPFS). To facilitate the foundational privacy requirement regarding the user's consent collection, we develop a smart contract to generate and enforce the consent to be digitally signed by the customer. The consents will be systematically stored in a blockchain having tamper-proof property which is useful for auditing. Regarding the data privacy issue, we propose an optimized cryptographic protocol by applying symmetric encryption with public key encryption to encrypt the customers' credential files and employ the ciphertext policy attribute-based encryption (CP-ABE) to encrypt the blockchain transactions. Since CP-ABE provides a one-to-many encryption with fine-grained access control, it allows several FIs to access common encrypted transactional data in the blockchain of the same client based on the access policy defined. Specifically, we devise the policy update algorithm to enable efficient re-encryption based on a less complicated policy tree structure. Finally, our system allows users to update their e-KYC data with any banks or FIs engaging in the blockchain. The updated e-KYC data is broadcasted in the ledger and the synchronization of the updated data is done by the responsible smart contract.

This paper is structured as follows. Section 2 presents related works. Section 3 explains the theoretical background used in our proposed approach. Section 4 presents our proposed system model. Section 5 provides the security analysis of our scheme. Section 6 provides the evaluation analysis and experiments. Section 7 gives conclusion and future work.

II. RELATED WORKS

At present, blockchain technology and smart contracts have been leveraged in many application areas. Particularly, blockchain-based identification and authentication framework have been proposed by many works [1], [2], [7], [8], [12], [15] and it has been demonstrated that a blockchain is efficient for identification and authentication management. However, the process of e-KYC is much more complicated than simple authentication task. Rather, it involves secure credential registration, KYC document management, secure and lightweight verification process between clients, multiple FIs, and a dedicated blockchain platform. In addition, new kinds of remote and spoofing attack to the KYC system need to be countered [4]. Recent research works related to a blockchain-based e-KYC focus on devising a framework for secure user identity management and credentials verification as well as optimizing the communication overhead of the interaction among financial institutes.

In [3], the authors proposed a KYC document verification scheme using the IPFS system and blockchain technology. In this approach, the customers register their identity information with the bank and their credentials are hashed and encrypted by using gpg4win as an encryption tool. However, this paper does not concern itself with the privacy and traceability of transactions in the blockchains.

In [5], Shabair *et al.* proposed a blockchain-based KYC in the form of proof-of-concept (PoC) system. The proposed system was conducted in private blockchain environments over the Grid'5000 a large-scale distributed platform. In [6], Norvill *et al.* presented a system that allows automation and permissioned document sharing over the blockchain to reduce the KYC process. In [9], Allah *et al.* proposed a Hyperledger

Fabric network for KYC optimization model. In this model, the customer has full right to own the smart contracts in which customer KYC data is stored in the distributed ledger database. However, these works did not address the security and key management issue of KYC process.

In [10], Kapsoulis *et al.* proposed a way to implement e-KYC system using smart contracts and IPFS. In this work, KYC document operations such as create, read, update and delete are done through the set of smart contacts. The KYC documents are stored in the IPFS and through the private contract method. The security of the KYC transaction is managed by specific nodes in the blockchain with administrator privileges. However, there are no encryption used to protect the KYC data.

Regarding the privacy preserving technique applied for securing blockchain database, CP-ABE has received the attention of several research works [17]–[21], [24], [26], [27]. In [17], Bramm *et al.* proposed a Blockchain-based Distributed Attribute-Based Encryption (BDABE) scheme allows the attributes to be created and deleted dynamically at any time by a transaction on the blockchain. The proposed scheme supports mapping between multiple attribute authorities to assign the attributes to the users. It offers the flexibility for supporting secure and efficient user attributes management in the blockchain system.

In [18], Fan *et al.* proposed a traceable data sharing scheme using blockchain and CP-ABE. In this scheme, data is encrypted by a CP-ABE method and a secret key can be generated based on the system parameters available in the private blockchain. In the blockchain, the data owner can obtain the identity of data consumer and control data sharing based on the predefined access policy.

Yuan *et al.* [19] and Wu *et al.* [20] employed a CP-ABE approach to support data privacy protection and fine-grained sharing in the blockchain system. In these schemes, any changes to the data are recorded on the blockchain and the access policy is enforced to manage the different permissions of access. If there is any key abuse case initiated by any malicious users or authorities, the system provides audit trails to support the traceability of cryptographic operations and transaction activities.

Guo *et al.* [21] proposed a traceable attribute-based encryption with dynamic access control (TABE-DAC) scheme based on the combination of CP-ABE based linear secret sharing scheme (LSSS) and blockchain. The proposed scheme achieves fine-grained sharing of encrypted private data on cloud, traceability of users' private key leakage, and flexible policy update. The authors also introduced a hash function in the key and ciphertext generation to reduce the computation cost of such operations.

In [24], Gao *et al.* proposed a secure ciphertext-policy and attribute hiding access control scheme and blockchain. The CP-ABE is used to protect the data stored in the blockchain. However, this scheme uses composite order groups for their crypto implementation which results in expensive computation cost.

Recently, Dwevedi *et al.* [25] proposed a Zero-Knowledge Proof (ZKP) authentication scheme and the encryption scheme called ZKNimple for supporting lightweight encryption in IoT-based applications. With the proposed scheme, the authentication is achieved through the ZKP property while the security of key exchange and data is preserved through password-authenticated method and Feistel encryption.

In [26], Bhaskaran *et al.* proposed a design and implementation of a smart contract for consent-driven and double-blind data sharing on the Hyperledger Fabric blockchain platform. The smart contract for generating customer's consent was developed and published on the blockchain. The authors also presented public key sharing on the blockchain to multiple providers for encrypting the document. However, the consent provided by the customer has no digital signature binding.

To the best of our knowledge, we provide the first attempt applying CP-ABE for a blockchain-based KYC management with the user-controlled capability for protecting sensitive data contained in the blockchain. Existing schemes focus on protecting data files shared in cloud while the privacy of transaction data in the blockchain is overlooked. In addition, none of the above research has addressed the practical security and privacy issue with the aim of achieving both efficient security and privacy management compliance related to customer consent using digital signature in the e-KYC system.

III. BACKGROUND

This section describes the concept of blockchain used to support identity and access management system. Then, we provide the basic theory of CP-ABE.

A. KYC PRIVACY AND SECURITY COMPLIANCE

As the emergence of FinTech innovation and virtual banking has revolutionized the global financial service industry, several front-end services have shifted online. e-KYC is one such service that regulators of many countries have implemented policies that allow FIs to implement e-KYC verifications and approve customer applications. Based on the thorough review of a survey of KYC regulations done by Price Waterhouse and Coopers [28], Technical Standard for Digital Identification Systems published by World Bank Group [29], and the report on existing remote on-boarding solutions in the banking sector by EU commissions [30], the security and privacy-related compliance regulated by financial institutions around the globe take customer due diligence as the core consideration and emphasize the following four common requirements for digital identification including KYC compliance.

- Verification of customer identification informationmust be truly authenticated multiple factors and data sources. The proof of identity (POI) must be identifiable and technically and legally valid without tampering. Multiple sources of POI issued by government units and trusted ID providers are required.

TABLE 1. KYC security requirements and our e-KYC trustblock features.

Security and Privacy Requirement	Feature of Our e-KYC TrustBlock
Verification of customer identification information	Our scheme allows any form of POIs to be registered and singed by the customer. We also have a smart contract to support secure verification of the encrypted documents. The authenticity of the customers is firmly verified through their digital signature while the examination of the documents is vetted with FI's officer.
Protection of Customers' credentials or PII	All customers' credentials are protected based on AES and RSA encryption
Auditing Feature	All e-KYC transactions are recorded in the blockchain with the encrypted format
Consent	We develop two algorithms including (1) *create e-consent* and (2) *enforce e-consent* to systematically create digital consent and enable the customer to digitally sign the consent.

- Privacy of customers' credentials or PII should be protected. Encryption and digital signing based on PKI should be employed [29].
- Auditing feature for all transactions and its lineage must be provided.
- Collecting the customers 'credentials must obtain consent from the customers.

With the above requirements, we introduce a blockchain-based e-KYC to enhance accessibility, verification efficiency with high trust and accountability. Our proposed scheme satisfies the requirements as shown in the Table 1.

B. BLOCKCHAIN IN IDENTITY MANAGEMENT SYSTEM

Blockchain technology delivers a decentralized database where multiple nodes are linked to one another by the communication network. Blockchains are constructed from cryptographic mechanism, data storage, networking, and incentive schemes to support decentralized transaction management where multiple parties can check, execute, and store the data. Specifically, the blockchain stores transaction details and each completed block is assigned with a cryptographic ID called hash value. Since 2009 Satoshi [13] introduced the BitCoin concept based on the use of blockchain technology. It thus came into the public's view as a proven technology that facilitates secure and distributed cryptocurrency.

In addition to the decentralized transactional data storage and sharing, blockchain technology can empower its technical use and implementation flexibility with ''smart contracts''. Smart contracts are programmable and self-executable code that enforce predefined actions whenever a given set of conditions is met [14]. With the benefits of decentralized model, transparency, traceability, and immutability, blockchain has been now employed by many application areas including KYC platform.

C. CIPHERTEXT POLICY ATTRIBUTE-BASED ENCRYPTION (CP-APE)

Bethencourt *et al.* [11] originally proposed the formal concept of ciphertext policy attribute-based encryption in 2007. Technically, the core construct of CP-ABE construct is relied on bilinear maps where its mathematical formulation is shown below.

Bilinear Map: Let G_0 and G_1 be two multiplicative cyclic groups of prime order p and e be a bilinear map $e : G_0 \times G_0 \rightarrow G_1$. Let g be a generator of G_0. Let $H : \{0, 1\}^* \rightarrow G_0$ be a hash function that the security model is in random oracle.

The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in G_1$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degeneracy: $e(g, g) \neq 1$.

Definition 1: Let a set $\{P_1, P_2, \ldots, P_n\}$ be given. A collection $\mathbb{A} \subset 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\in \forall B, C$: *if* $B \in \mathbb{A}$ *and* $B \subset \mathrm{C} \longrightarrow \mathrm{C}\mathbb{A}$.

An access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e. $\mathbb{A} \subset 2^{\{P_1, P_2, \ldots, P_n\}}/\emptyset$.

Definition2 (Access Tree T [11]): Let T be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. The *kofn* threshold gate is also allowed in T, in this case $k_x = k$ where k is the threshold value determined in the *kofn* gate.

IV. OUR PROPOSED APPROACH

A. SYSTEM OVERVIEW

This section describes the system model of e-KYC TrustBlock and provides the details of its system components.

Figure 1 presents the overview of our e-KYC TrustBlock System Model

The system model consists of the following entities: authority, clients, financial institutes, IPFS, blockchain, and three smart contracts.

- **Authority**: The authority generates the public parameter *PK* and the master private key *MSK* of the system. The authority keeps the *MSK* secret and publishes *PK* available for the subscribers. The authority also

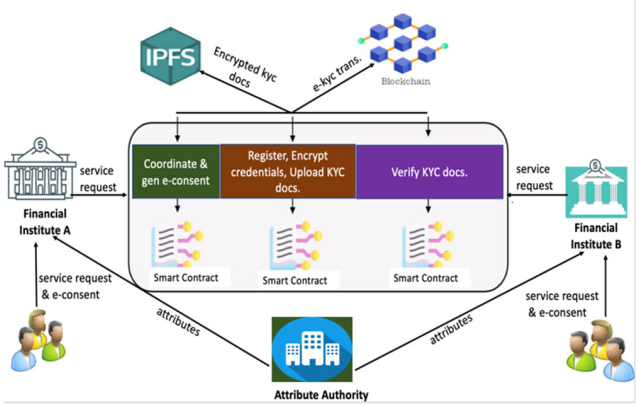

FIGURE 1. e-KYC trustblock system model.

generates a secret key generated based on the CP-ABE method and that key is issued to each financial institution (FI).

- **Clients** are the customers of financial institutes who join the blockchain-based KYC. Each customer has her own key pair used to encrypt and decrypt her credential data. To allow the credentials to be stored in any FI's database or in the cloud system, the FI must get the consent digitally signed by the client.
- **IPFS** is a cloud database that stores encrypted documents of KYC bound to each user account. It serves for user's credentials to generate transaction for cryptocurrency. It houses distributed hash table (DHT) keeping the address of the hash value of the clients' credential files which are encrypted in the IPFS storage.
- **Blockchain** is used to store the transactions of all KYC-related activities. All sensitive transactions of the clients are encrypted. The data on the blockchain is tamper-proof based on hash value and cryptography mechanism, which also prevents some illegal activities.
- **Smart contracts** are used to control and automate all KYC processes. In our system, there are three smart contracts including (1) Register contract is responsible for authenticating users, enrolling new users, and uploading the encrypted credentials to the IPFS, (2) Master contract is responsible for controlling client profiles, keeping hash value of the citizen ID of all clients for interacting with IPFS, and e-consent generation, and (3) Verify contract is responsible for KYC verification.

In the next section, we describe two core processes of our system. It includes client registration and e-KYC document uploading, and e-KYC verification. We describe the details of each process through the smart contracts developed for automating core e-KYC processes.

B. CLIENT REGISTRATION

The client registration process comprises the following steps.

1. The client registers to the system with her identity information and public key.
2. The wallet in the blockchain platform returns a key pair ($PubK_{Client\_id}$, $PrivK_{Client\_id}$)to a client.
3. FI calls *Master contract* to generate e-consent.
4. The Client digitally signs e-consent by using $PrivKey_{client\_ID}$
5. FI calls the *Register contract* to enroll the client in the system.
6. The client submits her credential documents *Crenden Files* to the FI. Then, FI stores the *Crenden files* in its local database
7. Register contract generates an AES session key to encrypt the e-KYC document and asks the client to encrypt the session key by using a client's public key $PubKey_{client\_ID}$. The encrypted *Crendenfile (Enc-CredenFile)* and the encrypted session key *(ESK)* are uploaded to the IPFS storage and the blockchain respectively.
8. The IPFS storage stores the data file into a corresponding storage node and generates hash value of file id, client's citizen id with SHA-256 and automatically returns a hash value *h* which is kept in the DHT table and the Master contract. This hash value is used as the index to link to the *EncCredenFile* located in IPFS.

The detail of the Register contract is presented as follows.

Algorithm 1 Register

```
Procedure
struct AESData{bytes encryptedKey;}
function registerIdentity (clientAddress, userId,
clientName, country, Image, PassportID, userAccount)
        Userstorage client= clients[userAddress];
        //to check that the client did not already exist
        require(! client.set);
        //store the client
        users[userAddress] = Client({
                id: clientId,
                name: clientName,
                publickey: publicKey,
                Image: CrendenFile
        });
        emit EncryptFiles(Image, AESKey);
                EncryptKey(AESKey, PubKey_client_ID)
        //Store a collected image and encrypted key ESK
into IPFS distributed system with a hash of CredenFileID
        FileID = StoreImage(EncCrenden)
        //Transform the ID and other personal info (e.g.,
PassportID to a new hash value)
        h = TransformData(CredenFileID, userId)
        ContractAddress = Deploy(clientId,
clientAddress, clientAccount, h)
}
function deleteIdentity(clientAddress )
external;
functionstoreClientDataHash(cientId, dataHash) public {
        clientDataHashes[clientId] = dataHash;
end procedure
```

Below shows the function of the Master contract that supports e-consent generation and enforcement for the process of client registration and verification process.

Algorithm 2 Create e-Consent

Input: Parameter $P = (pu_1, \ldots, pu_n)$ where p is the purpose for processing personal e-KYC credentials *Creden*, parameter *CP* denotes the consent process which can be the consent used for registration stage (InReg) or the consent used for verification stage (InVer), *DS* is the data subject or the client, *FI* is the financial institute, *S* is the sensitivity level which can be Low, Medium, High or Critical

Output:e-Consent *C*

$C \leftarrow$ e-Consent()
for each purpose $P \in (p_1, \ldots, p_n)$ ^consent process *CP*

do

P,CP ←*pu, CP{InReg, InVer}*

$S \leftarrow$ SensitivityLevel(Low, Medium, High, Critical)

CD=ConsentData(*Creden*, *FI,DS*)
return *C* ← Consent (*CD*, *P*, *CP*)

The above algorithm is used to create e-consent where the purpose such as storing, disclosing, transferring, and exporting credential data of the data subject or client is specified for the registration or verification processing transaction. The output is an e-consent generated to ask the client to digitally sign. Below is the function for enforcing e-consent to the client.

Algorithm 3 Enforce e-Consent

Procedure
Function enforce_e-consent(clientId)
if (msg.sender!= owner) {throw; }
let privateKey = new clientId(accounts[selectedAccountIndex].key, 'hex')
ifconsent==true **then**
registerIdentity sign = registerIdentity($PrivK_{Client\_id}$)
end if
if consent==false {throw;}
}
end procedure

C. e-KYC VERIFICATION PROCESS

The specific steps of the protocol in this process are as follows:

1. The client submits the request for the e-KYC verification by using her citizen id to the FI she is contacting.
2. The requesting FI calculates the hash value and transfers it to the *Verify contract*.
3. The *Verify contract* compares the hash value of the newly submitted value and checks with the one stored in Master contract.
4. If the hash value is found, the Verify contract checks the address of the files in the DHT in the IPFS to get the address of the corresponding *EncCrenden file* and corresponding *ESK*.
5. The *Verify contract* sends a request to the Master contract to generate e-consent.
6. The *Verify contract* transmits the *EncCrenden file* and *ESK* with e-consent to the FI requesting KYC verification.
7. The client digitally signs an e-consent and decrypts her *ESK* by using the client's private key $\text{PrivKey}_{client\_id}$ and use the session key to decrypt the *EncCredenFile.*
8. The requesting FI stores the *CredentFile* of the client into their local database.
9. The system records the verified transaction and the state of the smart contracts in the blockchain.

The details of the Verify contract are shown below.

Algorithm 4 Verification

Procedure VerifyProcess(*requestID*, *citizenID*)
emit DecryptFile(*h*, *privatekey*);
FileEnc = GetImage(*h*)
DecryptAESKey= (*ESK*, $PrivK_{Client\_id}$)
DecryptImg = TransformData(*EncCredenFile*, *AESKey*)
currentClient=Verify(*citizenID*)
match=compare(*h,* currentUser)
if match == true **then**
Address = ContractAddress(*h*)
clientEncCreden = Address.IPFS.getFile(h.FileID)
e-consent(h.userID)
CredentFile = DecryptFile(*h*, privateKey)
end if
SaveToLocal(CredentFile)
end procedure

D. PRIVACY-PRESERVATION OF SENSITIVE DATA IN BLOCKCHAIN

Due to the decentralized nature of blockchains, e-KYC transaction data related to personal information of clients, state of smart contract are replicated and stored on untrusted nodes. This makes the transaction data insecure and it causes the issue of personal data privacy compliance.

In our scheme, we propose an attribute-value encryption scheme to structure the transaction data in the blockchain that is encrypted by a transaction key which is a symmetric encryption for protecting the sensitive data in the e-KYC blockchain. Since blockchains are tamper-proof, manipulating and deleting data in the blockchain is difficult. The attribute-value pair of PII and smart contract state is encrypted by the Master contract done during the TLS communication. For instance, the transaction structure of e-KYC done at time *t* consists of <TransID, {e-KYC operation,

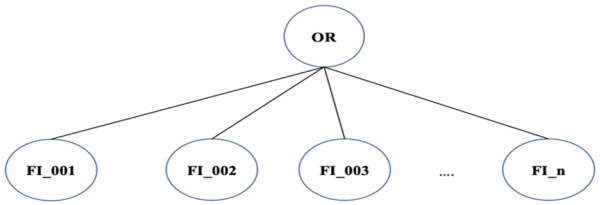

FIGURE 2. Access policy.

TABLE 2. Notations used in our scheme.

Notation	Meaning
S_k	Set of all attributes issued by authority k
S_{fi_id}	Set of all attributes issued to FI fi_id
SK_k	A secret key which belongs to authority k.
PK_k	Public key which belongs to authority k.
$PubKey_{MasterContract}$	A public key issued to the Master contract
$PubKey_{fi\_id,}$	A public key issued to each FI fi_id
$PrivKey_{fi\_id}$	A public key issued to each FI fi_id
$FISK_{fi\_id}$	Secret key generated by the authority based on CP-ABE method and issued to each FI fi_id
EDK_{fi_id}	EDK is an encrypted form of a $FISK$ which is encrypted by a FI's public key.
$SymKey$	A symmetric key created from the AES algorithm.
ACP_{Pid}	An access control policy used to encrypt the data in blockchain. It is handled by the Master contract
ACP'	An updated access control policy
CT_M	An encrypted data in the blockchain
CT_K	An encrypted $SymKey$

type}, {FI Name, Value}, {ClientID, Value}, {ClientName, Value}, {Address, Value}, {FinancialService, Value}, ..., {SmartContract State, Value}>. The tuple of transaction is encrypted and written in the blockchain.

To provide secure and fine-grained access to transaction data stored in the blockchain, the transaction key is encrypted with the CP-ABE method and the encrypted transaction key (Enc_TranKey) is stored in the blockchain. In our scheme, the client id and the block used are managed by the *Master contract*. The FIs participating in the e-KYC platform can verify the transaction, if needed. To this end, the transactions can be updated by adding financial institution ID *FIid* to the access policy configured in the master contract. Then, the encrypted transaction key is re-encrypted. The new FI then can decrypt the transaction stored in the blockchain without updating any information in the blockchain.

Figure 2 shows the access policy used to encrypt the transaction key.

As shown in Fig.2, the access policy consists of the FI ids and OR gate. The number of attributes is optimized and its structure becomes less complex for the encryption and decryption process.

The cryptographic construct for secure access control to sensitive data in the blockchain consists of five phases as follows.

Phase 1 (System Setup): **CreateAttributeAuthority**(k) $\rightarrow$ PK_k, SK_k, $PK_{x.k}$. This algorithm uses an attribute authority ID(k) and selects a bilinear group $G_0$of prime order p with generator g. Then, it selects two random$\alpha, \beta \in Z_p$ to compute the public key defined as follows:

$$PK_k = \left\{G_0, g, h = g^{\beta}, f = g^{\frac{1}{\beta}}, e(g,g)^{\alpha}\right\}.$$

Then the authority computes the secret key SK_k as (β, g^{α}).

Phase 2 (Key Generation): In our model, we define two key types used by the data owner and financial institutes. The crypto process of key generation is as follows.

- **SymKeyGen**(*AESKeyGen, randomstring*)$\rightarrow$ AES_ key. The Master Contract takes as inputs SymKeyGen algorithm and random string to generate a 256-bit AES key. Then, the data owner uses public key encryption method by taking user's public key to encrypt the generated AES_key or *SymKey* and it is uploaded to the cloud.
- **FIKeyGen**($S_{fi\_d,}$, SK_k, $PubK_{fi\_id}$) $\rightarrow$ EDK_{fi_id}. FIKeyGen algorithm consists of two steps:
  (1) FISKGen. It takes as input a set of attributes ($S_{fi\_id,k}$) identifying the *FI_id*'s decryption key, AA's secret key (SK_k), and $PubK_{fi\_id}$ of FI *fi_id*. For each FI *fi_id*, the *AA* chooses a random r and $r_j \in \mathbb{Z}_p$, for each attribute $j \in S$. Then the FI decryption key ($FISK_{fi\_id}$) is computed as:

$$FISK_{j,k} = (D = g^{(\alpha_k + r)/\beta_{k,}} A_i \in S_k : D_i = g^r.H(i)^{r_i}, D'_i = g^{r_i}).$$

  (2) EDKGen. The algorithm encrypts the $FISK_{fi\_id}$ by using the public key of FI. The encryption is computed as:

$$\mathrm{ENC}_{\mathrm{RSA}}(PubK_{fi\_id}, FISK_{fi\_id}) \equiv EDK_{fi\_id}$$

Then, EDK_{fi_id} is sent to each client.

Phase 3 (Encryption): We present a two-layer encryption scheme comprising symmetric key encryption and CP-ABE encryption. The detail of the algorithm is described as follows:

ENC(PK_k, *SymKey, M, ACP*$_{Pid}$) $\rightarrow$(CT_M, CT_K). Each encryption layer is done through the following steps.

(1) Encrypt Message M: the algorithm is run by the Register contract. It takes symmetric key *SymKey* to encrypt transaction data *M*. The algorithm produces ciphertext CT_M and stores it in the IPFS on cloud. The function is defined as:

$$M \mapsto \mathrm{ENC}_{\mathrm{AES}}(SymKey, M) \equiv CT_M$$

(2) Encrypt *SymKey*: the algorithm takes as inputs authority public key PK_k, *ACPpid,* and *SymKey*. Then, the encrypted SymKey CT_K

is produced by the following encryption function:

$SymKey. \mapsto \text{ENC}_{\text{CP-ABE}}(PK_k, ACP_{Pid}, SymKey) \equiv CT_K$

The CT_K is then stored in the blockchain.

Phase 4 (Decryption): The decryption phase is done by the Master Contract and legitimate FI.

$\text{DEC}(PK_k, FISK_{fi\_id}, CT_M, CT_K) \rightarrow M$. The algorithm performs two following steps.

(1) Decrypt CT_K The algorithm takes as inputs a $FISK_{FI\_id}$ and CT_K. The output is a symmetric key *SymKey*. The function is defined as follows.

$$\text{SymKey} = \text{DEC}_{\text{CP-ABE}}(CT_K, FISK_{fi\_id})$$

(2) Decrypt CT_M. The algorithm takes as inputs *SymKey and* CT_M. It then produces data M.

$$M = \text{DEC}_{\text{AES}}(SymKey, CT_M)$$

Upon the successful decryption, the legitimate FI can verify the client credentials and make decision on KYC validation.

Phase 5 (Update Policy): In this phase, our scheme accommodates the policy change for supporting e-KYC by adding a new FI or revoking the existing FI. The new or existing attribute of the *fi_id* is updated. The policy update process is as follows.

(1) Update attribute in the policy
Case Adding a new fi_id into the policy.
$ACP'_{Pid} = ACP_{Pid} U \{\text{new}fi\_id\}$
Or
Case Deleting existing
$ACP'_{Pid} = ACP_{Pid} - \{fi\_id\}$

(2) Re-encrypt SymKey with a new ACP'_{Pid}
$\text{CT}'_{\text{K}} \mapsto \text{ENC}_{\text{CP-ABE}}(PK_k, ACP'_{Pid}, SymKey) \equiv CT'_K$

The cost of policy update of our scheme thus deals with only single encryption layer for re-encrypting the symmetric key.

V. SECURITY ANALYSIS

In this section, we present the security analysis of our proposed scheme by presenting the security model and security features.

A. SECURITY MODEL

In our model, we assume that the e-KYC platform in run on consortium blockchain. All blockchain nodes and the cloud are considered as honest but curious. Since our core cryptographic protocol for protecting the content of blockchain transaction is based on CP-ABE, the security model and its proof is shown in [11].

B. SECURITY FEATURES

In addition to the security model shown above, our proposed scheme also achieves the following security features.

1. **Privacy-preserving e-KYC credentials with client consent.** Our scheme applies a combination of symmetric encryption and public key encryption for protecting the e-KYC documents before they are stored in the cloud. To conform some privacy regulations related to cloud data privacy and auditing [22], the responsible smart contract generates e-consent and asks the client to sign the consent to allow the FI to use and store their personal data. This fully satisfies privacy preserving feature with non-repudiation.
2. **Secure and Fine-grained access control to transaction data stored in the blockchains.** Our scheme uses symmetric key encryption and CP-ABE encryption to encrypt the transaction data and secret key of FIs respectively. This ensures that the confidentiality of transaction data is only accessed by legitimate FIs. Since we apply the CP-ABE method, the detailed proof of its security can be referred to the original CP-ABE [11].
3. **Collusion attacks Resistance:**Assuming that two different FIs, FI *A* and FI *B* collude to attack our system by using their secret keys which contain different attribute sets, CSP (Cloud Service Provider) still sends them the CT_K. However, they cannot combine the attributes to gain access to the encrypted symmetric key because of the CP-ABE key construction property [11] and the access policy specification.
4. **Traceability:**All user access activities regardless of clients and participating FIs are available in the blockchain. Any authorization information and smart contract state are retained as immutable transactions. Any FI member or third party auditors will know who performed the activities or accessed the locally stored data and they cannot deny the access operation. Also, legitimate FIs can detect unauthorized accesses or illegal attempts through verification.

VI. EVALUATION AND EXPERIMENT

This section describes the evaluation of our implementation of our proposed scheme through the evaluation analysis and experiments.

A. FUCTIONAL ANALYSIS

We compare the functional system between our proposed scheme with two blockchain-based KYC schemes [3], [26], and two blockchain-based IDM schemes including L. Guo *et al.*'s scheme [21] and S. Gao *et al.*'scheme [24].

As shown in Table 3, all schemes use blockchain and cloud storage. To protect the shared data, scheme [3] and [26] apply public key encryption to encrypt the KYC documents and upload the ciphertext to store in cloud and blockchain respectively. Scheme [24] also provides the features of customer consent shared in the blockchain. In scheme [21] and [24], LSSS based CP-ABE is used to encrypt the key and file while our scheme is based on tree-based CP-ABE

TABLE 3. Comparison of system functions.

Scheme	Cloud -Block chain	Client Consent	Privacy preserving documents and transactions	Access Policy Update
[3]	✓	X	X	X
[21]	✓	X	X	X
[24]	✓	X	X	X
[26]	✓	✓	✓	X
Ours	✓	✓	✓	✓

TABLE 4. Comparison of the computation cost.

Scheme	Computation Cost on Cloud/Blockchain	
	Encryption	Decryption
[3]	*PubEnc*	*PubDec*
[21]	$SymEnc+(4+\|l\cdot n\| +\|l\|)C_e+(1+\|l\cdot n\|)C_m$	$SymDec+1+2\ \|Att\|\ C_e +2\ \|Att\|C_m+3C_p$
[24]	$(2+\|l\cdot n\|)C_e +(2+\|l\cdot n\|C_m$	$\|S\|C_m+1+\|S\|C_p$
[26]	*PubEnc*	*PubDec*
Ours	$SymEnc+PubEnc+(2 +2\|T\|)C_e+C_m$	$SymDec+\|N\|C_e +(1+2\|S\|)C_p+ \|N\|C_p$

which provides more intuitive expression of the policy and less complexity.

In addition, our scheme provides the e-consent based on digital signing along with the e-KYC and IDM process. Hence, our scheme can satisfy privacy compliance without implementing additional consent management form this application. Regarding the privacy-preserving KYC data and transaction, our scheme supports the confidentiality of both ID data files located in the cloud and sensitive transactions stored in the blockchain while the other two schemes only concern the privacy of the ID data files. Finally, only our scheme allows the policy update to be done efficiently by the authorized party.

B. COMPUTATION COST ANALYSIS

We compare the computation cost of our proposed scheme with [3], [21] and [24]. The following notations are used to describe the computation cost of all schemes.

- C_e: Exponentiation and XOR operation cost
- C_p: Pairing operation cost
- C_m: Multiplication operation cost
- $|S|$: The size of user attribute set
- $|T|$: The number of leaf nodes in access control policy.
- $|N|$: The number of nodes in access control policy.
- $|l|$: The number of rows in the LSSS matrix.
- $|n|$: The number of columns in the LSSS matrix.
- $|Att|$: The number of attributes satisfying the policy.
- *SymEnc/Dec:* Symmetric Encryption/Decryption cost based on 256-AES
- *PubEnc/Dec*: Public key encryption cost based on 1024-bit RSA

As shown in Table 4, scheme [3] and [26] use the user's public key encryption to encrypt KYC documents and they are stored in the cloud and blockchain respectively. Only the user that has the corresponding private key can decrypt the file for verification. In this scheme, the cost of encryption and decryption are subject to the size of the key and the file. The encryption cost of scheme [21] and [24] relies on the size of the matrix while our scheme is subject to the no. of leaf nodes in the access tree. In the decryption process, the major cost of all schemes varies on the bilinear pairing operations and the no. of attributes contained in the key. However, our scheme is designed to use only the ID attributes of the legitimate financial institutes joining the blockchain. Hence the number of attributes used in the access policy and attribute-based decryption is much fewer than those in [21] and [24] where the number of attributes in the access policies are characterized by the data owners and authorized users.

C. COMMUNICATION COST ANALYSIS

The communication cost of accessing plain data for the cryptographic-based access control is generally subject to the size and times of sending and receiving the cryptographic elements such as the signature and secret key sent between the data user, the cloud, and the blockchain. In [21], the user needs to access the encrypted key stored in the blockchain and decrypt the key. Hereafter, the user then accesses the cloud where the ciphertext of data is stored and uses the symmetric key to decrypt the data. In [24], the signature proof based on the ElGamal public key cryptosystem is generated and sent to the user every time the user requests access to the system. Then, the user can use a secret key to decrypt the ciphertext stored in the cloud. In Mamun *et al.*'s scheme [3] and ours, the system checks the request of users to verify the identity documents and returns the ciphertext and encrypted key to the client. Hence, there is no communication cost on the client side to connect to the cloud or blockchain to get the ciphertext for the decryption. Therefore, the communication cost of [21] and [24] are higher than [3] and ours.

D. PERFORMACE ANALYSIS

For the experiment, the evaluation was focused on two scenarios: measuring the encryption and verification (request verification and decryption) performance of the ID document and the blockchain transaction. In our experiment, we did a simulation to compare the performance test of the encryption

TABLE 5. Simulation parameters.

Resource	Parameter	Specification
Hardware		
Server	CPU	Intel® Xeon® E-2224 3.4GHz, 8M cache,
	RAM	16 GB
	HDD	960 GB SSD
Software		
	OS	Linux ubuntu16.04 TLS
	VMWare	ESXi 7.0
	Crypto Library	Java Pairing-Based Cryptography
Blockchain		
	Block Size	512 KB
	Transaction Timeout	1000 MS

and document verification time between our scheme and three other works including [3], [21], and [24].

All experiments were conducted on Intel®Xeon®E-2224 3.4GHz, 8M cache. We used the VM to run Ethereum blockchain in the Linux ubuntu16.04 TLS. The smart contracts were developed using the Solidity language [31]. The standard AES and RSA encryption and signing provided by the Ethereum blockchain are used for [21] and our scheme. The CP-ABE toolkit and Java Pairing-Based Cryptography [23] are used to implement all schemes. Table 5 summarizes the simulation parameters used in our experiments.

In the experiment setting, we conducted the performance evaluation by comparing the e-KYC documents encryption and decryption or the verification performance of our scheme and three works including Mamun *et al.*' scheme [3], Guo *et al.*'s scheme [21], and Gao *et al.*'s scheme [24]. To compare the run times of different schemes, the file sizes of the identity document were varied and the processing time was measured. Since the performance of the identity documents encryption and decryption in scheme [21] and [24] is subject to the number of attributes used in the access policy and the secret key, we also included this factor in their test results. After running the test 100 times, the average time of the encryption and decryption of our scheme and related works Fig. 3 and Fig.4 respectively.

Figure 3 and Figure 4 show that our scheme consumed least processing time of both encryption and decryption. This is because our scheme applies symmetric encryption to encrypt and decrypt the data and use a public key method for encrypting and decrypting the symmetric key. The performance is thus efficient as the property of symmetric encryption and decryption and the processing time is only dependent on the file size. In [3], the RSA encryption is applied directly to encrypt and decrypt the data and

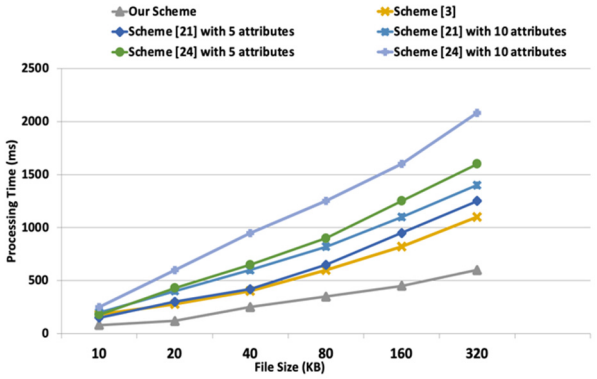

FIGURE 3. Encryption time.

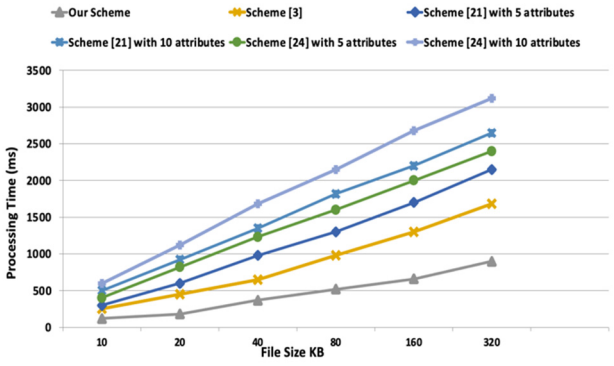

FIGURE 4. Decryption time.

ciphertext. The performance is greater than ours and it is propositional to the file size. L. Gao *et al.*'s scheme [21] with five attributes of the access policy yield a comparable encryption and decryption time to [3] as this scheme applies a symmetric key to encrypt the data and uses the CP-ABE technique to encrypt the symmetric key. However, when the number of attributes was increased to 10, the runtimes of both encryption and decryption are linearly increased. The scheme [24] has the highest encryption and decryption time since it applies the ABE method to encrypt and decrypt the data and ciphertext directly. When the attribute size increases to 10, the runtime increases double of the 5 attributes.

For the blockchain transaction encryption and decryption performance, ours is comparable to [21] where symmetric encryption and CP-ABE are used. However, our scheme invokes CP-ABE method to encrypt transaction key by using an access policy consisting of a fixed number of financial institutes which is generally much fewer than the number of client attributes in real scenario. In real practice, our scheme should provide more efficient running time for both credential documents and transaction encryption/decryption than existing schemes.

In addition to evaluating the performance of encryption and decryption, we also set up the experiment to measure the policy update cost of three schemes including scheme

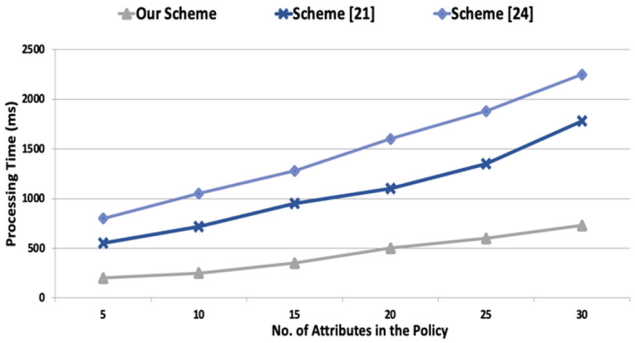

FIGURE 5. Policy update cost.

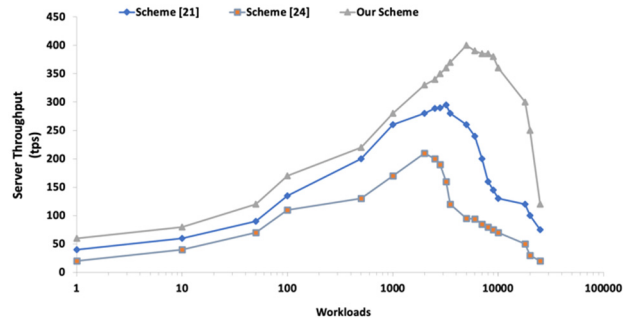

FIGURE 6. Throughput of policy update.

[21], [24], and ours. We measured the policy update time by considering the update setup cost and ciphertext re-encryption. The ciphertext size is 50 KB. Fig. 5 and Fig.6 show the policy update cost when the number of attributes in the policy was increased and the policy update throughput when there were the increased requests for policy update respectively. For the throughput test, the policy size was fixed at five attributes and we used Apache JMeter to measure the performance.

As displayed in Fig. 5, the processing time for all scheme is propositional to the increased policy size. Clearly, our scheme outperforms [21] and [24] because our scheme does not deal with the re-generation of signature as in [21] and re-computation of hidden policy to update the ciphertext. In our scheme, when there is a change of policy, the re-encryption of the transaction key is required without additional cost for re-computation of any cryptographic elements.

Thanks to the least effect of the policy update cost described above, our scheme provides the highest throughput performance as it can accommodate more requests which are the workloads emanated from the policy update impact. As shown in Fig.6, our scheme yielded the best throughput at around 400 tps when there were 5,000 requests while scheme [21] and scheme [24] achieved 295 tps with 3,200 requests and 200 tps with 2,800 requests respectively. After they reached the max throughput, their performance kept declining as the computation resources were exhausted.

Significantly, the experimental results confirmed that our proposed e-KYC TrustBlock is efficient and scalable in supporting e-KYC registration and verification process with transaction traceability.

VII. CONCLUSION

We have presented the privacy-preserving e-KYC approach based on the blockchain. Our proposed scheme delivers secure and decentralized authentication and verification of the e-KYC process with the user's consent enforcement feature. In our scheme, the privacy of both customers' identity documents stored in the cloud is guaranteed by the symmetric key and public key encryption while the sensitive transaction data stored in the blockchain is encrypted by symmetric key encryption and CP-ABE. Our scheme also allows the KYC data to be updated by the data owner or the customer. In addition, we devised an access policy update algorithm to enable dynamic access authorization. For the evaluation, we performed comparative analysis between our scheme and related works in terms of the computation cost, the communication cost, and performance. The experimental results showed that our scheme outperforms existing schemes in terms of performance, comprehensive KYC compliance features, and the scalable access control mechanism. For future works, we will test a larger sample of data in the real cloud environment and measure the throughput of the system in accommodating high number of e-KYC registration and verification requests. In addition, we will investigate the technique to enable batch verification of e-KYC transactions stored in the blockchain with the searchable encryption feature.

REFERENCES

[1] Y. Zhong, M. Zhou, J. Li, J. Chen, Y. Liu, Y. Zhao, and M. Hu, "Distributed blockchain-based authentication and authorization protocol for smart grid," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–15, Apr. 2021, doi: 10.1155/2021/5560621.

[2] S. Y. Lim, P. T. Fotsing, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, and R. Ismail, "Blockchain technology the identity management and authentication service disruptor: A survey," *Int. J. Adv. Sci. Eng. Inf. Tech.*, vol. 8, pp. 1735–1745, Sep. 2018.

[3] A. A. Mamun, A. Al Mamun, S. R. Hasan, S. R. Hasan, M. S. Bhuiyan, M. S. Bhuiyan, M. S. Kaiser, M. S. Kaiser, M. A. Yousuf, and M. A. Yousuf, "Secure and transparent KYC for banking system using IPFS and blockchain technology," in *Proc. IEEE Region Symp. (TENSYMP)*, Jun. 2020, pp. 348–351.

[4] M. Pic, G. Mahfoudi, and A. Trabelsi, "Remote KYC: Attacks and counter-measures," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Nov. 2019, pp. 126–129.

[5] W. Shbair, M. Steichen, and J. François, "Blockchain orchestration and experimentation framework: A case study of KYC," in *Proc. 1st IEEE/IFIP Int. Workshop Manag. Managed Blockchain (Man Block)*, Jeju Island, South Korea, Aug. 2018, pp. 23–25.

[6] R. Norvill, M. Steichen, W. M. Shbair, and R. State, "Demo: Blockchain for the simplification and automation of KYC result sharing," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 9–10, doi: 10.1109/BLOC.2019.8751480.

[7] T. Mikula and R. H. Jacobsen, "Identity and access management with blockchain in electronic healthcare records," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Prague, Czech Republic, Aug. 2018, pp. 699–706.

[8] S. Wang, R. Pei, and Y. Zhang, "EIDM: A ethereum-based cloud user identity management protocol," *IEEE Access*, vol. 7, pp. 115281–115291, 2019, doi: 10.1109/ACCESS.2019.2933989.

[9] N. Ullah, K. A. Al-Dhlan, and W. M. Al-Rahmi, "KYC optimization by blockchain based hyperledger fabric network," in *Proc. 4th Int. Conf. Adv. Electron. Mater., Comput. Softw. Eng. (AEMCSE)*, Mar. 2021, pp. 1294–1299.
[10] N. Kapsoulis, A. Psychas, G. Palaiokrassas, A. Marinakis, A. Litke, and T. Varvarigou, "Know your customer (KYC) implementation with smart contracts on a privacy-oriented decentralized architecture," *Future Internet*, vol. 12, no. 41, pp. 1–13, 2020.
[11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2007, pp. 321–334.
[12] I. Gutierrez-Aguero, S. Anguita, X. Larrucea, A. Gomez-Goiri, and B. Urquizu, "Burnable pseudo-identity: A non-binding anonymous identity method for ethereum," *IEEE Access*, vol. 9, pp. 108912–108923, 2021.
[13] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Jan. 8, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[14] J. P. Moyano and O. Ross, "KYC optimization using distributed ledger technology," *Bus. Inf. Syst. Eng.*, vol. 59, no. 6, pp. 411–423, Dec. 2017.
[15] A. Chowdhary, S. Agrawal, and B. Rudra, "Blockchain based framework for Student identity and educational certificate verification," in *Proc. 2nd Int. Conf. Electron. Sustain. Commun. Syst. (ICESC)*, Aug. 2021, pp. 916–921.
[16] *GDPREuropeanUnionGuidelines*. Accessed: Aug. 12, 2021. [Online]. Available: https://gdprinfo.eu/
[17] G. Bramm, M. Gall, and J. Schütte, "BDABE-blockchain-based distributed attribute based encryption," in *Proc. 15th Int. Conf. e-Bus. Telecommun.*, 2018, pp. 99–110.
[18] Y. Fan, X. Lin, W. Liang, J. Wang, G. Tan, X. Lei, and L. Jing, "TraceChain: A blockchain-based scheme to protect data confidentiality and traceability," *Softw., Pract. Exper.*, vol. 52, no. 1, pp. 115–129, Jan. 2022, doi: 10.1002/spe.2753.
[19] C. Yuan, M. Xu, X. Si, and B. Li, "Blockchain with accountable CP-ABE: How to effectively protect the electronic documents," in *Proc. IEEE 23rd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2017, pp. 800–803.
[20] A. Wu, Y. Zhang, X. Zheng, R. Guo, Q. Zhao, and D. Zheng, "Efficient and privacy-preserving traceable attribute-based encryption in blockchain," *Ann. Telecommun.*, vol. 74, nos. 7–8, pp. 401–411, Aug. 2019.
[21] L. Guo, X. Yang, and W.-C. Yau, "TABE-DAC: Efficient traceable attribute-based encryption scheme with dynamic access control based on blockchain," *IEEE Access*, vol. 9, pp. 8479–8490, 2021, doi: 10.1109/ACCESS.2021.3049549.
[22] M. Barati, G. S. Aujla, J. T. Llanos, K. A. Duodu, O. F. Rana, M. Carr, and R. Ranjan, "Privacy-aware cloud auditing for GDPR compliance verification in online healthcare," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4808–4819, Jul. 2022, doi: 10.1109/TII.2021.3100152.
[23] *PBC (Pairing-Based Cryptography) Library*. Accessed: Jan. 5, 2022. [Online]. Available: https://crypto.stanford.edu/pbc/
[24] S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma, "TrustAccess: A trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5784–5798, Jun. 2020.
[25] A. D. Dwivedi, R. Singh, U. Ghosh, R. R. Mukkamala, A. Tolba, and O. Said, "Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for Internet of Things," *J. Ambient Intell. Humanized Comput.*, pp. 1–11, Sep. 2021, doi: 10.1007/s12652-021-03459-4.
[26] K. Bhaskaran, P. Ilfrich, D. Liffman, C. Vecchiola, P. Jayachandran, A. Kumar, F. Lim, K. Nandakumar, Z. Qin, V. Ramakrishna, E. G. Teo, and C. H. Suen, "Double-blind consent-driven data sharing on blockchain," in *Proc. IEEE Int. Conf. Cloud Eng. (IC E)*, Apr. 2018, pp. 385–391.
[27] F. Ghaffari, E. Bertin, N. Crespi, S. Behrad, and J. Hatin, "A novel access control method via smart contracts for internet-based service provisioning," *IEEE Access*, vol. 9, pp. 81253–81273, 2021.
[28] (Jan. 2016). *Know Your Customer: Quick Reference Guide, Understanding Global KYC differences by PWC*. Accessed: Feb. 28, 2022. [Online]. Available: https://www.pwc.com/gx/en/financial-services/publications/assets/pwc-anti-money-laundering-know-your-customer-quick-reference-guide.pdf
[29] (2018). *Technical Standard for Digital Identification Systems Published by World Bank Group*. Accessed: Feb. 28, 2022. [Online]. Available: https://olc.worldbank.org/system/files/129743-WP-PUBLIC-ID4D-Catalog-of-Technical-Standards.pdf
[30] *European Commission's Expert Sub Working Group 1, Electronic Identification and Remote Know Your Customer Processes*. Accessed: Feb. 28, 2022. [Online]. Available: https://ec.europa.eu/info/sites/default/files/business_economy_euro/banking_and_finance/documents/report-on-existing-remote-on-boarding-solutions-in-the-banking-sector-december2019_en.pdf
[31] C. Dannen, *Introducing Ethereum and Solidity*. Berkeley, CA, USA: Apress, 2017. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-2535-6

SOMCHART FUGKEAW (Member, IEEE) received the bachelor's degree in management information systems from Thammasat University, Bangkok, Thailand, the master's degree in computer science from Mahidol University, Thailand, and the Ph.D. degree in electrical engineering and information systems from The University of Tokyo, Japan, in 2017. He is currently an Assistant Professor with the Sirindhorn International Institute of Technology, Thammasat University. His research interests include information security, access control, cloud computing security, big data analysis, and high performance computing. He has served as a reviewer for several international journals, such as IEEE Access, the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Dependable and Secure Computing, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Big Data, the IEEE Transactions on Network Science and Engineering, the IEEE Transactions on Network and Service Management, the IEEE Transactions on Parallel and Distributed Systems, Computer & Security, the IEEE System Journal, and *ACM Transactions on Multimedia Computing Communications and Applications*.