Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3200891, 14 pages
https://doi.org/10.1155/2022/3200891

WILEY | Hindawi

Research Article

A Blockchain-Based Key-Revocation Access Control for Open Banking

Khaled Riad [1,2] **and Mohamed Elhoseny** [3,4]

[1]*Computer Science Department, College of Computer Sciences & Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia*
[2]*Mathematics Department, Faculty of Science, Zagazig University, Zagazig 44519, Egypt*
[3]*College of Computing and Informatics, University of Sharjah, Sharjah 27272, UAE*
[4]*Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt*

Correspondence should be addressed to Khaled Riad; khaled.riad@science.zu.edu.eg

Received 9 November 2021; Accepted 30 December 2021; Published 31 January 2022

Academic Editor: Nima Jafari Navimipour

Open banking allows banks and financial sectors to easily access the customers' financial data which is revolutionizing. It also provides the customers with excellent cloud access to various providers' wide range of financial services. The storage of such sensitive services and data on cloud servers is a double-edged sword. It can ease and support fine-grained access to such services/data anywhere and anytime, supporting the open banking system. But, on the other hand, data privacy and secrecy are a challenge. Thus, efficient access control should exist for open banking's services and data to protect cloud-hosted financial sensitive data from unauthorized customers. This paper proposes a new access control scheme that employs blockchain for the key-revocation process. We implement the smart contract's functions on the Ethereum platform and test the contract's code on the Kovan Testnet before deploying it to the Mainnet. Although the customer is authenticated to open banking, his key/s can be revoked according to the status response of the bank branch. Thus, his access to financial services and data is denied. We did comprehensive experiments for the revocation status response time, data exchanged until receiving the revocation status, and the time spent updating the policy. Also, we compared the results of our proposed scheme with two well-known methods—Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP). The experimental results show that our proposed scheme (BKR-AC) has a faster response time than Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP) in case of nonrevoked keys/certificates and a slower response time in case of revoked keys to avoid nonrevoking a revoked key. But the data exchanged is an average for BKR-AC between CRL and OCSP, which is still a tiny amount and accepted. The security analysis proved that our scheme is secure against some well-known attacks on open banking systems. In addition, it is also secured against the chosen-text attack by employing the challenge-response authentication mechanism.

1. Introduction

Nowadays, most of the world and even some superpowers live in tough economic times. But, at the same time, the thing that cannot be neglected or ignored in any way is the online banking and finance sector. In such harsh economic conditions, the banks and financial institutions are under even greater scrutiny than ever. The key considerations of the banks and financial institutions are physical security and access control. Moreover, the open banking revolution allows banks and financial institutions to easily exchange the customers' financial data through Application Programming Interfaces (APIs). The main goal behind open banking is to improve the current infrastructure and introduce new features, which down the communication costs among different institutions. Access control can be quite challenging when a bank or financial institution has numerous branches and facilities spread over a large geographical area. Therefore, any access control deployed in the banking and finance sector must provide complete value for money and return on investment.

Authentication is the starting point for any successful access control. Authentication verifies the customer's identity and enables authorization. The authorization is the allowed permissions for a customer after the authentication process. For example, the bank's customer can use an identity (e.g., a username name) to log in to that online banking service. Still, the bank's authorization policy must ensure that this client is only authorized to access their own account's information/services. The bank and financial institution implicitly create and maintain the authorization policies. It is unbelievable that these authorization policies are kept unchanged forever. Access control is the method used to update and enforce such policies.

Since most of the banking and finance sector's services and data are currently hosted on the cloud, there should be a flexible, scalable, intelligent access control to protect sensitive financial data. Moreover, the user may be authenticated to the online banking but has a revoked key/s. Therefore, this user will not be able to access financial services. Financial institutions can use solutions that offer built-in security mechanisms, including encryption, two-factor authentication, and authorization. However, access controls are more critical to protecting customers and corporate information than facilities and property. Correct configuration of access privileges is crucial for protecting information from unauthorized access and protecting computer systems from abuse. Thus, we are proposing a new blockchain-based access control scheme that can revoke the customers' key/s after the authentication to online banking.

The process of revoking a previously issued key is similar to that of revoking a previously issued certificate before its expatriation. This revocation process is critical to any public key infrastructure [1]. The secure website could be vulnerable to attack if a new one replaces the certificate. Generally, a replaced certificate will remain useable by any attacker until reaching the expiry time. Thus, the replaced certificates must be revoked. The same applies to the keys. There are multiple certificate revocation schemes, for example, Online Certificate Status Protocol (OCSP) [2, 3], Certificate Revocation Tree (CRT) [4], and Certificate Revocation List (CRL) [5]. Since blockchain is efficient in introducing a decentralized certificate revocation management [6], we employed it to introduce an efficient key-revocation access control for open banking.

1.1. Motivation. The banks and financial institutions must not depend only on the naive online banking authentication system. Also, the key/s generated for an authenticated customer must not be working until their expiry date. Still, they must frequently pass under a revocation process to reevaluate their validity to be authorized for the open banking services. To the best of our knowledge in the practical portion, TDSi company (https://www.tdsi.co.uk/) is one of the UK's leading manufacturers of integrated access control systems, offering an extensive range of readers, controllers, and software systems. TDSi provides a suite of solutions that meet the banking and finance industry's many demanding access control needs. On the other hand, TDSi is mainly based on EXgrade PRO software, which provides centralized control of location security using an IP connection to each branch. This is sufficient to raise the alarm for problems that are likely to occur for each branch's central authority managing the access control. Thus, EXgrade PRO is not suitable for protecting sensitive cloud-hosted financial data.

There are numerous branches, facilities, and thus customers spreading worldwide in the banking and finance sector. Thus, there is an urgent need for access control that

(i) Avoids using the central authority for managing the access control since it can be a central point of bottleneck and attack

(ii) Supports flexibility and integration with the systems in each branch

(iii) Benefits from the huge capabilities of blockchain for managing the keys assigned to customers during the session

(iv) Secure open banking from some well-known attacks such as *Account Aggregation*, *Personal Finance Management*, and *Instant Credit Risk*

The abovementioned issues are behind our motivation to propose a blockchain-based key-revocation access control scheme for open banking.

1.2. Main Contributions. To the best of our knowledge, the blockchain is introducing a promising technology for efficient decentralized security solutions [7]. The main contributions of this paper are as follows:

(i) We employed blockchain to propose an efficient decentralized key-revocation access control scheme. The proposed scheme is compatible with the current open banking systems. It can facilitate and secure access to cloud-hosted sensitive financial data and only authorize the nonrevoked customers to benefit from the open banking services such as *Account Aggregation*, *Personal Finance Management*, *Instant Credit Risk*, *Subscription Management*, and *Opening New Accounts*.

(ii) We implemented the contract's functions on the Ethereum platform. Also, we tested the contract code on the Kovan Testnet before deploying it to the Mainnet. Moreover, we implemented the cloud storage based on our private cloud environment built using OpenStack.

(iii) We proved the security of BKR-AC against some well-known attacks such as *Insecure Ids*, *Forced Browsing Past Access Control Checks*, *Path Traversal*, *Client-Side Caching*, and *chosen-text attacks*.

(iv) We evaluated the proposed scheme comprehensively based on time, data exchanged, and policy update time. Moreover, we compared the proposed scheme with two well-known schemes (Certificate Revocation List (CRL) [8, 9] and Online Certificate Status Protocol (OCSP) [3]) in terms of the status response time and the exchanged data.

Finally, no customer can be authorized forever in our model, but its authorization is continuously evaluated, and the assigned key/s can be revoked at any stage.

2. Proposed Workflow

Our solution workflow indicated in Figure 1 introduces five basic entities (user/browser, authentication servers, branch policy, synchronized peer network, synchronized chain of blocks, and cloud storage) and two helpful entities (Bank Web Server and Fraud Detection). Each of these entities has a significant role in the proposed model:

(i) *User/Browser*. It is a great number of bank customers that are requesting access to cloud-hosted financial data throughout the online banking system from various different devices. Each customer willing to access the online banking is assigned a Customer ID (CID). This CID is given by the Bank Web Server and never changes for that customer.

(ii) *Branch Policy*. It is a multiprocess authority in charge of deciding the access control policies by each branch. This authority is identified by Branch ID (BID).

(iii) *Authentication Servers*. In this part, the customer identified by CID provides his username and password and any more credentials to be successfully authenticated. The authentication servers are a *dispatch server*, *database manager server*, and a set of *matching servers*. This step is fulfilled with the help of the Fraud Detection server and the branch policy entity.

(iv) *Synchronized Peer Network*. It is the Ethereum platform using a programmed contract.

(v) *Synchronized Block of Chains*. It is a set of mined blocks that are linked using hashes. Each block keeps a hash of the previous block in the synchronized chain.

(vi) *Cloud Storage*. It is a set of cloud storage servers with huge capabilities to be able to handle numerous data storage and access requests. The cloud storage servers have no right to access control.

The basic steps for an honest customer requesting access to the cloud-hosted financial data are as follows. The customer can request access through the online banking system. First, the Bank Web Server will assign the customer (subject) a unique CID. After that, the authentication servers will either authenticate the CID or not based on the branch policy, Fraud Detection entities, the username, password, and any other credentials provided by the customer. Next, the blockchain contract's functions will be initialized for the authenticated customer identified by CID. Then, a mined block will be added to the synchronized chain of blocks. After that, the customer requesting to be authorized will be done through the synchronized peer network. Finally, only authorized customers can access the cloud-hosted data.

2.1. Expected Vulnerabilities. In general, open banking must have a documented access control policy. If this documentation does not exist, open banking is likely vulnerable. There are some specific access control issues, as follows:

(i) *Insecure Ids*. If an attacker can guess a customer's Id, and there is no adequate validation scheme to ensure that the current customer is not authorized, then, the attacker can access the private information of the guessed Id. Thus, the web applications for open banking should not rely on Id's secrecy for protection.

(ii) *Forced Browsing Past Access Control Checks*. Open banking must require customers to pass certain checks before being granted access to the open banking URL. These checks must not be by-passable.

(iii) *Path Traversal*. This attack tries to access the normally not directly accessible files by providing relative path information as part of a request.

(iv) *Client-Side Caching*. It is the process of accessing the open banking website from shared computers. The browsers frequently cache web pages that attackers can access. The open banking developers should use multiple mechanisms to ensure that customers' browsers do not cash their pages.

Therefore, our scheme must be secure against the attacks mentioned above by ensuring the following:

(i) Validating the correctness of the access control implementation by executing a detailed code review

(ii) Performing penetration testing to determine whether there are problems in the access control scheme

(iii) Extensively testing the access control scheme to ensure there is now a way to bypass

3. Blockchain-Based Key-Revocation Access Control

Our blockchain-based key-revocation access control (BKR-AC) scheme is based on blockchain for building the key-revocation list (KRL) to secure the banking and finance cloud-hosted data. This scheme formally defines the functional keys to achieve the desired features. The functional key consists of all the permissions/services associated with a key during the running session. The customer can be assigned one or more function keys and thereby be given all of the permissions/services related to the key. It should be mentioned that the key-revocation process is accomplished using the contract's functions.

The system model shown in Figure 2 integrates the proposed BKR-AC scheme with the bank branch authentication system. According to Definition 1, it is composed of seven major entities.

Definition 1. A blockchain-based key-revocation access control (BKR-AC) scheme with dynamic revocation based on the key-revocation list requires the customer to be

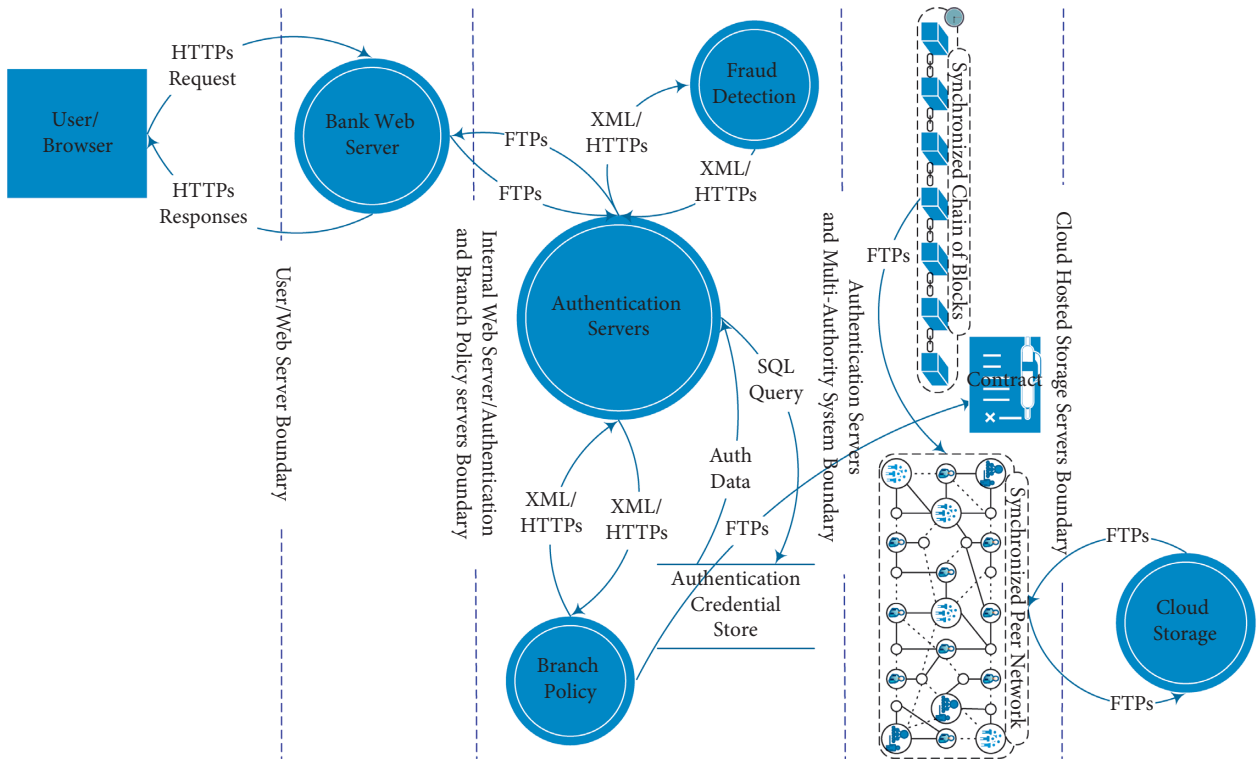

Figure 1: The detailed workflow for our solution to effectively manage the banking customers' access control for the cloud-hosted data.

assigned a nonrevoked functional key to gain the permissions/services associated with the key during the running session. To be a nonrevoked customer, he must be successfully authenticated by the bank's authentication servers and create, deploy, and execute the contract's functions. In addition, the customer can only access the requested cloud-hosted data if he has a nonrevoked functional key based on the latest key-revocation list. The scheme comprises six entities: users requesting access, branch policy, authentication servers, contract functions, a synchronized chain of blocks, a synchronized peer network, and cloud storage.

3.1. Users Requesting Access. Since open banking is revolutionizing the use cases offered by banks and financial institutions, it introduces diverse digital offers. Also, the number of customers is rapidly increasing. Moreover, the customers can use different interfaces to request access to their financial data. Thus, the banks and financial institutions should pay attention to digitizing the customers' experience. In our proposed system, each customer is identified by a unique CID, which represents the customer's identity. The bank branch is responsible for assigning each customer a CID only for the first time joining the open banking system.

3.2. Branch Policy. Each bank branch identified by a unique BID inherits the main bank branch's main policies. Then, each BID assigns these policies to its customers. The most popular and common open banking use cases policies are as follows:

(i) *Account Aggregation*. It is mainly offered by several financial services companies (from different providers). It enables the customers to use an API for getting a detailed overview of their accounts.

(ii) *Personal Finance Management (PFM).* It is the process of giving the customers the tools required for providing a complete overview of their financial situation, for example, showing how much money the customer has left to spend this month.

(iii) *Instant Credit Risk*. It is the process of indicating the likelihood of accepting a loan for a customer before applying. This process is based on using comparison sites for loans and credit cards. Thus, it allows the "buy now pay later" functionalities.

(iv) *Subscription Management*. It detects all the recurring payments from the customer and shows them in one interface. It can be a subscription for a service, a membership, monthly bills, and so on.

(v) *Opening New Accounts.* To open a new account, the banks must get as much information as possible about the customer before authorizing a new account opening. Open banking grants the bank data flow for the customer requesting a new account.

3.3. Authentication Servers. Mainly the authentication server is a software or an application that enables authentication of a subject (entity) attempting to access a specific network. In our scheme, the authentication servers reside in multiple dedicated computers (DB Manager Server, multiple

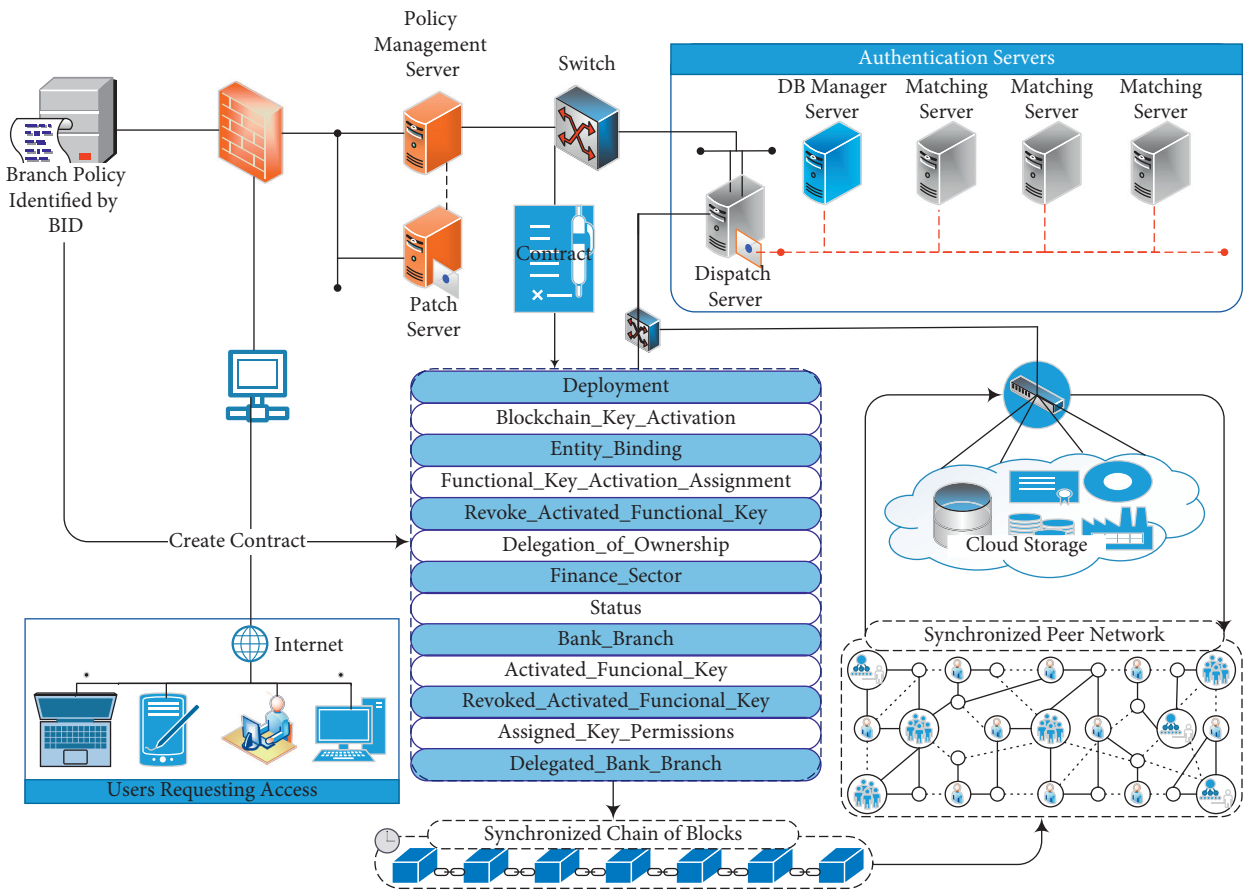

FIGURE 2: The proposed BKR-AC scheme is integrated with the bank branch authentication system. The scheme is composed of six entities: users requesting access, branch policy, authentication servers, contract functions, a synchronized chain of blocks, a synchronized peer network, and cloud storage.

matching servers, and a dispatch server). The authentication process determines whether some subject is who or what they proclaim themselves. The requested attributes from a customer accessing the authentication server are the username, password, security token, and One Time Password (OTP).

We have two scenarios (interactive and noninteractive) for the interaction of the customer identified by CID with the authentication servers based on *Hashcash*. Each of the two scenarios computes the cost of adding a cryptographic puzzle (token). First, the token proves that the customer has successfully done the predefined cryptographic puzzle. Then, the token is verified by the authentication servers.

3.3.1. Customer and Authentication Server Interactive Scenario. In this scenario, a specific authentication server sends a token to the customer. The customer has to solve this token and send its evaluated value to the authentication server. The customer successfully solves the token if the Evaluated Challenge Value (ECV) equals the predefined Challenge Value (CV) by the authentication server, as illustrated in Algorithm 1. The algorithm takes as input the Requested Services (RS) and the random bit-string (Rbs). It should be mentioned that ‖ is the concatenation between two bit-strings.

3.3.2. Customer and Authentication Server Noninteractive Scenario. There is neither a specific authentication server nor a server challenge value in this scenario. However, this scenario has been used in the proof of work consensus algorithm in Ethereum, as shown in Algorithm 2.

3.4. Contract's Functions. This section introduces a detailed description of the contract's functions. We are considering the parameters mentioned in Table 1 for these functions:

(i) Deployment(): it deploys the contract on the Ethereum network and establishes its account. We send an Ethereum transaction containing the code of the compiled smart contract without specifying any recipients. The contract has been compiled first to assure that the web apps and the Ethereum Virtual Machine (EVM) can understand it. The compilation process will produce the Application Binary Interface (ABI). The apps will use ABI to understand the contract and call its functions.

(ii) Blockchain_Key_Activation(B_Name, BID): it is used to assign the authority (organization) responsible for assigning, revoking, updating, and storing the functional keys. In our scheme, the

Input: (RS) and (Rbs)
Output: True/False
(1) The authentication server computes CV = *Challenge*(RS, Lb) // *Lb* is the number of left bits in the random bit-string *Rbs*
(2) The authentication server sends CV to the customer CID
(3) CID searches for Rbs ∋ $Hash(RS||CV||Rbs) \equiv {}_{Lb}0^k$ // k is the number of authentication servers
(4) **if** $(Hash(RS||CV||Rbs) \equiv {}_{Lb}0^k) = True$ **then**
(5) The authentication Server ←CID sends RS, Rbs
(6) The authentication server calculates ECV = $Hash(RS||CV||Rbs) \equiv {}_{Lb}0^k$ /∗ ECV is *True* if $H(RS||Rbs)$ starts with *Lb* bits filled with zero ∗/
(7) **if** ECV = *True* **then**
(8) Return ECV
(9) **else**
(10) Return *False*

Algorithm 1: Interactive scenario.

Input: (RS) and (Rbs)
Output: True/False
(1) The CID calculates Rbs ∋ $Hash(RS||Rbs) \equiv {}_{Lb}0^k$
(2) The CID publishes RS and Rbs
(3) All the authentication servers calculate ECV = $Hash(RS||Rbs)$ /∗ ECV is *True* if $H(RS||Rbs)$ starts with *Lb* bits filled with zero ∗/
(4) **if** ECV = *True* **then**
(5) Return ECV
(6) **else**
(7) Return *False*

Algorithm 2: Noninteractive Scenario.

Table 1: List of considered parameters.

B_Name	Name of the functional key assigning branch
BID	Identity of the functional key assigning branch
CID	Customer's identity
SVT_Session	Functional key start validation time
EVT_Session	Functional key end validation time
RR_Code	Revocation reason code of a revoking a functional key before the EVT_Session

branch policy identified by BID is the functional key assigning authority.

(iii) Entity_Binding(BID, CID): this function verifies that the requesting customer which is identified by CID owns an external account address on Ethereum.

(iv) Functional_Key_Activation_Assignment(BID, Key, Permissions, SVT_Session, EVT_Session): it assigns a functional key to a customer identified by CID and verified by the Entity_-Binding(BID, CID) function.

(v) Revoke_Activated_Functional_Key(BID, RR_Code): the assigning branch BID revokes an assigned functional key before reaching the EVT_Session for a reason identified by RR_Code.

(vi) Delegation_of_Ownership(): it returns the CID of a delegated customer.

(vii) Bank_Branch(): it returns the name (B_Name) of the functional key assigning branch.

(viii) Activated_Functional_Key (CID): it returns a list of the active functional keys for a specific customer (CID).

(ix) Revoked_Activated_Functional_Keys (CID): it returns a list of the revoked functional keys for a specific customer (CID).

(x) Assigned_Key_Permissions/Services(): it returns a list of permissions/services for a specific key.

(xi) Delegated_Bank_Branch(): it returns the BID of a delegated branch.

3.5. Synchronized Chain of Blocks. Each block keeps a hash of the previous block in the synchronized chain. Since the hash is cryptographically derived from the block data, the hash

binds the blocks together. If the attacker changes any previous block, it will invalidate all subsequent blocks because all subsequent hashes will change. All customer transactions are grouped into blocks to ensure that all customers maintain a synchronized state and agree on the exact history of the transactions. The blocks are arranged strictly as each new block references its original block, and the transactions within the blocks are also organized. Each block consists of

(i) timestamp: it is the mining time of the block
(ii) blockNumber: it is the number of the block within the synchronized chain of blocks
(iii) baseFeePerGas: it is the minimum fee for each gas required to add a new transaction in this block
(iv) difficulty: it is the mining effort for that block
(v) mixHash: it is a unique identifier for that block
(vi) parentHash: it is the mixHash for the previous block in the synchronized chain of blocks
(vii) transactions: it is the set of transactions included in that block
(viii) stateRoot: it represents the entire state of the system
(ix) nonce: it is a hash used to prove that the block has gone through a proof of work mechanism (anyone who wants to add new blocks to the chain must solve a difficult puzzle that requires a lot of computing power) by combining it with the mixHash

3.6. Synchronized Peer Network. We implemented our scheme's contract functions on the Ethereum platform using a programmed contract. Before deploying the contract code to the Mainnet (the primary public Ethereum production blockchain), we tested our contract code on the Kovan Testnet. The Kovan Testnet is a proof-of-authority Testnet for running Open Ethereum clients. It uses a small number of nodes to validate transactions and create new blocks—stacking their identity in the process.

3.6.1. Key Generation in Ethereum. The Ethereum public key PK is a point on an elliptic curve. It is generated using Algorithm 3. It consists of two numbers joined together. The numbers composing the Ethereum public key are produced by the multiplication of the *generator point G* with those the for the private key k that is generated by one-way calculations. Thus, anyone having a private key can easily calculate the public key, but he cannot calculate the private key using the public key.

For example, consider that we have the following private key:

$$k = f8f9a2f43c8366ccb0871505030d7b27c0554 d3cc79bcdf41b2805606451f318. \quad (1)$$

Thus, after calculating the public key $\mathrm{PK} = (x, y) = k * G$, where the public key is represented as a point in the elliptic curve,

$$\begin{aligned} x &= 8e144cdef1034dea269874dd09df \\ &\quad b4bee6f3308c84785c82f103454693dae07f, \\ y &= 63b4c38c5e2b0d8528d7fa2f64d45d4a1ede8 \\ &\quad d9af14cdb9478d042f84c32dcd7. \end{aligned} \quad (2)$$

In Ethereum, the public key is represented by 65 bytes (130 hexadecimal characters). The Ethereum network uses the Keccak-256 cryptographic hash function to generate the Ethereum addresses.

3.7. Ethereum Addresses. The addresses in Ethereum are unique identifiers that are generated from the contracts or public keys based on the Keccak-256 cryptographic hash function. Thus, from the previous example, the public key is the concatenation of x and y.

$$\begin{aligned} \mathrm{PK} = {} & 8e144cdef1034dea269874dd09df \\ & b4bee6f3308c84785c82f103454693da \\ & e07f63b4c38c5e2b0d8528d\ldots. \end{aligned} \quad (3)$$

Then, we use the Keccak-256 to find the hash of this public key:

$$\begin{aligned} \mathrm{Keccak}-256(\mathrm{PK}) = {} & 6a5fc542ec614b5da5735268006 \\ & d3c1e\ d826552a\ d17150473d3 \\ & dcf1c086aa0f8. \end{aligned} \quad (4)$$

Then, we only keep the rightmost 20 bytes as our Ethereum address:

$$006d3c1ed826552ad17150473d3dcf1c086aa0f8. \quad (5)$$

It should be mentioned that most of the Ethereum addresses start with 0x, which indicates their hexadecimal encoding.

3.8. Cloud Storage. It is a private cloud environment built using OpenStack by employing three physical servers. The first server in our private cloud environment is the Controller Node, and the second one is the Neutron Node. Each Controller Node and the Neutron Node is 48 cores CPU, 128 GB RAM, and 5 TB disk. The third server in our private cloud environment is the Nova Compute Node, and its configuration is 24 cores CPU, 128 GB RAM, and 2 TB disk. Our scheme employs this private cloud environment to store the customers' information.

4. Security Analysis

Our scheme must be secure against the attacks mentioned in Section 2.1. This section introduces the detailed security analysis for our scheme's entities to prove the BKR-AC capabilities to defend against various attacks. The Ethereum network uses the Keccak-256 hash function.

4.1. Hash Function. The hash function plays an essential role in the security of our scheme and securing its identity over

Input: $(x^3 + ax + b - y^2)\%p$, where a, b are two constant coefficients
Input: A prime number $p \in Z_p$ and an elliptic generation point G
Input: The order of subgroup n and a cofactor h
Output: Private k and public key PK
(1) Private key $k \leftarrow$ Choose a random number from $\{n-1, \ldots, 3, 2, 1\}$
(2) Public key PK$\leftarrow$ Compute $k \times G$
(3) Return private key k and public key PK

Algorithm 3: Key generation in Ethereum.

the Ethereum network. It builds the cost function as cryptographic puzzles.

Definition 2. A hash function $H(x)$ is computationally efficient for mapping a finite length bit-string x to output a fixed-length random bit-string Y. It has three cryptographic properties:

(1) Preimage resistance: it is infeasible to find the input x' for a prespecified output y; i.e., the result for finding any preimage x' (input) such that $H(x') = y$ is not known
(2) 2nd-preimage resistance: it is computationally infeasible to find a 2nd-preimage $x' \neq x$ such that $H(x') = H(x)$
(3) Collision resistance: it is computationally infeasible to find two inputs x, x' such that $H(x) = H(x')$

Therefore, if an attacker guesses a customer's Id, he will not get the same output of the hash function (2nd-preimage resistance). Thus, our scheme is secure against the *Insecure Id's attacks*. Also, if an attacker got the past access control checks (output of the hash function) for the previous browsing, he will not get the same input that produced the guessed past access control checks (preimage resistance). Thus, our BKR-AC is secure against the *Forced Browsing Past Access Control Checks*. Moreover, providing an approximate path will not generate an accurate hash result. Thus, our scheme is secure against the *Path Traversal*. Finally, the collision resistance property of the hash function prevents the *Client-Side Caching* because we could not find the same hash result for two different inputs (two customers' browsers).

4.2. Challenge-Response Authentication. For online banking, the authentication mechanism must be robust. Thus, we employed the signature-based challenge-response authentication as introduced in Algorithms 1 and 2. The verifier (bank branch) should have enough information to verify the validity of the public key contained in the customer's received certificate. Let r_{CID} and t_{CID} represent a random number and timestamp generated by CID. Let $\mathrm{cert}_{\mathrm{CID}}$ be the public key certificate that contains CID's signature. Thus, we can generate the following authentication scenarios:

(1) Unilateral authentication with timestamps:

$$\mathrm{CID} \longrightarrow \mathrm{BID}: \mathrm{cert}_{\mathrm{CID}}, t_{\mathrm{CID}}, \mathrm{BID}, S_{\mathrm{CID}}(t_{\mathrm{CID}}, \mathrm{BID}). \quad (6)$$

Once received, BID verifies that the timestamp is acceptable; the received identifier BID is its own and checks that the signature over CID and BID is correct.

(2) Unilateral authentication with random numbers:

$$\mathrm{CID} \longleftarrow \mathrm{BID}: r_{\mathrm{BID}},$$
$$\mathrm{CID} \longrightarrow \mathrm{BID}: \mathrm{cert}_{\mathrm{CID}}, r_{\mathrm{CID}}, \mathrm{BID}, S_{\mathrm{CID}}(r_{\mathrm{CID}}, r_{\mathrm{BID}}, \mathrm{BID}). \quad (7)$$

BID verifies that the clear-text identifier is its own, and using the $\mathrm{cert}_{\mathrm{CID}}$ for CID, it verifies that the signature of CID is valid over the clear-text random number r_{CID}. Thus, the signed r_{CID} explicitly prevents chosen-text attacks.

(3) Mutual authentication with random numbers:

$$\mathrm{CID} \longleftarrow \mathrm{BID}: r_{\mathrm{BID}},$$
$$\mathrm{CID} \longrightarrow \mathrm{BID}: \mathrm{cert}_{\mathrm{CID}}, r_{\mathrm{CID}}, \mathrm{BID}, S_{\mathrm{CID}}(r_{\mathrm{CID}}, r_{\mathrm{BID}}, \mathrm{BID}),$$
$$\mathrm{CID} \longleftarrow \mathrm{BID}: \mathrm{cert}_{\mathrm{BID}}, \mathrm{CID}, S_{\mathrm{BID}}(r_{\mathrm{BID}}, r_{\mathrm{CID}}, \mathrm{CID}). \quad (8)$$

CID verifies that the clear-text identifier is its own, and using the $\mathrm{cert}_{\mathrm{BID}}$ for BID, it verifies that the signature of BID is valid over the clear-text random number r_{BID}. Thus, the signed r_{BID} explicitly prevents chosen-text attacks.

5. Results and Discussion

To effectively evaluate the performance of the proposed scheme, we investigated the performance in two directions (time and data analysis).

5.1. Time Analysis. We investigated the time analysis into two cases against the number of nonrevoked and revoked keys. Also, we analyzed the branch policy update time against the number of revoked keys. Moreover, we compared the status response time for our scheme with two well-known methods.

5.2. Status Response Time against the Number of Nonrevoked Keys. This part represents the time spent by the bank branch until giving a decision on the revocation status against the number of nonrevoked keys while considering four levels of

concurrent requests (100, 200, 300, and 400) at the same time to the bank branch, as shown in Figure 3.

The results show the following. (i) For the same number of concurrent requests, the average time for the status response time is increasing according to the number of nonrevoked keys, but the increase is not significant. It is about a 4.32 ms difference when increasing the number of nonrevoked keys from 4 to 60 while considering 400 concurrent requests. (ii) For the same number of nonrevoked keys, the average time is increasing according to the number of concurrent requests, but the increase is not significant. It is about a 2.98 ms difference when increasing the number of concurrent requests from 100 to 400 while considering 60 nonrevoked keys.

5.3. Status Response Time against the Number of Revoked Keys. This part represents the time spent by the bank branch until giving a decision on the revocation status against the number of revoked keys while considering four levels of concurrent requests (100, 200, 300, and 400) at the same time to the bank branch, as shown in Figure 4.

The results show the following. (i) For the same number of concurrent requests, the average time for the status response time is increasing according to the number of revoked keys. It is about a 49.67 ms difference when increasing the number of revoked keys from 4 to 60 while considering 400 concurrent requests. (ii) For the same number of revoked keys, the average time is increasing according to the number of concurrent requests. It is about a 24.36 ms difference when increasing the number of concurrent requests from 100 to 400 while considering 60 revoked keys.

5.4. Branch Policy Update Time against the Number of Revoked Keys. This part represents the time spent by the bank branch to update its policies against the number of revoked keys while considering four levels of concurrent requests (100, 200, 300, and 400) at the same time to the bank branch, as shown in Figure 5.

The results show the following. (i) For the same number of concurrent requests, the average time for the policy update is increasing according to the number of revoked keys. It is about a 2.5 ms difference when increasing the number of revoked keys from 4 to 60 while considering 400 concurrent requests. (ii) For the same number of revoked keys, the average time is increasing according to the number of concurrent requests. It is about a 1.58 ms difference when increasing the number of concurrent requests from 100 to 400 while considering 60 revoked keys.

5.5. Status Response Time Comparison with Two Well-Known Schemes. This comparison represents the status response time required to respond with the revocation status for our scheme (BKR-AC) and two well-known methods (CRL and OCSP) against the number of revoked and nonrevoked keys for BKR-AC and the number of revoked and nonrevoked certificates for both CRL and OCSP schemes. We conducted the measures at 400 concurrent requests, as shown in Figure 6.

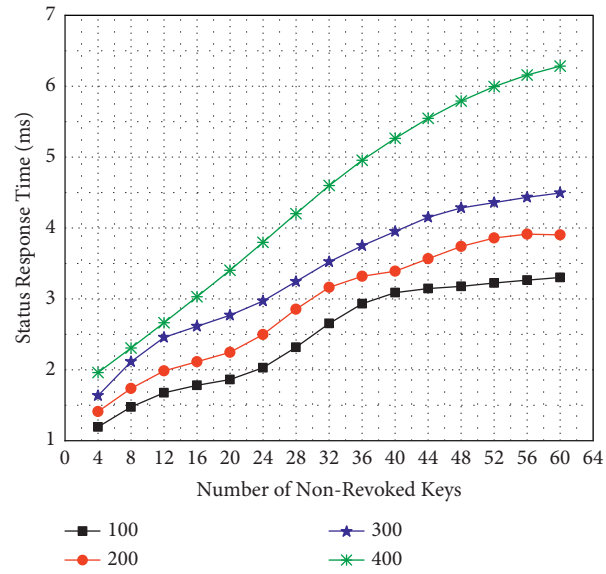

Figure 3: The status response time required to give a decision on the revocation status against the number of nonrevoked keys at four different levels of concurrent requests.

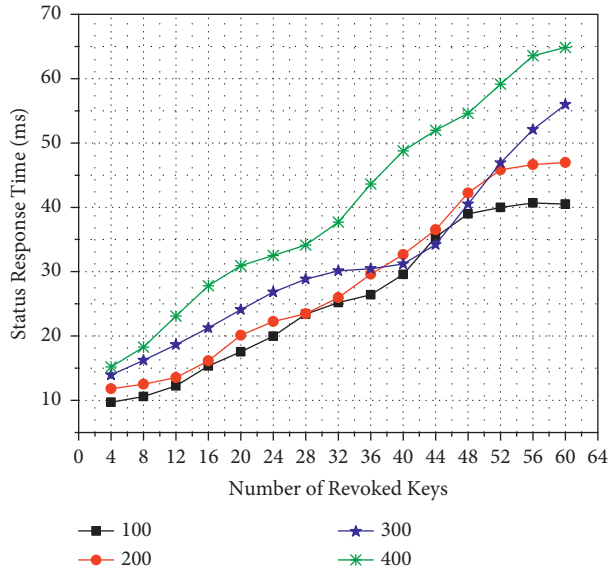

Figure 4: The status response time required to give a decision on the revocation status against the number of revoked keys at four different levels of concurrent requests.

The results show the following. (i) The time spent by BKR-AC in the case of nonrevoked keys is smaller than that for CRL and OCSP. When considering 60 nonrevoked keys/certificates, the BKR-AC (nonrevoked) is 6.3 ms, the CRL (nonrevoked) is 32.9 ms, and OCSP (nonrevoked) is 26.6 ms. (ii) The time spent by BKR-AC in case of revoked keys is greater than that for CRL and OCSP. When considering 60 revoked keys/certificates, the BKR-AC (revoked) is 64.9 ms, the CRL (revoked) is 55.1 ms, and OCSP (revoked) is 50.9 ms. Finally, this indicates that BKR-AC has a rapid response time in the case of nonrevoked keys/certificates.

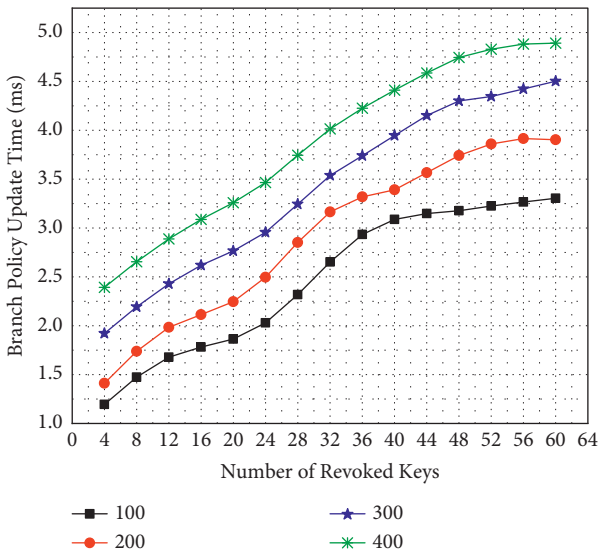

Figure 5: The time required to update the bank branch policy against the number of revoked keys at four different levels of concurrent requests.

But if there exist multiple revoked keys, the response time of BKR-AC is high as it goes through various verification steps to avoid nonrevoking a revoked key.

6. Data Analysis

We investigated the data analysis into two cases against the number of nonrevoked and revoked keys. Also, we compared the amount of exchanged data for our scheme with two well-known methods.

6.1. Transferred Data against the Number of Nonrevoked Keys. This part represents the exchanged data by the bank branch until giving a decision on the revocation status against the number of nonrevoked keys while considering four levels of concurrent requests (100, 200, 300, and 400) at the same time to the bank branch, as shown in Figure 7.

The results show the following. (i) For the same number of concurrent requests, the average exchanged data is increasing according to the number of nonrevoked keys. It is about a 52.02 KB difference when increasing the number of nonrevoked keys from 4 to 60 while considering 400 concurrent requests. (ii) For the same number of nonrevoked keys, the average exchanged data is increasing according to the number of concurrent requests. It is about a 23 KB difference when increasing the number of concurrent requests from 100 to 400 while considering 60 nonrevoked keys.

6.2. Transferred Data against the Number of Revoked Keys. This part represents the exchanged data by the bank branch until giving a decision on the revocation status against the number of revoked keys while considering four levels of concurrent requests (100, 200, 300, and 400) at the same time to the bank branch, as shown in Figure 8.

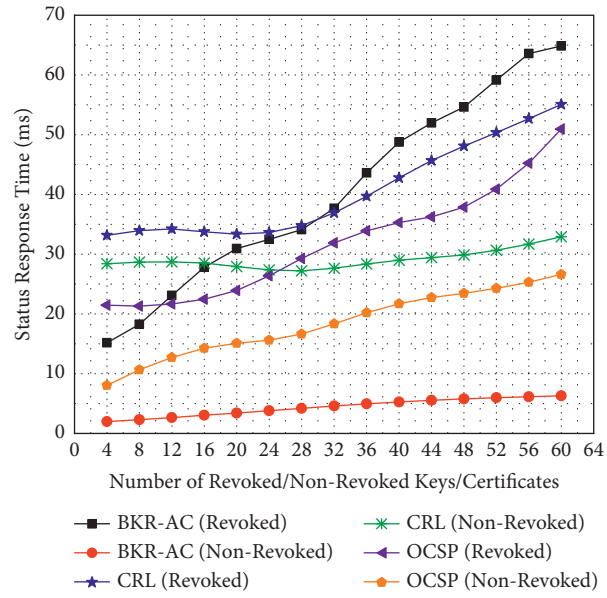

Figure 6: The status response time required to respond with the revocation status for BKR-AC and two well-known schemes (CRL and OCSP) against the number of revoked and nonrevoked keys/certificates at 400 concurrent requests.

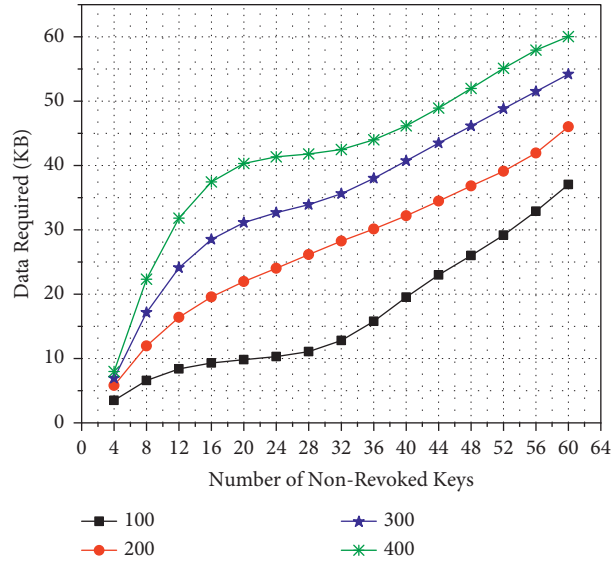

Figure 7: The amount of exchanged data required to give a decision on the revocation status against the number of nonrevoked keys at four different levels of concurrent requests.

The results show the following. (i) For the same number of concurrent requests, the average exchanged data is increasing according to the number of revoked keys. It is about a 52.42 kB difference when increasing the number of revoked keys from 4 to 60 while considering 400 concurrent requests; and (ii) For the same number of revoked keys, the average exchanged data is increasing according to the number of concurrent requests. It is about a 19.22 kB difference when increasing the number of concurrent requests from 100 to 400 while considering 60 revoked keys.

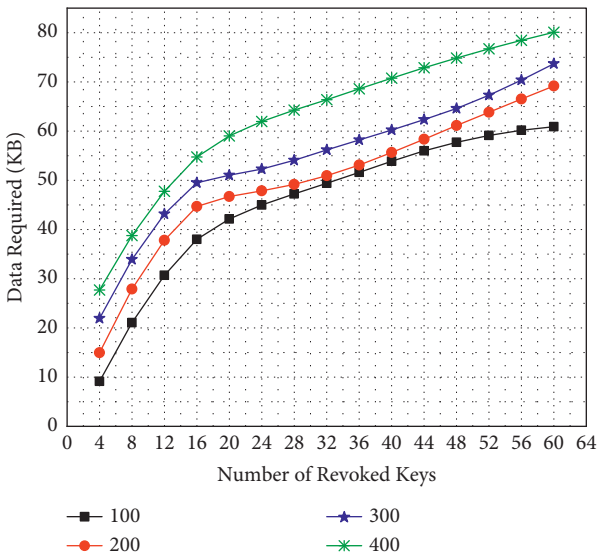

Figure 8: The amount of exchanged data required to give a decision on the revocation status against the number of revoked keys at four different levels of concurrent requests.

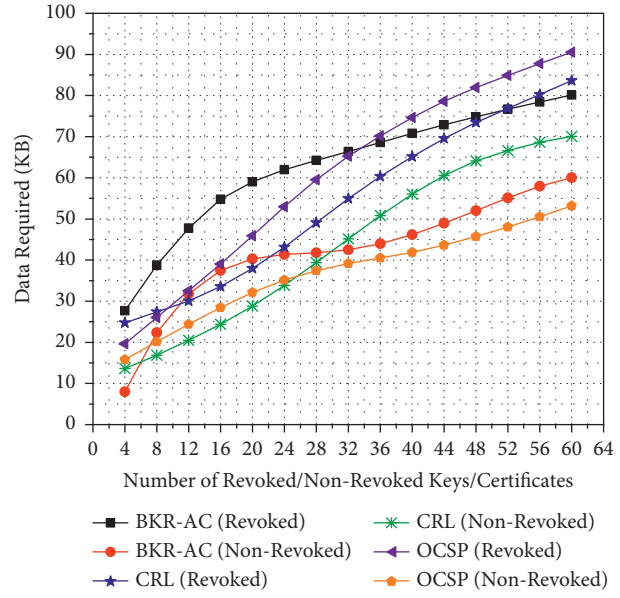

Figure 9: The amount of exchange data required to respond with the revocation status for our scheme (BKR-AC) and two well-known schemes (CRL and OCSP) against the number of revoked and nonrevoked keys/certificates at 400 concurrent requests.

6.3. Transferred Data Comparison with Two Well-Known Schemes. This comparison represents the amount of exchange data required to respond to our scheme's revocation status (BKR-AC) and two well-known schemes (CRL and OCSP) against the number of revoked and nonrevoked keys for BKR-AC and the number of revoked and nonrevoked certificates for both CRL and OCSP schemes. We conducted the measures at 400 concurrent requests, as shown in Figure 9.

The results show the following. (i) The data exchanged by BKR-AC in the case of nonrevoked keys is an average between that for CRL and OCSP. When considering 60 nonrevoked keys/certificates, the BKR-AC (nonrevoked) data is 60 kB, the CRL (nonrevoked) data is 70 kB, and OCSP (nonrevoked) data is 53 kB. (ii) The data exchanged by BKR-AC in case of revoked keys is an average between those for CRL and OCSP. When considering 60 revoked keys/certificates, the BKR-AC (revoked) data is 80 kB, the CRL (revoked) data is 83.6 kB, and OCSP (revoked) data is 90.6 kB. Finally, this indicates that BKR-AC has lower exchanged data than CRL and higher traded data than OCSP in the case of nonrevoked keys/certificates.

7. Related Work and Discussion

Several systems and schemes for managing the certificate/key-revocation process are as follows. This section focuses on some significant famous certificate revocation schemes that are standardized and still used to date. Also, we introduce some well-known blockchain-based certificate revocation schemes. Moreover, we discuss their capabilities and compare them with our proposed scheme.

7.1. Certificate Revocation List and Its Derivatives. The Certificate Revocation List (CRL) [8, 9] is deployed and standardized by the Internet Engineering Task Force (IETF). A Certification Authority (CA) and CRL issuer periodically publish a dated and signed list of revoked certificates. Also, there is an optional Registration Authority (RA) to which the CA delegates some certain management functions. Generally, when revocation status information is provided using CRLs, the CA can work as the CRL issuer. For example, in the X.509 Version 3 certificate, the public key must confirm that the introduced private key is related to the correct remote subject. The confirmation process is done using the hidden certificates in the public key values to subjects. This is asserted using a trusted CA which digitally signs each certificate. Since the signature of the certificate and its timelines can be independently checked by the clients, the certificate distribution process can be done through untrusted communication servers. Moreover, the certificates can be stored in unsecured storage systems.

Once a certificate is issued, it should be valid for use during its whole validity period. However, due to some circumstances, a certificate may become invalid before the end of its validity period. These circumstances include a change of name or association between the subject or the private key becoming suspicious or compromised. Regarding any of these circumstances, the CA must revoke the certificate whatever its validity period. X.509 lets each CA periodically issue a signed and time-stamped list identifying the revoked certificates. This list is made freely available in a public repository, where a serial number identifies each revoked certificate. Thus, the new CRL is issued based on a regular periodic basis. One of the advantages of this revocation scheme is distributing the CRLs by the same means as the certificates themselves through the untrusted servers and communications media. On the other hand, its main

limitation is using untrusted communications and servers. Thus, the revocation is limited to the CRL issue period.

An improvement to CRL is introduced in [10]. It provides a solution to the nonuniform growth in the CRL distribution points. In this scheme, two CRL extensions were used. The first one allows scope statements and is called *CRL Scope Field*. The second one is called *Status Referral Fields*; it updates the partitioning CRL distribution points. The CRL with status referral extension is sent as a response when a subject wants to verify a certificate. This response covers the verified certificate and a pointer to the new location of the CRL related to the certificate in the request.

There is an alternative scheme called Certificate Revocation Status Directory (CRS Directory) [4]. In this scheme, there are two additional fields for the certificate structure. The CA daily sends the signed statements about the status of the certificates issued by a single issuer to the CRS directory. Moreover, there are signed statements to each certificate within its validity period. If a subject requests the status of a certificate, the response will come from the CRS directory with the complete information needed by the subject to verify its request. This scheme downstates the communication overhead between the subjects and the server. On the other hand, it has a considerable communication overhead between the CA and the server.

7.2. Online Certificate Status Protocol and Its Derivatives. The Online Certificate Status Protocol (OCSP) [3] allows applications to determine the revocation status of identified certificates. It is mainly used to achieve the applicable operational requirements to provide more timely revocation information than that introduced by CRLs. Also, it can be used to obtain additional status information. The client of OCSP instantiates a status request to a specific OCSP responder. The acceptance of the certificate is suspended until the responder provides a response. The OCSP is responsible for specifying the exchanged data between an application verifying the status of one or more certificates and the responding server that responds with the certificate's status. It should be mentioned that the server's response needs to be signed by a key.

The certificate status is one of three values, good, revoked, or unknown. The "good" state means that no valid certificate with the same serial number is revoked. The good state does not imply that the certificate was ever issued or that this response came within the certificate's validity period. Thus, the responding server will make additional extensions of the response regarding the current certificate status. The "revoked" state is either temporary revocation or permanent revocation. We can get this status when the responsible CA does not have a record for issuing a certificate with its current serial number. The "unknown" state means that the responding server does not know about the requested certificate. This usually happens when the verification request indicates an unrecognized issuer that this responder does not serve. Finally, the certificate with a revoked state should be rejected, while the certificate with an unknown state means that it is not revoked yet, but this responder can not determine it. Thus, in the unknown state, the client should decide whether to try another source of status information.

The OCSP is a centralized approach that can be considered as a single point of attack or failure for its server. It starts to verify the certificate without verifying the validity of the serial number for the requested certificate. Thus, it can be a victim of the denial of service attack, which can flood it with multiple verification requests with a fake serial number for the requested certificate. Also, the OCSP is timely consuming [11, 12]. It can not be used for offline systems; it works online only. Moreover, the OCSP has a huge privacy risk. Because the responder has detailed information about the verified certificates to each end-user, thus, the responder can track the sites visited by these users [12].

8. Blockchain-Based Certificate/Key-Revocation Schemes

Generally, blockchain has multiple helpful features that can solve the main challenges of the traditional public key infrastructure (PKI) systems. For example, and not as a limitation, blockchain-based solutions are distributed. Thus, it avoids the central point of failure or attacks. Furthermore, blockchain never uses a trusted third party and never needs prior trustworthiness in the system. Various open-source implementations support the variety and effectiveness in building the solutions [13].

There are some recent applications for blockchain in certificate revocation. The authors in [14] considered that certificate revocation as an effective method to prevent potential attacks. The authors employed blockchain to simplify the network structure in the vehicular communication system and maintenance of the Certificate Revocation List. Another application of certificate revocation scheme using blockchain for vehicular communications is introduced in [15]. An efficient contribution for VANETs based on blockchain is introduced in [16] to realize a privacy-preserving authentication model. Moreover, the authors in [17] introduced a conditional privacy-preserving authentication protocol for vehicular Ad Hoc networks based on blockchain. Also, the authors in [18] introduced a decentralized certificate system using the Ethereum blockchain. This scheme provides blockchain certificate services for college students, and the authors in [19] reviewed the traditional centralized PKI systems and proved that they are subjected to security concerns. Therefore, the authors proposed a decentralized PKI infrastructure on the top of the blockchain to solve the security issues of the centralized PKI systems. Another contribution to the PKI using blockchain is introduced in [20]. This scheme is based on the Ethereum blockchain.

The authors in [21] introduced Certcoin. It satisfies the main features of blockchain. It comprises five functions: registration, update, lookup, verification, and revocation. In this scheme, the owner identified with an identity ID posts a transaction to the blockchain to revoke one of its public keys. The revocation in Certcoin is done by the owner only, which has some cons; for example, the owner needs to be online all the time. Furthermore, the owner/person may not have enough knowledge to handle the revocation. On the other

hand, there is an excellent opportunity for the owner to identify the malicious users because they are behaving maliciously. Moreover, Certcoin must check the blockchain revoked certificates list to verify a given certificate status which is very time-consuming.

An extension to Certcoin [21] is introduced by Axon and Goldsmith [22]. The authors presented a privacy-aware blockchain-based PKI, which improves Certcoin privacy-aware. Furthermore, it supports short-term key updates and user-controlled disclosure. In this scheme, users who previously used public keys can be disclosed either by the user or the network majority. But the revocation mechanism is still the same as in Certcoin. The authors in [23] introduced Authcoin. It uses a challenge-response-based validation in the authentication process. It also benefits from the advantages of blockchain storage systems. It is secure against Sybil attacks, but it uses the exact revocation mechanism used in Certcoin.

The authors in [24] introduced a smart contract-based PKI system. It is based on a smart contract on the Ethereum platform and a web-of-trust model. The smart contract cares about the entity publishing a set of attributes, signatures, and even revocations on the blockchain. An Ethereum address represents each entity; a revocation function allows the different entities to revoke their signatures and the key. The process of checking revocation status is done on the blockchain. Another blockchain-based PKI framework is introduced in [25]. It can manage X.509 certificates. The authors extended the standard X.509 certificate to be integrated with blockchain. It depends on the smart contract that acts on two lists: a white list and a black list. The white list is used for the created certificates, and the black list is used for the revoked certificates only.

Moreover, blockchain allows digital information to be distributed but not copied. Thus, blockchain technology created the backbone of a new type of Internet. Blockchain can manage access control in different ways for various specific environments. The authors in [26] guarantee the suitability of access control policies evaluation based on blockchain. They codify attribute-based access control policies as smart contracts and deploy them on a blockchain, thus transforming the policy evaluation process into an entirely distributed smart contract execution. In [27], the authors proposed a novel blockchain-based distributed key management architecture with fog computing to reduce latency and multiblockchains operated in the cloud to achieve cross-domain access. The proposed scheme utilizes blockchain technology to satisfy the decentralization, fine-grained auditability, high scalability, extensibility requirements, and the privacy-preserving principles for hierarchical access control in IoT.

9. Conclusion

The proposed model facilitates and secures access to the cloud-hosted sensitive financial data and open banking services. We introduced a new blockchain-based key-revocation access control scheme that has achieved a set of remarkable goals:

(1) Employing the power and resiliency of blockchain to revoke the customer's keys at shallow status response time

(2) Introducing the compatibility of the proposed scheme with the current open banking systems

(3) Testing the contract's code on Kovan Testnet before deploying it to the Mainnet on the Ethereum platform

(4) Proving the security of BKR-AC against:

  (i) Insecure Id's
  (ii) Forced Browsing Past Access Control Checks
  (iii) Path Traversal
  (iv) Client-Side Caching
  (v) Chosen-text attacks

(5) Conducting comprehensive evaluation based on the status response time and the data transmitted until receiving a revocation decision. Also, we evaluated the policy update time at the bank branch at various numbers of revoked keys

(6) Comparing the proposed scheme with two well-known schemes (CRL and OCSP) regarding the status response time and the exchanged data while considering 400 concurrent requests

The experimental results have indicated that BKR-AC can respond with the status response within a reasonable and acceptable processing time. Considering very rough conditions and huge traffic overhead, the average status response time for 400 concurrent nonrevoked customers is 6.283 ms when considering 60 nonrevoked keys. Furthermore, the average data exchanged until receiving the revocation status for 400 contemporary nonrevoked customers is 60 KB when considering 60 nonrevoked keys. Also, the policy update response time when considering 60 revoked keys and 400 concurrent customers is 4.89 ms. Therefore, the average time and exchanged data rates are small and accepted by both the bank branches and the nonrevoked customers. When considering the number of nonrevoked keys/certificates, the status response time for our scheme (BKR-AC (nonrevoked) = 6.3 ms) is shorter than that of CRL (nonrevoked) = 32.9 ms) and OCSP (OCSP (nonrevoked) = 26.6 ms). On the other hand, when considering the revoked keys/certificates, the status response time for our scheme (BKR-AC (Revoked) = 64.9 ms) is greater than that of CRL (CRL (nonrevoked) = 55.1 ms) and OCSP (OCSP (nonrevoked) = 50.9 ms). When considering the number of nonrevoked keys/certificates, the data exchanged for our scheme (BKR-AC (nonrevoked) = 60 kB) is an average between those of CRL (CRL (nonrevoked) = 70 kB) and OCSP (OCSP (nonrevoked) = 53 kB). Also, when considering the revoked keys/certificates, the data exchanged for our scheme (BKR-AC (revoked) = 80 kB) is an average between those of CRL (CRL (revoked) = 83.6 kB) and OCSP (OCSP (revoked) = 90.6 kB).

Finally, the proposed BKR-AC has a lower response time than CRL and OCSP in nonrevoked keys/certificates and a higher response time in case of revoked keys to avoid

nonrevoking a revoked key. But the data exchanged is an average for BKR-AC between CRL and OCSP, which is still a tiny amount and accepted.

Data Availability

The data used to support the findings of this study are available from the authors upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Y. Liu, W. Tome, L. Zhang et al., "An end-to-end measurement of certificate revocation in the web's PKI," in *Proceedings of the 2015 Internet Measurement Conference, IMC'15*, Association for Computing Machinery, New York, NY, USA, 2015, page 183–196.

[2] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "Online Certificate Status Protocol- OCSP," 1999.

[3] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, *Rfc 6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP*, Internet Engineering Task Force (IETF), Fremont, CA, USA, 2008.

[4] M. Naor and K. Nissim, "Certificate revocation and certificate update," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 561–570, 2000.

[5] I. Ozcelik and S. Anthony, "Cryptorevocate: a cryptographic accumulator based distributed certificate revocation list," in *Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 865–872, Las Vegas, NV, USA, January 2021.

[6] Y. C. E. Adja, B. Hammi, S. Ahmed, and S. Zeadally, "A blockchain-based certificate revocation management and status verification system," *Computers & Security*, vol. 104, Article ID 102209, 2021.

[7] P. J. Taylor, T. Dargahi, D. Ali, R. M. Parizi, and K.-K. R. Choo, "A systematic literature review of blockchain cyber security," *Digital Communications and Networks*, vol. 6, no. 2, pp. 147–156, 2020.

[8] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, and W. Polk, *Rfc 5280: Internet x.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force (IETF), Fremont, CA, USA, 2008.

[9] P. Yee Akayla, *Rfc 6818: Updates to the Internet x.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force (IETF), Fremont, CA, USA, 2013.

[10] ITU-T Recommendation and X, "Information technology–open systems interconnection–the directory: public-key and attribute certificate frameworks," *Series X: Data Networks, Open System Communications and Security*, pp. 1–254, International Telecommunication Union, Geneva, Switzerland, 2016.

[11] E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh, "The case for prefetching and prevalidating TLS server certificates," in *Proceedings of the 19th Annual Network & Distributed System Security Conference (NDSS 2012)*, vol. 12, San Diego, CA, USA, February 2012.

[12] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh, "Towards short-lived certificates," *Web 2.0 Security and Privacy*, Springer, Berlin, Germany, 2012.

[13] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: a state of the art survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 858–880, 2019.

[14] A. Lei, Y. Cao, S. Bao et al., "A blockchain based certificate revocation scheme for vehicular communication systems," *Future Generation Computer Systems*, vol. 110, pp. 892–903, 2020.

[15] H.-G. Kim, "Certificate revocation scheme based on the blockchain for vehicular communications," *Journal of the Korea Society of Computer and Information*, vol. 25, no. 7, pp. 93–101, 2020.

[16] F. Xia, Q. Shi, Q. Xie, and L. Liu, "An efficient privacy-preserving authentication model based on blockchain for vanets," *Journal of Systems Architecture*, vol. 117, Article ID 102158, 2021.

[17] C. Lin, D. He, X. Huang, N. Kumar, and K.-K. R. Choo, "BCPPA: a blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7408–7420, 2021.

[18] R. Xie, Y. Wang, M. Tan et al., "Ethereum-blockchain-based technology of decentralized smart contract certificate system," *IEEE Internet of Things Magazine*, vol. 3, no. 2, pp. 44–50, 2020.

[19] Y. Li, Y. Yu, C. Lou, N. Guizani, and L. Wang, "Decentralized public key infrastructures atop blockchain," *IEEE Network*, vol. 34, no. 6, pp. 133–139, 2020.

[20] A. Rashid, A. Masood, H. Abbas, and Y. Zhang, "Blockchain-based public key infrastructure: a transparent digital certification mechanism for secure communication," *IEEE Network*, vol. 35, no. 5, pp. 220–225, 2021.

[21] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," *IACR Cryptology*, vol. 803, 2014.

[22] L. Axon and M. Goldsmith, "Pb-pki: a privacy-aware blockchain-based PKI," in *Proceedings of the 14th International Conference on Security and Cryptography SECRYPT*, Madrid, Spain, July 2017.

[23] B. Leiding, C. Cap, T. Mundt, and S. Rashidibajgan, "Authcoin: validation and authentication in decentralized networks," 2016, http://arxiv.org/abs/1609.04955.

[24] M. Al-Bassam, "Scpki: A smart contract-based PKI and identity system," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, BCC'17*, pp. 35–40, Association for Computing Machinery, New York, NY, USA, 2017.

[25] A. Yakubov, W. Shbair, A. Wallbom, D. Sanda, and R. State, "A blockchain-based pki management framework," in *Proceedings of the First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) Colocated with IEEE/IFIP NOMS*, pp. 23–27, Tapei, China, 2018.

[26] D. Di Francesco Maesa, P. Mori, and L. Ricci, "A blockchain based approach for the definition of auditable access control systems," *Computers & Security*, vol. 84, pp. 93–119, 2019.

[27] M. Ma, G. Shi, and F. Li, "Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario," *IEEE Access*, vol. 7, pp. 34045–34059, 2019.