Islam *et al.*
EURASIP Journal on Information Security (2023) 2023:7
https://doi.org/10.1186/s13635-023-00142-3

EURASIP Journal on Information Security

RESEARCH Open Access

User authentication and access control to blockchain-based forensic log data

Md. Ezazul Islam[1*], Md. Rafiqul Islam[2], Madhu Chetty[1], Suryani Lim[1] and Mehmood Chadhar[1]

Abstract

For dispute resolution in daily life, tamper-proof data storage and retrieval of log data are important with the incorporation of trustworthy access control for the related users and devices, while giving access to confidential data to the relevant users and maintaining data persistency are two major challenges in information security. This research uses blockchain data structure to maintain data persistency. On the other hand, we propose protocols for the authentication of users (persons and devices) to edge server and edge server to main server. Our proposed framework also provides access to forensic users according to their relevant roles and privilege attributes. For the access control of forensic users, a hybrid attribute and role-based access control (ARBAC) module added with the framework. The proposed framework is composed of an immutable blockchain-based data storage with endpoint authentication and attribute role-based user access control system. We simulate authentication protocols of the framework in AVISPA. Our result analysis shows that several security issues can efficiently be dealt with by the proposed framework.

Keywords Blockchain, Authentication, Access control, Edge network, Security

1 Introduction

In day-to-day life, citizens of a city or community use various utility services such as water, electricity, gas, healthcare, communication, attendance facilities, and so on. They store a huge amount of data in the system. After storing data at any time, such as after few months or few years, dispute may arise from any end (customer or service holder) of the service. On the other hand, logging of data is very much useful for forensic purposes [1], which may help reduce disputes related to real-life issues. Therefore, the primary target of the proposed framework is to store user log data in a tamper-proof data storage and provide forensic access control for dispute resolution.

For any dispute resolution in a community, recording user activity log data is highly required to investigate any dispute event. On the other hand, loss of data can turn the whole system into a catastrophe. The authentication scheme by Sharma et al. [2] lacks tamper-proof data storage. The use of blockchain can make data tamper-proof as it is itself immutable [3]. As data is a highly confidential asset, proper user identification and authentication are indispensable parts of the system. After identifying and authenticating users, granular access control should be used to access the individual data or record. Authors in [4–6] use blockchain technology to facilitate tamper-proof data storage, but Jangirala et al. [5] do not use asymmetric cryptography and lack focus on the integrity of authentication keys. We use asymmetric cryptography for authentication key generation and maintain their integrity along with access control of users. Here we should identify who can access which data or record considering the type of access right (roles and privileges). To ensure controlled access to data, management of user roles and privileges are required together to form an access control framework. So, in this paper, we propose a blockchain-based authentication and access control framework for forensic log data.

*Correspondence:
Md. Ezazul Islam
e.islam@federation.edu.au; ezaz.time@gmail.com
[1] Institute of Innovation, Science and Sustainability, Federation University Australia, Victoria, Australia
[2] Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh

In our proposed framework, citizen-specific edge devices store activity log data in the blockchain data storage, where the blockchain itself is an immutable data structure residing in a cloud storage. That immutability of blockchain data ensures zero possibility of data tampering [7]. We implement that immutable data storage in permissioned blockchain within an edge network as edge computing also increases the scalability by executing computations close to the end devices [8]. In our edge network, we also use blockchain for smart contract. The data storage is in a cloud server, so geographically dispersed users can access them. For accessing, edge device, edge server, corporate user, and forensic user are examples of users of these data. During the accessing of activity log data storage, each of the devices must be identified and authenticated to restrict any intrusion, where intrusion may lead to a severe threat to any citizen's (user) privacy. The access control module holds the user roles, attributes, and policies related to data access. According to these data, the authentication module decides access to proper users based on their user credentials and keys. For authentication purposes, some keys are pre-generated and some are generated at runtime to make the system more secured. To generate authentication keys, we combine secret key (part of user credentials) and existing asymmetric cryptography (RSA). In addition, we apply hash functions for the integrity of the keys. For restricting any unwanted activity, access control and authentication framework work together. Policy level admins approve user creation. In parallel, the framework generates appropriate keys where required. Policy level admins manage user roles, attributes, and policy data. The following sections elaborate on these procedures.

The key contributions of this paper are as follows:

1. Design of a framework of authentication and attribute as well as Role-Based Access Control (ARBAC) has been proposed for securing forensic data.
2. Design of two authentication protocols (users to edge server and edge server to main server) along with their formal analyses and simulations using AVISPA.
3. Evaluation of how an authentication and ARBAC framework can facilitate forensic users to access an immutable data storage according to their roles and privileges.
4. An approach that combines edge network, cloud storage, and blockchain data structure with a newly designed authentication and access control framework, where blockchain contributes as an immutable data structure.

The rest of the paper is organized as follows. First, we review related literature, background concepts, and the problem statement. Then we introduce our proposed framework that includes the system design, the design of the protocols, and formal analyses of the protocols. Next, we discuss the use cases of our proposed framework. Subsequently, we describe the blockchain data storage, consensus and the system functionality interacting blockchain. Then, a discussion on the security aspects of the framework follows. Finally, we conclude the paper.

2 Related works

According to Noura et al. [9], confidentiality, integrity, and authentication of devices (or users) are essential to restrict the misuse of data where data logs can play a role of evidence in digital forensics. Chen et al. [10] describe blockchain as an immutable data structure that facilitates append-only data storage. It makes blockchain a non-alterable data structure. To add any new block to the blockchain ledger, consensus is achieved by proof of work according to Andreas Ellervee et al. [11]. Koutsoupias et al. [12] describe how consensus is achieved in blockchain for Bitcoin (public blockchain platform). Public blockchain platforms allow anonymous users to join the consensus [13], which increases the public exposure of peers and data storage raising the possibility of security concerns. So, we adapt permissioned blockchain (also known as consortium blockchain) as it only allows authenticated users from within the system for consensus, reducing public exposure [14]. In addition, a lightweight consensus can achieve scalability in a permissioned blockchain [4]. So, our proposed framework uses authenticated devices within the system to achieve consensus. Kirli et al. [15] denote smart contract as a self-executing computer program in a blockchain. In our proposed framework, smart contract acts as an interface between users and the blockchain data storage. It works based on some predefined conditions. Only if the conditions met then specific permissions of operations are granted. Samanta et al. [16] stated that introduction of smart contract reduces the need of keeping mediator who can ensure trust among parties; rather, smart contract itself acts as an automated middleman. Smart contract facilitates any transaction to be trustworthy by replacing intermediaries [17]. All kinds of users access data through smart contract. Edge devices through edge network and smart contract push data into data storage. Xu et al. [18] and Uddin [19] et al. describe that edge devices can help reduce the load on the main or cloud server.

In Internet of Things (IoT), authentication, and access control are very important security issues according to Joshi et al. [20] and Ali et al. [21]. As multiple endpoint devices communicate with each other, trust-building is a must for all the IoT devices to ensure trustworthiness

within the distributed blockchain system stated by Yuanyu Zhang et al. [22]. For this trust maintaining process in our framework, there are some security techniques that help to authenticate and apply access control mechanism among devices. For authentication purpose, asymmetric mutual authentication of all the devices can be used which is described by Ali et al. [21], additionally with our new incorporation of secret key generation module. We use public key cryptography (asymmetric cryptography) for basic services to provide enhanced authentication and access control. Both Attribute Based Access Control (ABAC) [23] and Role-Based Access Control (RBAC) are showed by Rajpoot et al. [23]; for better access controlling, we have integrated these two access control concepts for our proposed framework. RBAC holds a pre-defined set of roles of users presented by Hu et al. [24]. Kamboj et al. [25] utilized RBAC-based authentication system for blockchain network, but they used a public blockchain platform. The hierarchy of devices and their variety in an edge network are not suitable for the openness of such a public blockchain platform. On the other hand, ABAC holds the collection of attributes that collectively form policy described by Hu et al. [24]. In our proposed framework, we show how ABAC and RBAC combinedly form ARBAC (Attribute and Role-based Access Control) mechanism that uses attributes to form roles for a variety of devices and users. For these devices and users, our proposed framework generates the authentication keys that provide better features in terms of the integrity of the keys and scalability than Jangirala et al. [5].

3 System design

To generate authentication keys, we combine password as a secret key with asymmetric key. We further ensure the integrity of the authentication keys using cryptographic hash functions. This section describes various authentication key generation processes and presents formal analyses of the protocols.

3.1 System architecture

Communications between edge devices and edge servers take place through the smart contract in edge network, when edge servers and smart contract work as an interface between edge devices and main server [26]. Blockchain facilitates smart contract as its inherent technology [27]. On the other hand, a forensic user also requires an edge server to communicate with the main server. Overview of the situations depicted in Fig. 1, which exhibits the whole scenario. For the whole system to be functional, all the users must be preregistered in the ARBAC repository. Each edge device is assigned against its regional edge server; each time an edge device tries to connect with edge server it requires to pass the authentication phase; in the same way if edge server wants access to main server, it is given access only if authentication is passed. On the other side, a forensic user is required to go through authentication process if s/he or it wants access to main server, for this first level authentication is done by edge server then edge server passes it to main server for the next level authentication. Each area has a relevant edge server, so the second level authentication is important as it also helps the main server identify the area for which the forensic user requests data access.

3.2 Assumptions and notations

Edge server (ES) contains users' data. Edge server will be location oriented. As such, each region of a city has an edge server. There are n edge servers of n regions of a city. After processing data, edge server sends them to corporate/main server for permanent storing. There will be a separate blockchain ledger for each service-oriented data to modularize the storage for reducing data access and block verification time. For example, there will be separate blockchain ledgers for phone logs, water/electricity bills, and so on.

1. All the devices/users must be registered users. Any user can be registered to one or more server(s).
2. Users must be authenticated by respective edge server(s) through smart contract provided by blockchain.
3. Each edge server must be registered server as a user to cloud or main server. On the other hand, forensic users must be registered to the main server. All users of the main server will be authenticated using separate authentication process.

3.3 User authentication to edge server

In this process, each user or device is a registered user. The system assigns or suggests ID (id) to each user and s/he chooses a password (pwd) according to the server's suggestion. In this section, server represents the edge server. The respective server stores ID and password in hashed form. Each server has its id and a registered user gets it. A user must pass through an authentication process to get service from the server.

1. The user sends her/his id, pwd, and id of the server from which s/he wants to get service. For example, if the i_{th} user wants service from the k_{th} server, the user then sends the following credentials to the server:

$$id_{ik}, id_k, and pwd_{ik}$$

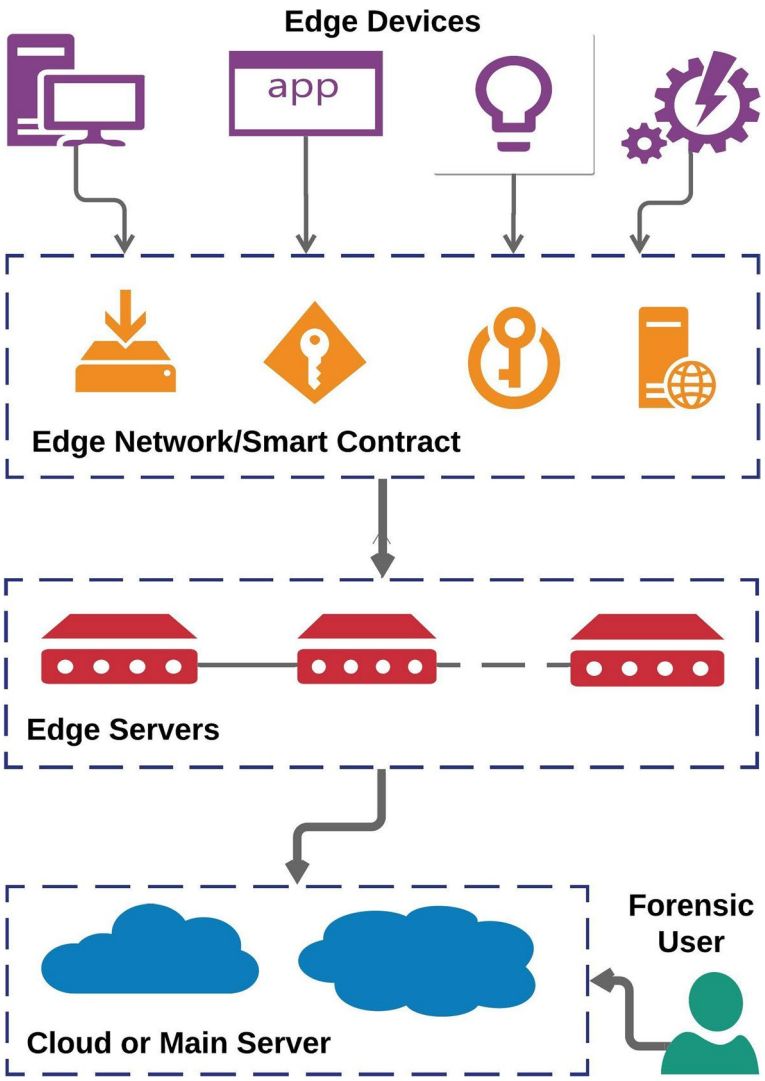

Fig. 1 Overview of the smart architecture

More detail on the above credentials is available in Table 1.

2. After getting the credentials from the user, the server generates a secret number as below:

$$SN_{ik} = H(id_{ik}||pwd_{ik}||id_k||TS_i||R_k)$$

Timestamp means date and time when the number is computed.

3. Next, the server computes a session key as:

$$K_{s-ik} = H(id_{ik}||SN_{ik}||id_k||N_{ik})$$

4. The server generates an authentication key for the user as follows:

$$K_{au-ik} = H(id_{ik}||K_{s-ik}||TS_i + 1)$$

Table 1 Notation and respective description for user authentication to edge server

Symbol	Description
H	Hash function, such as SHA-2
id_{ik}	ID of the i_{th} user of k_{th} edge server
id_k	ID of the k_{th} Edge server
pwd_{ik}	Password of i_{th} user of k_{th} edge server
TS_i	Timestamp of the server for i_{th} user
R_k	Random Number of k_{th} edge server
N_{ik}	Nonce for i_{th} user of k_{th} Edge Server
TS_{ess}	Timestamp when computing ESS (Edge Server's Secret)
U_{ik}	The i_{th} user of the k_{th} edge server
ES_k	The k_{th} Edge server

$TS_i + 1$ denotes next timestamp (date and time) for the i_{th} user.

5. The server sends the authentication key, $K_{au\text{-}ik}$ to the user.
6. Next, the user submits the $K_{au\text{-}ik}$ to the server to get the service.

This protocol seems a bit time-consuming due to a variety of computations, but distributed nature of edge servers will reduce the execution load by decentralizing the execution. We conduct a formal analysis of the above protocol as follows.

3.3.1 Formal analysis of the protocol

1 The protocol have three messages between the user and the server. We have used the notations used in [28] according to GNY logic.

Message 1. $U_{ik} \rightarrow ES_k : id_{ik}, pwd_{ik}, id_k$
Message 2. $ES_k \rightarrow U_{ik} : H(id_{ik}||K_{s-ik}||TS_i + 1)$
Message 3. $U_{ik} \rightarrow ES_k : K_{au-ik}$

2 The parser algorithm would describe the protocol as follows:

(a) $U_{ik} \ni id_{ik}, pwd_{ik}, id_k$
(b) $ES_k \triangleleft : *id_{ik}, *pwd_{ik}, *id_k$
(c) $ES_k \ni id_{ik}, pwd_{ik}, id_k$
(d) $ES_k \ni H(X)$
(e) $ES_k \ni H(X')$
(f) $U_{ik} \triangleleft : *K_{au-k}$
(g) $U_{ik} \ni K_{au-ik}$

3 Protocol Analysis: It can be assumed that the following holds at the starting of every run of the protocol.

(a) $U_{ik} \ni id_{ik}, pwd_{ik}, id_k$
(b) $ES_k| \equiv U_{ik}$
(c) $ES_k \ni H(id_k||pwd_k||id_k||TS_2||R_k)$
(d) $ES_k| \equiv \#(TS_i)$
(e) $ES_k| \equiv \#(R_k)$
(f) $ES_k| \equiv \#(SN_k)$
(g) $ES_k \ni H(id_{ik}||SN_{ik}||id_k||N_{ik})$
(h) $ES_k| \equiv \#(N_{ik})$
(i) $ES_k \ni H(id_{ik}||K_{s-ik}||TS_i + 1)$
(j) $ES_k| \equiv \#(k_{s,ik})$
(k) $ES_k| \equiv \#(TS_i + 1)$
(l) $ES_k| \equiv \#(K_{au-ik})$
(m) $U_{ik}| \equiv ES_k| \equiv K_{au-ik}$
(n) $ESk| \equiv K_{au-ik}$

3.4 Authentication key generation and storing process

The $K_{au,\,ik}$ is the final product of this phase, where this key is assigned to i_{th} user of k_{th} server by authentication key table illustrated in Fig. 2. This key is created and assigned during user creation on request of user. In Fig. 3, user requests access to edge server with proper credential and response is given upon successful authentication key generation by edge server.

When a user wants to get access to an edge server, the server computes authentication key using steps 1 to 4 (in sub-section:user authentication to edge server). If the currently computed key is equal to the key stored in the table, the user is authenticated and the response will be a success message. Otherwise the response will be a failure message.

3.5 Access token for corporate user

Now we explain how a user to the main server is authenticated. As we mentioned, the users of the main server are edge servers, corporate users, and forensic users. A corporate user can get access to the edge server as well as main server. The edge server uses special tag to the corporate user, which is generated as follows:

$$TAG_{ck} = H(id_k||pwd_{ck}||K_{pr,es}||TS_{tag})$$

where pwd_{ck} is the password of a corporate user of the k_{th} edge server and $K_{pr,\,es}$ is the private key of the edge server.

When a corporate user wants access to main server s/he must go through edge server. The server authenticates the user and checks its tag also. If her/his tag is also okay, his access will render to the main server by sending the encrypted tag. The tag is encrypted by the private key of the edge server. The main server authenticates the edge server and decrypts the tag. Next, it gives access to the corporate user.

3.5.1 Operations for access token for corporate user

The four parties related to the phase for communication between corporate user and main server are depicted in Fig. 4, which shows corporate user is the requester to get access to main server with the predefined user credentials. The same figure shows ARBAC repository stores user credentials.

3.5.2 Simulation of Fig. 4 using AVISPA

AVISPA is a cryptographic protocol verification tool based on HLPSL language [29] which is used in our work to verify the goals mentioned below for the authentication of TAG_{ck}.

1 Authentication of TAG_{ck} when sent from ES to U
2 Authentication of TAG_{ck} when sent from U to MS
3 No attack by intruder

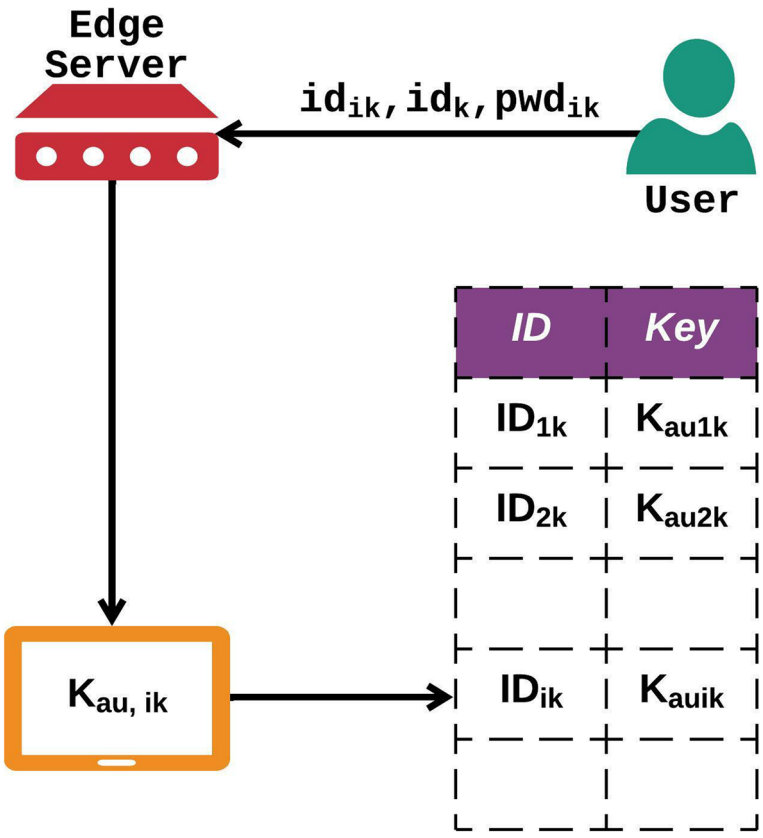

Fig. 2 Authentication key generation and storing process

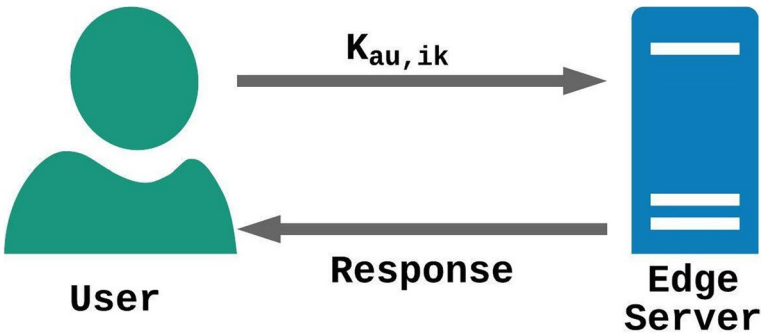

Fig. 3 Dialogue between user and edge server

We depict the whole authentication process in Fig. 4. Both Figs. 5 and 6 show verification result of goals of the process, where these figures show that AVISPA successfully verifies authentication of TAG_{ck} when transmitted among edge server(ES), user(U), and main server(MS). No intruder can impersonate the edge server, user, or main server as the verification results prove that the protocol depicted in Fig. 4 ensures authentication of TAG_{ck}. When TAG_{ck} is encrypted by $K_{pr, es}$ and sent to user, sender of TAG_{ck} is authenticated by the receiver. On the other hand, user sends the encrypted TAG_{ck} to main server by encrypting the message using private key of sender and main server uses public key of sender for authentication of the message. So, in both situations, intruder cannot impersonate any user as authentication is maintained using public-private keys of the users.

On the other hand, when abovementioned public-private keys are not used to send and receive TAG_{ck} among the users, authentication cannot be

Corporate User	Edge Server	Main Server	ARBAC Repository
1 Request Access (id_k,pwd_{ck})	1.1 Request user credentials		1.1.1 Return user credentials
	1.2 Checks user validity		1.2.1 Returns user credentials
	1.3 Generates TAG_{ck}		1.3.1 Stores TAG_{ck}
	1.4 Encrypts TAGck		
	1.5 Sends Encrypted TAG_{ck} to user		
1.5.1 Receives Encrypted TAG_{ck}			
2 Requests using Encrypted TAG_{ck}		2.1 Decrypts Encrypted TAG_{ck} using $K_{pb,\,es}$	
		2.2 Access granted	

Fig. 4 Corporate user gains access to main server through edge server

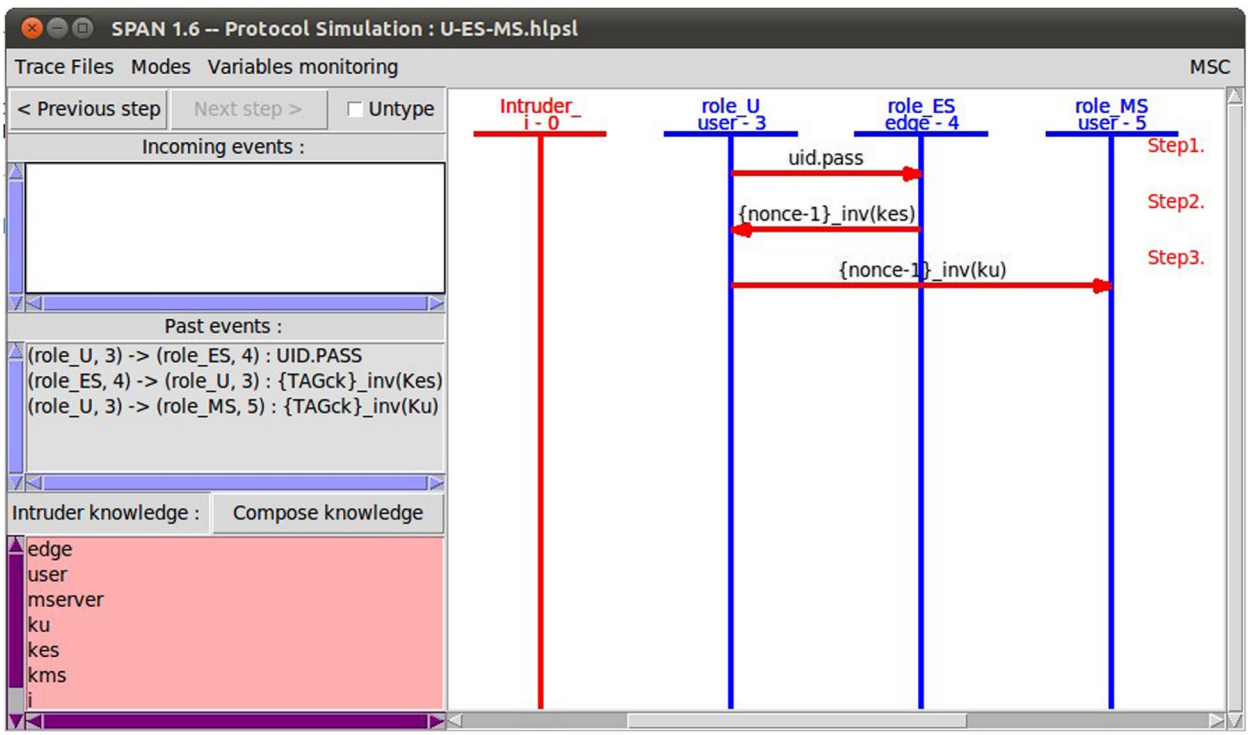

Fig. 5 Simulation of protocol in Fig. 4 using AVISPA—no attack by intruder

maintained. In this situation, intruder takes over and impersonates user and edge server which is depicted in Fig. 7. Due to authentication failure, the protocol is proved to be unsafe by AVISPA verification tool which we depict in Fig. 8.

3.5.3 Computation time for access token for corporate user

Key generation time shown in Table 2 is recorded using Intel Core i3 2GHz CPU with RAM of 4 GB. For this experimental phase: corporate user, edge server, main server, and ARBAC Repository were setup within a single

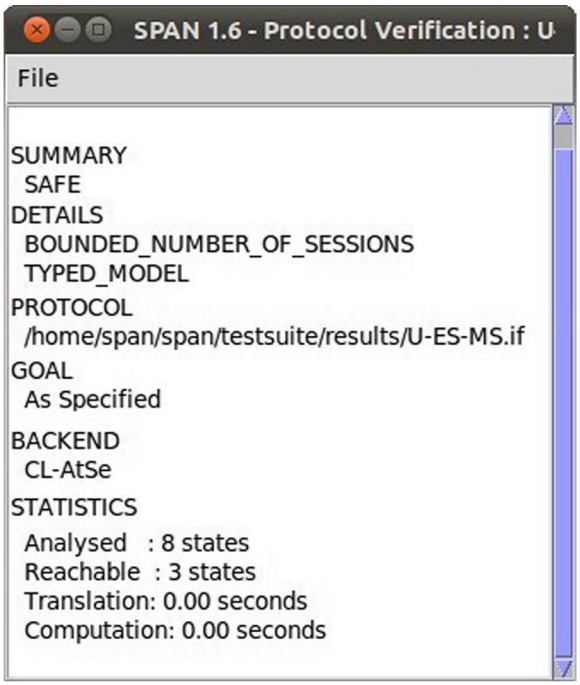

Fig. 6 Simulation of protocol in Fig. 4 using AVISPA—when protocol is safe

system. Secret keys are generated using PHP as a scripting language in Apache server. Time shown in the table is recorded in milliseconds. For this, PHP benchmark tool is used to measure execution time.

To generate TAG_{ck} which is mentioned in Table 2, it requires private key of the edge server. This private key is a part of public key-based cryptosystem. Private and public key are the examples of asymmetric key cryptosystem [30]. For this phase of experiment, 2048 bits RSA asymmetric keys are created using OpenSSL [31].

In Fig. 9, comparison of computational times shows that encryption of TAG_{ck} takes the highest time. Decryption of the tag takes around 0.16698 ms. On the other hand, creation of TAG_{ck} is the fastest among all of the three, which is also shown in Table 2.

3.6 Authentication of edge server by main server

Each edge server and forensic user are the registered user of the main server. The authentication process is as follows.

1 The main or cloud server generates its secret as follows:

$$MSS = H(id_{es,k}||id_{ms}||pwd_{es,k}||TS_{mss})$$

2 Next the server computes a token for the concerned edge server using the process:

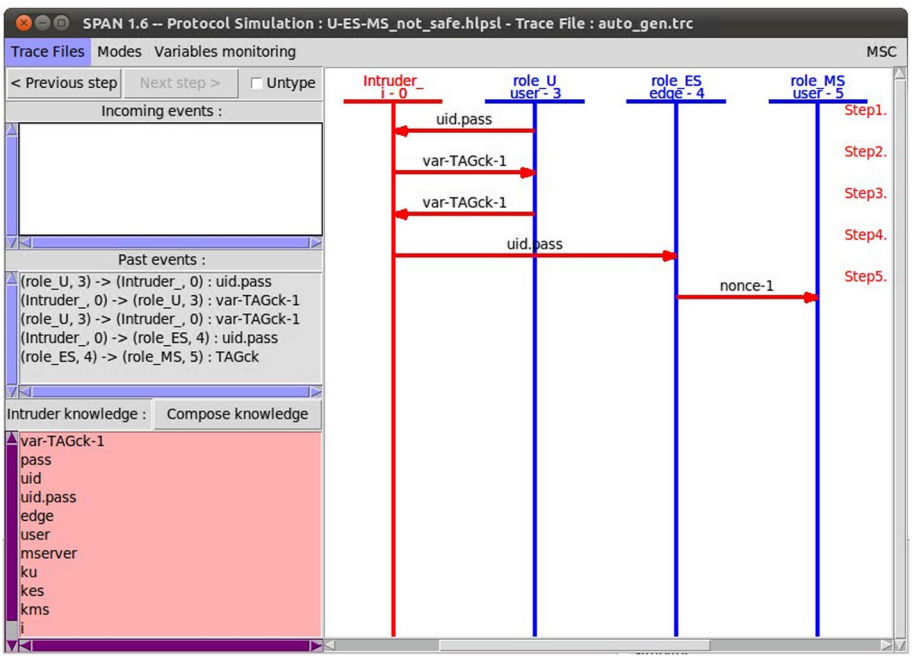

Fig. 7 Simulation of protocol in Fig. 4 using AVISPA—when authentication is not maintained

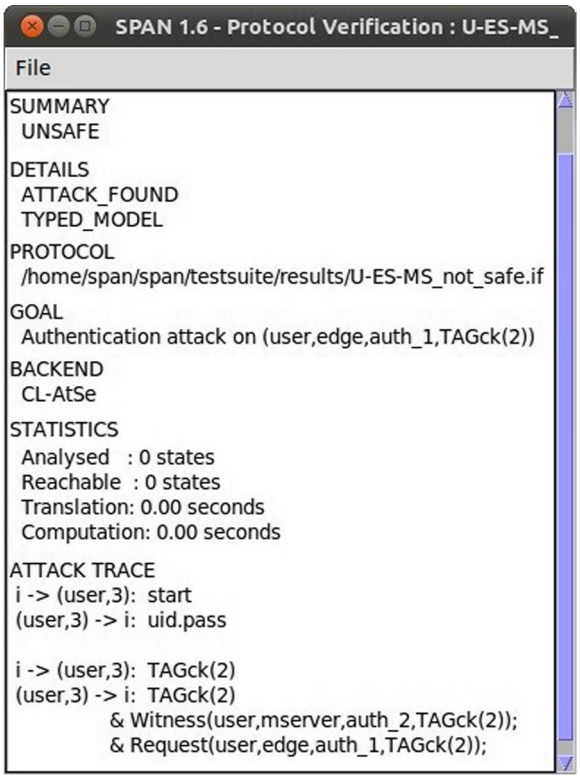

Fig. 8 Simulation of protocol in Fig. 4 using AVISPA—when authentication is not maintained

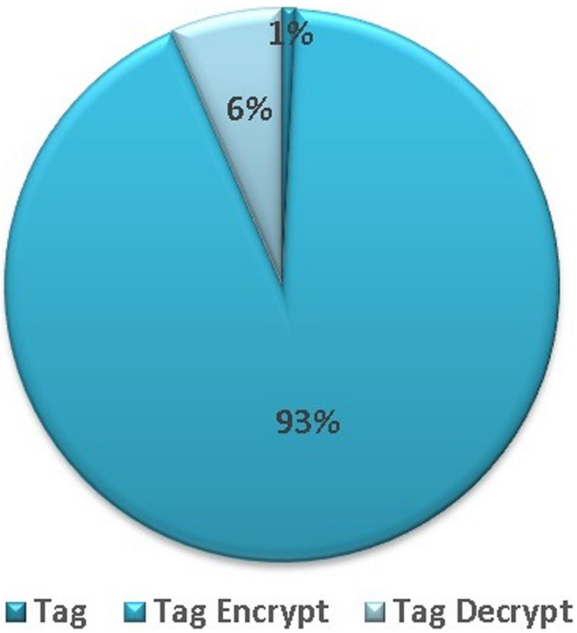

Fig. 9 Computational time to generate secrets for corporate user, edge server and main server

Table 2 Required Computational Time to Generate Secrets for communication among corporate user, edge server and main server, time in milliseconds

TAGck	0.019884109
Tag Encrypt	2.378845215
Tag Decrypt	0.166988373

$$ET = E(K_{pr,ms}, [MSS||TS_{ET}])$$

$K_{\text{pr, ms}}$ is a private key of main server. TS_{ET} is a timestamp.

3 Next, the server encrypts the token (ET) and sends it to the edge server:

$$C - ET = E(K_{pb,es}[ET||N1])$$

$K_{\text{pb, es}}$ is a public key of edge server and N1 is a nonce. After getting the cipher text (C-ET), the edge server opens it by decrypting it using its private key as $\text{D}(\text{K}_{\text{pr, es}}, [\text{C-ET}])$. After deciphering the token, the edge server gets ET and N1.

4 Next, the edge server sends access request as follows:

$$ER = E(K_{pb,ms}, [id_{es}||ET||N1])$$

The main server decrypts the ER (encrypted request) to the server using ET and N1. After finding ET and N1, which were sent by the main server, the respective edge server is authenticated.

All communications required for getting access to main server are depicted in Fig. 10, where at the first step ID and password of edge server are required for authentication purpose. Upon authentication, C-ET which is an access token required for edge server is issued and sent back for the next step to be accomplished. Having the C-ET, edge server issues and sends an access request (ER) to the main server. If verification of ER by main server is successfully completed, then the edge server is given access permission.

3.6.1 Operations for authentication of edge server by main server

The four parties related to the phase for communication between edge and main servers are depicted in Fig. 11, which shows edge server is the requester to get access to main server. Admin is the creator of edge server and assigns user credentials, where ARBAC repository stores user credentials. We conduct a formal analysis of this phase of the communication as follows.

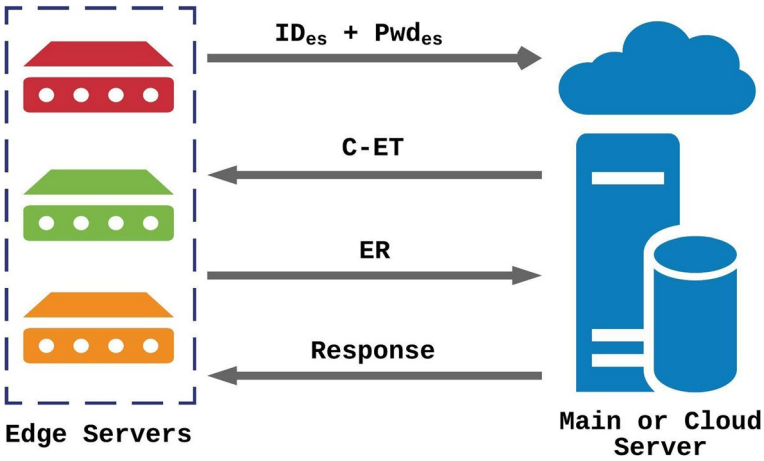

Fig. 10 Overview of dialogue between edge server and main server

Edge Server	Admin	Main Server	ARBAC Repository
1 Request user creation	1.1 Creates user		1.1.1 Stores user credentials
2 Request Access		2.1 Request user	2.1.1 Return user
		2.2 Checks user	2.2.1 Returns user
		2.3 Computes: MSS ET C-ET	
		2.6 Sends back C-ET	
2.6.1 Receives C-ET			
3 Computes ER using			
3.1 Sends ER		3.1.1 Decrypts ER	
		3.1.2 Finds ET and N1	
		3.1.3 Verify ET and	
		3.1.4 Access granted	

Fig. 11 Dialogue between edge server and main server

Table 3 Notation and respective description for edge server authentication to main server

Symbol	Description
$id_{es,k}$	ID of the k_{th} edge server
id_{ms}	ID of the main server
$pwd_{es,k}$	Password of k_{th} edge server
TS_{mss}	Timestamp when computing MSS (Main Server's Secret)
$K_{pr,ms}$	Private key of main server
$K_{pb,ms}$	Public key of main server
$K_{pr,es}$	Private key of edge server
$K_{pb,es}$	Public key of edge server
$\|\|$	Concatenation
N1	Nonce

3.6.2 Formal analysis of the protocol

1 The protocol has messages between the edge server and main server as follows. We have used the notations used in [28] according to GNY logic (see Table 3 for further notation detail).

Message 1. $ES_k \rightarrow MS : id_{es,k}, pwd_{es,k}, id_{ms}$
Message 2. $MS \rightarrow ES_k : \{ET, N1\}K_{pb,es}$
Message 3. $ES_k \rightarrow MS : \{id_{es,k}, ET, N1\}K_{pb,ms}$

2 The parser produces the following output:

(a) $ES_k \ni id_{es,k}, pwd_{es,k}, id_{ms}$

(b) $MS\triangleleft : *id_{es,k}, *pwd_{es,k}, *id_{ms}$
(c) $MS \ni H(x)$
(d) $MS \ni K_{pr,ms}$
(e) $MS \ni \{MSS, TS_{ET}\}K_{pr,ms}$
(f) $MS \ni N1$
(g) $ES_k\triangleleft : *\{\{MSS, TS_{ET}\}K_{pr,ms}, N1\}K_{pb,es}$
(h) $ES_k\triangleleft : *N1$
(i) $MS\triangleleft : *\{id_{es}, ET, N1\}K_{pb,ms}$
(j) $MS \ni ER$

3 Protocol Analysis: We assume that the following holds at the beginning of each run of the protocol.

(a) $ES_k \ni id_{es,k}, pwd_{es,k}, id_{ms}$
(b) $MS| \equiv id_{es,k}$
(c) $MS \ni H(id_{es,k}||id_{ms}||pwd_{es,k}||TS_{ms})$
(d) $MS| \equiv \#(TS_{mss})$
(e) $MS \ni K_{pr,ms}$
(f) $MS| \equiv \#(TS_{ET})$
(g) $MS| \equiv \xrightarrow{+K_{pb,es}} ES$
(h) $MS \ni ET$
(i) $MS| \equiv \#(N1)$
(j) $ES| \equiv \xrightarrow{+K_{pb,ms}} MS$
(k) $ES| \equiv N1$
(l) $ES| \equiv C - ET$
(m) $ES \ni C - ET$
(n) $MS \ni ER$
(o) $MS| \equiv ER$
(p) $ES| \equiv MS| \equiv ER$

Further verification of the above protocol is as follows.

3.6.3 Simulation with respect to Fig. 11 using AVISPA

Verification of the protocol depicted in Fig. 11 is conducted using HLPSL language in AVISPA. In the mentioned protocol, authentication of users who are sharing the C-ET over the network is verified. Below mentioned goals are achieved using AVISPA when main server passes C-ET to edge server.

1 Authentication of C-ET proves that authentication of ER is possible when encrypting using private key of main server.
2 No attack by intruder.

When main server sends C-ET to edge server, $K_{\text{pr,ms}}$ and $K_{\text{pb,es}}$ are used for authentication of sender and receiver. Using HLPSL language in AVISPA, this scenario has been simulated. It proves that authentication of sender and receiver are maintained; thus, no intruder can impersonate any of the users. Figure 12 shows intruder cannot impersonate user due to authentication using $K_{\text{pr,ms}}$ and $K_{\text{pb,es}}$. So, the protocol is safe with respect to authentication of users which we depict in Fig. 13.

On the other hand, unsafe state is depicted collectively in Figs. 14 and 15 when $K_{\text{pr,ms}}$ and $K_{\text{pb,es}}$ are not used to authenticate the users. These figures show that when authentication procedure is intentionally not maintained

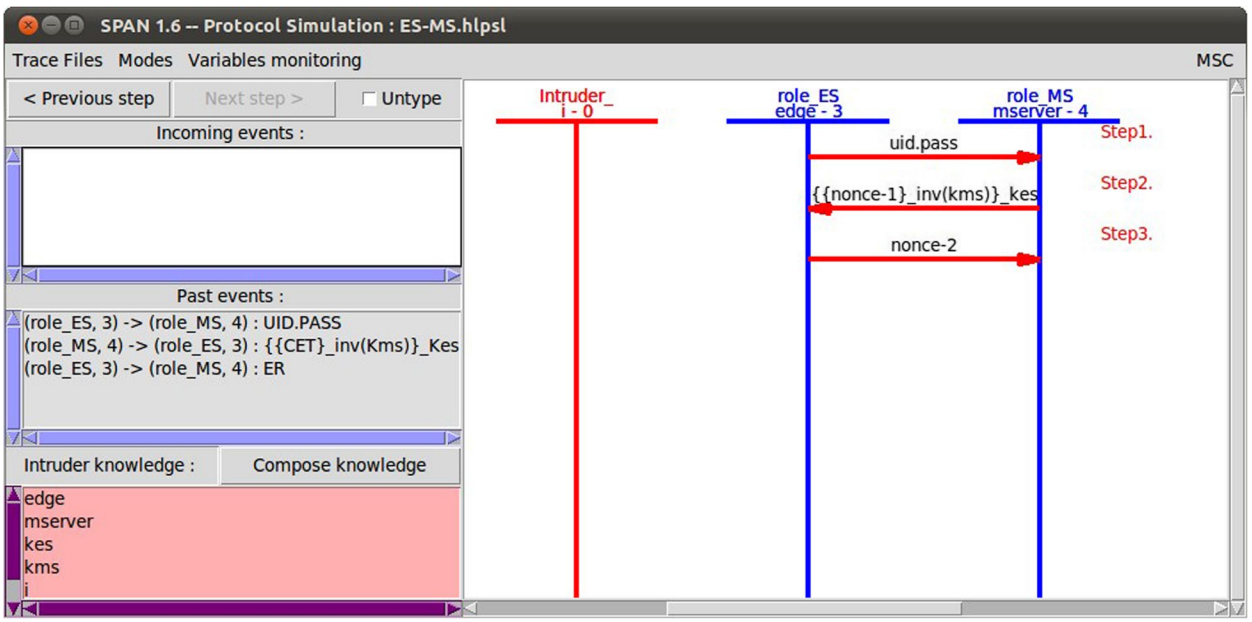

Fig. 12 Simulation of protocol in Fig. 11 using AVISPA—no attack by intruder

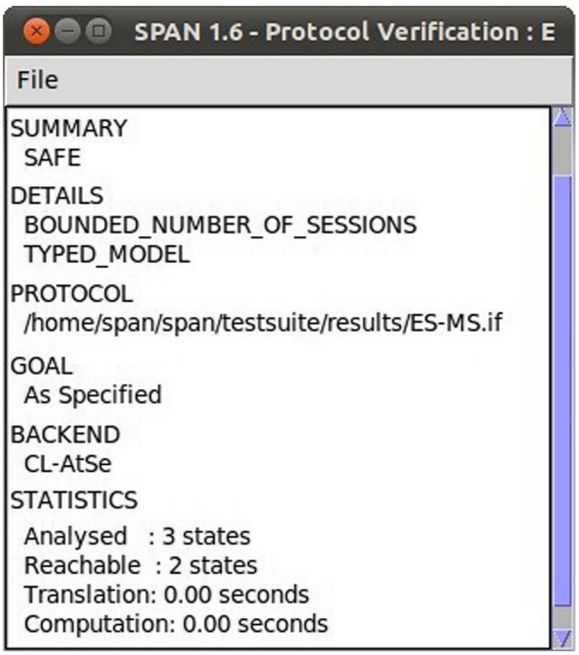

Fig. 13 Simulation of protocol in Fig. 11 using AVISPA—when protocol is safe

in AVISPA, intruder can impersonate users and the protocol is proved to be unsafe.

3.6.4 Computational time for authentication of edge server by main server

Key generation time shown in Table 4 is recorded in Intel Core i3 2GHz CPU with RAM of 4 GB. For this experimental phase: edge server, admin, main server and ARBAC Repository were setup within a single system. In that setup, secret keys are generated using PHP as a scripting language in Apache server. We use PHP benchmark tool for recording execution time. During generating all the secret keys mentioned in Table 4, both public and private keys of edge and main server are required. These keys are part of public key-based cryptosystem, in which asymmetric keys are used. For this phase, we use 2048 bits of RSA private and public key. These keys are created by admin when new user creation is done and stored into ARBAC repository.

Figure 16 shows comparison among execution time required for generating MSS, ET, C-ET and ER. Among them, ET is the slowest when created, that takes around 78% of total time required for all four. Among the four, the fastest generated secret is the MSS, which takes almost 0% (around 0.00758 ms) of time. On the other hand, C-ET and ER require 15% and 7% of total time respectively.

3.7 Asymmetric key generation by admin

Figure 17 shows the admin-level users have the responsibility to assign public and private keys required during each user creation. Where each RSA-based key is created with the help of OpenSSL. Each user gets a pair of keys, including a public key that is known to all, but the private key is only exposed to the respective user. In all phases of the proposed framework, for most of the secret key generations, public-private keys are required for authentication and verification purposes of the users.

After authentication of the users, ARBAC can be activated. Here we can use role-based and attribute-based access control for data in forensic use. This hybrid access control module with roles and attributes will be called ARBAC for the proposed framework. The role and attribute-based system is consisted of the following steps depicted in Fig. 18 to perform operation on the request object by the user. Here object refers to the resource requested by user. On the other hand, subject is the user who requests access to object. Providing access to proper object, all the repositories such as policy, subject attribute, and object attribute repositories are used.

1 Subject seeks access to object.
2 Access control module evaluates: rules, conditions, subject attributes, object attributes, and environment attributes [23].
3 Subject gained access to object.

As ARBAC is the next part after authentication, so no un-authorized user or agent can execute access control method. This ensures that only authenticated subject/users can gain access through ARBAC module and subjects are given privileges according to their roles. In our work, example of subject is user who seeks forensic data and the edge server that is the actual source of new data to be added into activity log ledger. On the other hand, example of object is the basic unit of each of the data stored in activity log ledger as a form of block in the blockchain, where block refers to the unit element of each blockchain ledger as depicted in Fig. 19.

3.8 Access request of object by subject

One common thing between subject and object is that both of them have attributes. These attributes help to identify the roles of any subject based on the subject attribute and access permissions of each subject are assigned accordingly. As such, a role of a subject may have attributes in name value pair like, department: water, userType: admin, objectType: water, read: yes, write: no, deleteUser: no, status: active. The role refers to

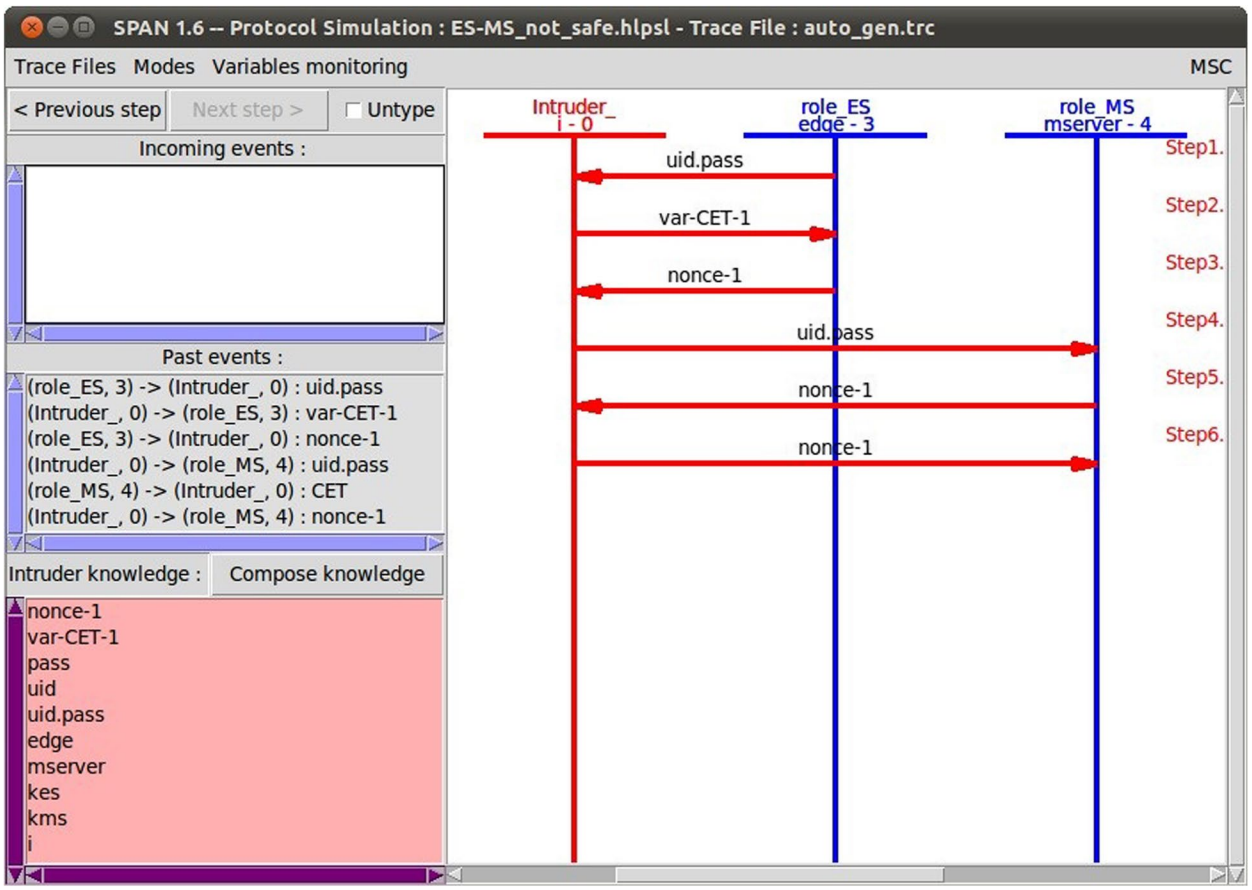

Fig. 14 Simulation of protocol in Fig. 11 using AVISPA—when public-private keys not used

a user from water department authority who has access to water data s/he has read operation privilege and no write privilege on object or data. As these are some confidential data of citizens, data can only be added by some edge servers but not by any human agent. On the other side, a single object consists of multiple attributes which help to identify each of them from the blockchain ledger. Example of object attributes: type:waterlog, status:active, location: 23.7104, 90.4074, creationTime:2038-01-19 03:14:07.

3.9 Attributes and privileges in ARBAC

Subject attributes, object attributes, rules, privileges, and environment attributes are processed in combination to achieve a decision whether a subject will get access to object or not. Privileges define which subject will be allowed which operation on which object. Again, rules are written in perspective of object type. So, privileges assigned to subjects through roles and rules related to objects are the bridging elements for allowing access to the users or subjects to perform specific operation with the contribution of attributes. Operation refers to the execution of any function on any object. Examples of operation are execute, read, write, delete, upload, download, and so on.

Subject and object attributes are created and assigned by the respective creators; these creators are the admin users who are not the forensic users. Creator of subject attributes are also the admin users who are responsible for managing all the subject attributes and subject itself. On the other hand, objects are created by the edge servers and object attributes are assigned by the edge servers according to the rules stored in policy repository. But object attributes are created and stored by the admin-level users. For all the above situations, admin-level users are the users which are not related to the forensic users. So forensic users and admin-level users are not the same type. For example, a forensic user only has the read operation privilege, but admin-level users have no write operation privilege on object but the write operation privilege only in the policy and attribute repository. Management of these admin-level users is out of the scope of the proposed ARBAC.

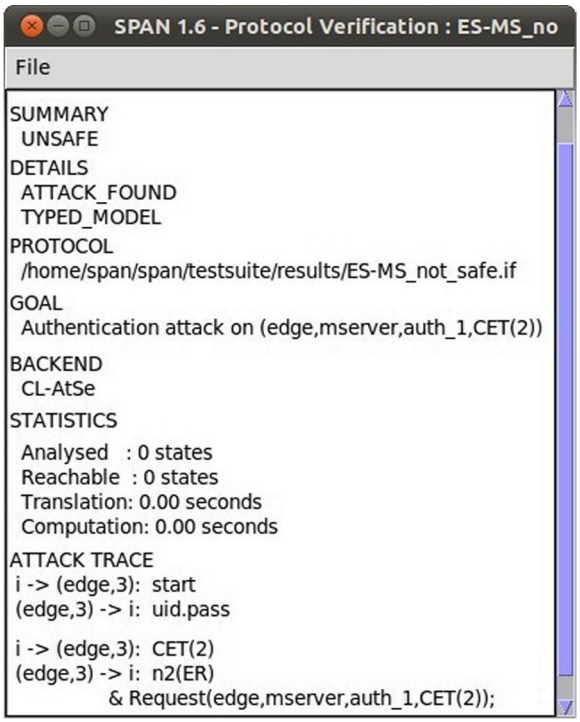

Fig. 15 Simulation of protocol in Fig. 11 using AVISPA—when public-private keys not used

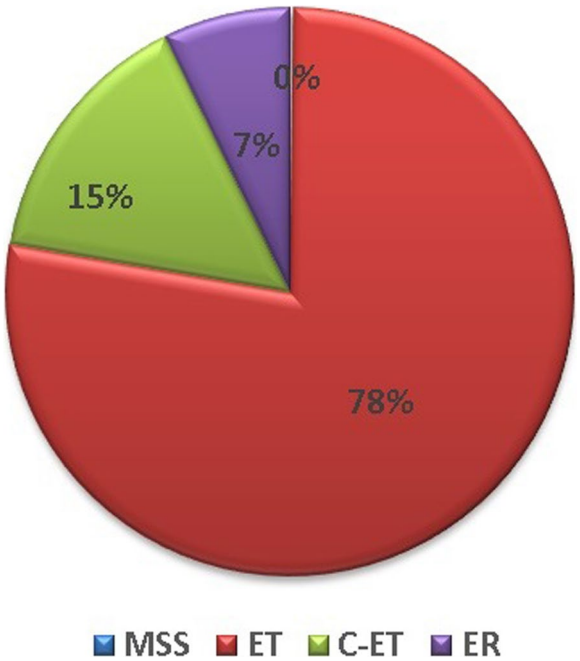

Fig. 16 Computational time to generate secrets for dialogue between edge and main server

Table 4 Required computational time to generate secrets for dialogue between edge and main server, time in milliseconds

MSS	0.007581711
ET	3.158712387
C-ET	0.613069534
ER	0.295686722

3.10 Types of access requests

There can be two varieties of user/subject requests to get object access—one is identifier-based request and another is attribute-based request [23]. In identifier-based request, against each request unique identifier of a single object is given by the subject and that object is retrieved only. For this approach, user or subject needs to know the unique identifier of that specific object. On the other hand, for attribute-based object requests, the subject can access multiple objects based on the attributes provided. An example of attribute based object request can be like:

$$Request =< session, (oType = phoneLog\& \\ forensicDept = telco\& \\ area = dhaka\&oStatus = active), read >$$

The above request denotes the owner of the session who is actually a user seeking for read operation approval only for the objects for which the object expression is matched upon evaluation. The object expression is evaluated for each of the objects in the repository. Only if the object expression is true for each object is added to the list of authorized objects for the owner of the current session.

To execute the above request by the subject, each subject/user must have the permission to get access of the required objects. For this, a permission request is placed as:

$$p = ((oType(o) = phoneLog\& \\ oStatus(o) = active), read)$$

In the above request, the subject is seeking for read operation permission for each object related to water log data from the blockchain ledger. But to get a permission to be approved, the subject has to fulfill some set of conditions. An example of condition can be:

$$c = (uMember(u) = admin\&accessTime() < \\ userDutyExpire(u))$$

Each function of the above condition denotes that the subject has to be an admin and access time should be within the office time. In the above object expression, access time and user duty expiration time are evaluated from some environmental attributes. Examples of

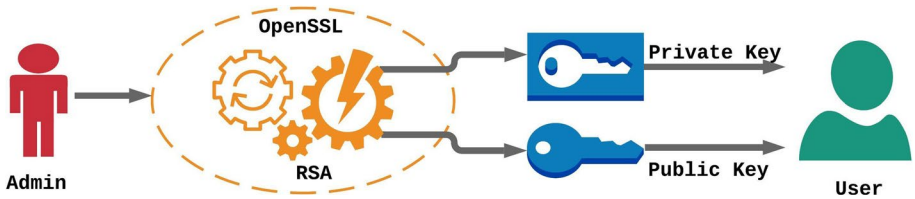

Fig. 17 Asymmetric key generation for all users

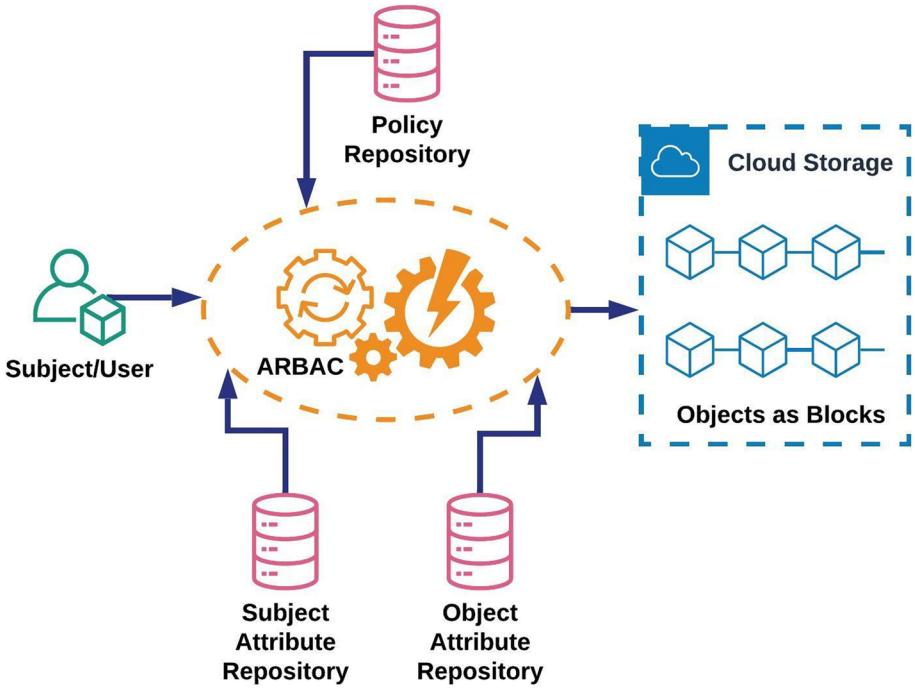

Fig. 18 Basic components of ARBAC

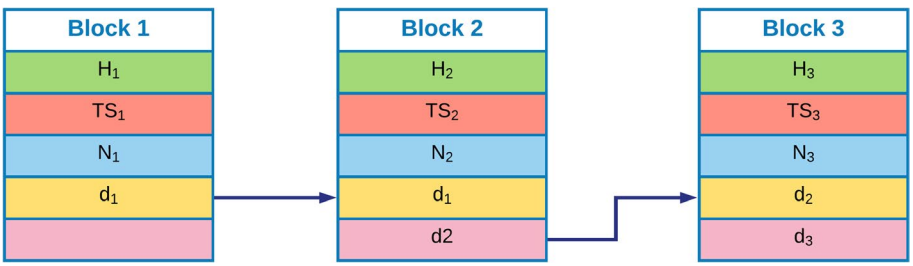

Fig. 19 A chain of three blocks

environmental attributes: current time, location and client device type.

To implement the ARBAC, required repositories are subject attribute repository, object attribute repository, and policy repository. Among them, subject attributes and data required for roles are stored in the subject attribute repository. They are used for subject-related data required for evaluating conditions/rules for approving permission to subject for accessing object. Object attribute repository holds all attributes related to objects those are required for allowing object access to certain user roles, it is required to enforce decision on which user role get access to relevant objects. Policy repository can be treated as a guardian of the whole ARBAC

system that holds the rules/conditions that actually help to determine whether a user should be granted any object access or not.

Environmental conditions and attributes are such as current time, time, location, system status, and any security threat. These are some independent attributes which are detectable on runtime by the help of execution system. Policy Decision Point (PDP) is liable for evaluating subject attributes, object attributes, and environmental conditions and gives a decision to the policy enforcement point [24, 32]. Policy Enforcement Point (PEP) receives object access request from subject and requests PDP for access control decision whether any subject will get object access approval or not [24, 32]. Actual access decision is made by PDP where PEP is the mediator among subject, object, and PDP. All the functional modules described in this sub-section are depicted in Fig. 20.

3.11 Communication between subject and ARBAC

After successful authentication of any user(subject), it requests for object access to ARBAC, where ARBAC is the sole responsible module for allocation of object

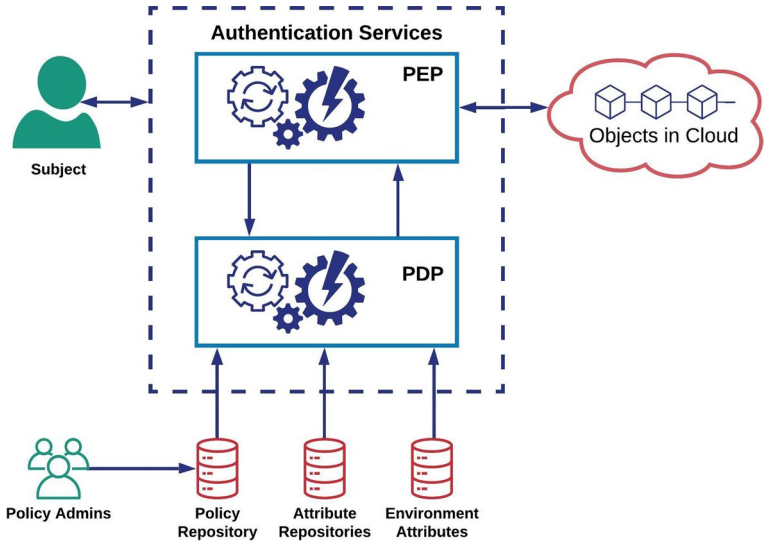

Fig. 20 Functional modules in ARBAC

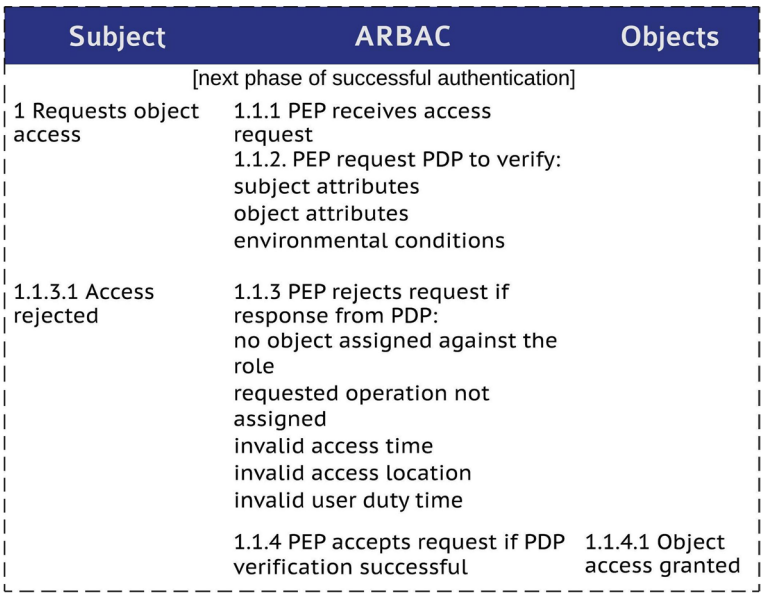

Fig. 21 Communication between subject and ARBAC

Table 5 Roles with user privileges and access time

RoleID	Role	RWX	DutyTime
RD001	Admin	4	0024
RD002	EdgeServer	6	0024
RD003	EdgeDevice	0	0024
RD004	ForensicUser	4	0917
RD005	CorporateUser	4	0917

Table 6 Forensic user and object assignment

ForensicID	ObjectType
FD001	OD001
FD001	OD004
FD002	OD002
FD002	OD003
FD003	OD004

Table 7 Types of objects

ObjectType	TypeName
OD001	AttendanceLog
OD002	WaterLog
OD003	ElectricityLog
OD004	PhoneLog

according to the roles and privileges of subject. In ARBAC, PDP evaluates subject attributes, object attributes, and environmental conditions collectively that helps PEP to decide whether a subject will get access permission to object or not. Here, PEP works as a mediator between PDP and subject. If PDP gives positive response to PEP, a subject gets object access permission. We depict these steps in Fig. 21. Next section describes use cases of ARBAC.

4 Use cases of ARBAC

In this section, we have shown some use cases of ARBAC module.

4.1 ARBAC roles and attributes

In Table 5, RWX denotes the user privileges where RWX of Admin is 4 represent binary 1 0 0. In this binary pattern, first bit represents R for read operation, second bit represents W for write operation, and X means delete operation. RWX = 4 for Admin means, this user only can read but no write and delete privilege assigned. In the same table, DutyTime for ForensicUser is 0917; it means forensic user's duty time is 09:00 AM to 05:00 PM.

Table 8 Forensic departments

ForensicID	ForensicName
FD001	Police
FD002	Fire
FD003	Telco
FD004	Municipality

Table 8 exhibits examples of forensic department names and Table 7 gives idea regarding types of objects. On the other hand, Table 6 represents the assignment of object type against each type of forensic department, where Table 7 shows a list of object types and the same for the forensic departments in Table 8.

4.2 Experimental use cases for ARBAC

Figure 22 shows that violation of environmental rules may occur due to invalid subject as subject requests access from any location where the location is not registered. On the other hand, in Fig. 23, another type of environmental rule violation is due to invalid access time which occurs when any user requests access out of his/her office time. In Fig. 24, this is shown that if a user with fire department user credential tries to access phone log data, then the request is rejected. Because a user credential used by fire department authority is assigned to a specific role relevant to a fire department person. The last depiction in Fig. 25 illustrates a user tries to delete certain object which is not possible because no user has a delete privilege. Blockchain being an immutable data structure, no party can delete or remove any block or object from the ledger. The detail on this is as follows.

5 Blockchain ledgers as data storage

A blockchain is a growing list of records (blocks) that are linked using the cryptographic hash value [33]. Each block contains a header or block number, a timestamp, a nonce, hash value of the previous block and transaction, or hash value of the current block. The transactions are verifiable and the data entered into it cannot be deleted [14]. Blockchain technology is useful in financial, commercial, industrial, and other sectors as well. In blockchain, the blocks are interlinked. Blockchain also provides data storage using distributed ledger for peer-to-peer communications [34]. Each block is constructed using a data structure called a Merkle tree as shown in Fig. 26.

In Fig. 26, T_1, T_2, T_3 and T_4 are transactions (records of a user) and H denotes a cryptographic hash function such as SHA-512. The upper level of the transactions d_1, d_2, d_3, and d_4 represent the hash codes (values) of the respective transactions. The hash value d_{12} is a new hash

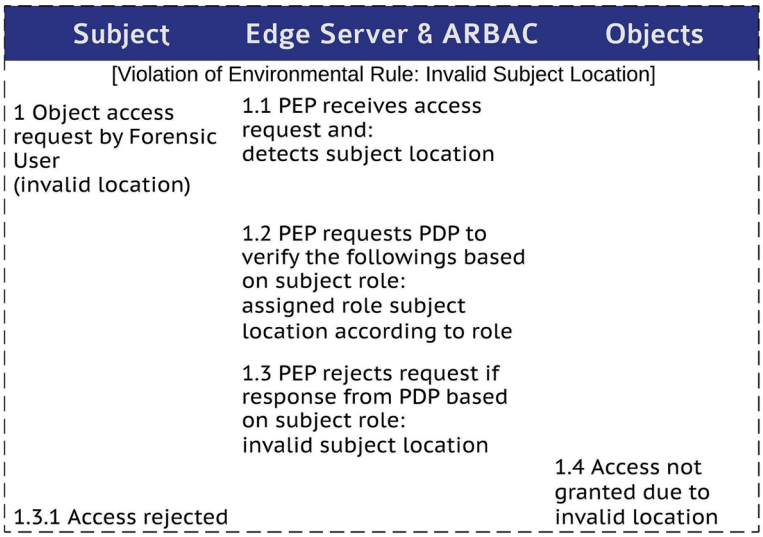

Fig. 22 Use case of ARBAC—invalid subject location

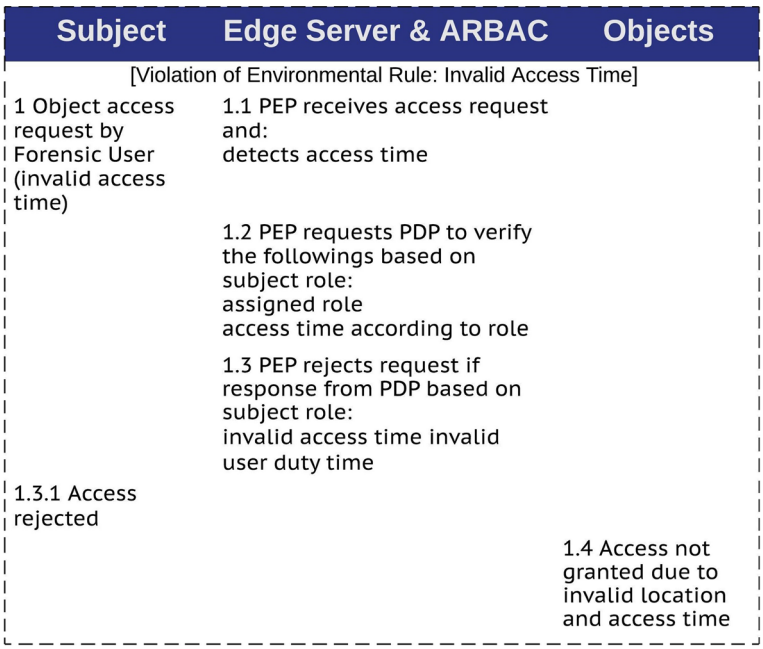

Fig. 23 Use case of ARBAC—invalid access time

value created from the hash values of its children in the tree. Similarly, the hash value d_{34} is produced from the hash values d_3 and d_4. At the root level, there is a final hash value, which is the output from the hash values of its children. Actually, in the root, there is other information such as block header or number, timestamp, a nonce, and the hash value of its previous block. A structure of a block is given in Fig. 27. In the figure, we consider the i_{th} block of a blockchain ledger. H_i, TS_i, and N_i represent the header, timestamp (date and time), and nonce (a random number or string). The hash values of the $(i\text{-}1)_{th}$ block and i_{th} block are denoted by d_{i-1} and d_i respectively. In a ledger (blockchain), the blocks are chained (linked) as shown in Fig. 19.

In our proposed framework, there is a blockchain ledger for each specific service, such a ledger is for

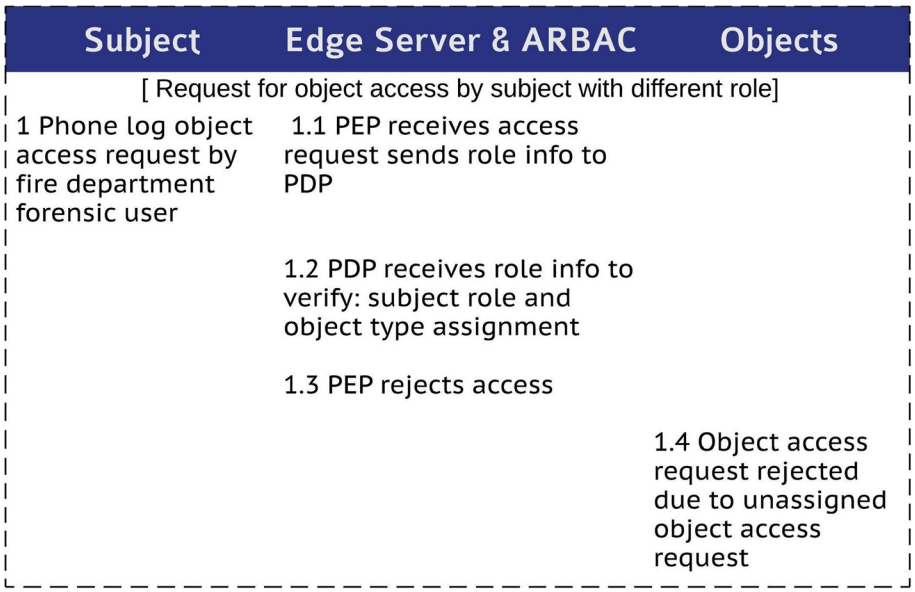

Fig. 24 Use case of ARBAC—subject with different role

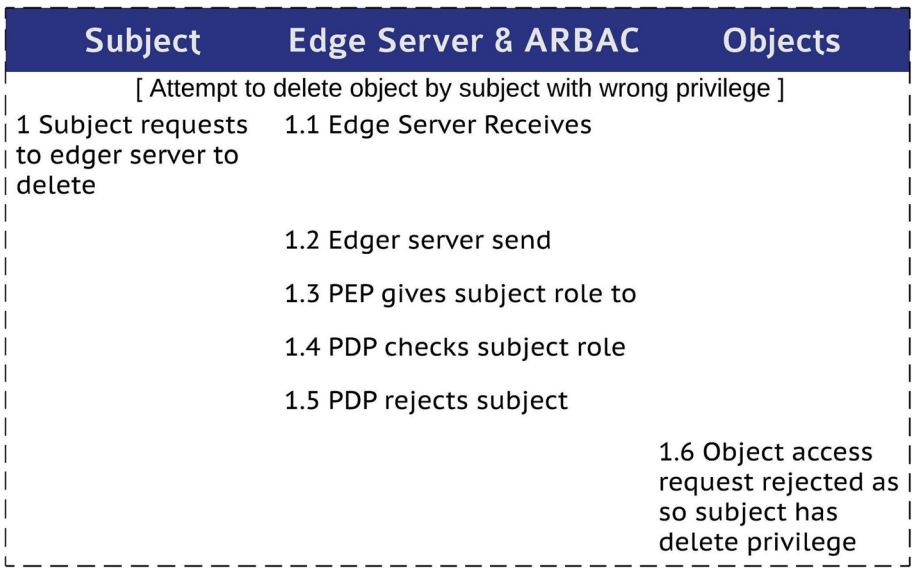

Fig. 25 Use case of ARBAC—subject with wrong privilege

electricity bills, another ledger is for water bills, and so on. All the ledgers are stored in a main server or a cloud server, which facilitates the proper management of data for forensic use in a smart city. The ledgers in the main server are shown in Fig. 28.

5.1 Lightweight proof of work and consensus

Proof of Work (PoW) consensus in Bitcoin is computationally time-consuming as it requires numerous participants to participate in consensus [35]. So, in the proposed framework, we minimize the number of participants by selecting edge servers and main server as the only users to add new block through a lightweight consensus. This lightweight consensus also increases scalability as we utilize it within a consortium blockchain [4]. To add a new block to a blockchain ledger, an edge device finds relevant edge server through smart contract and sends the data to relevant edge server after successful authentication. So, broadcasting of new block is not required as an edge device finds its related server required. For this

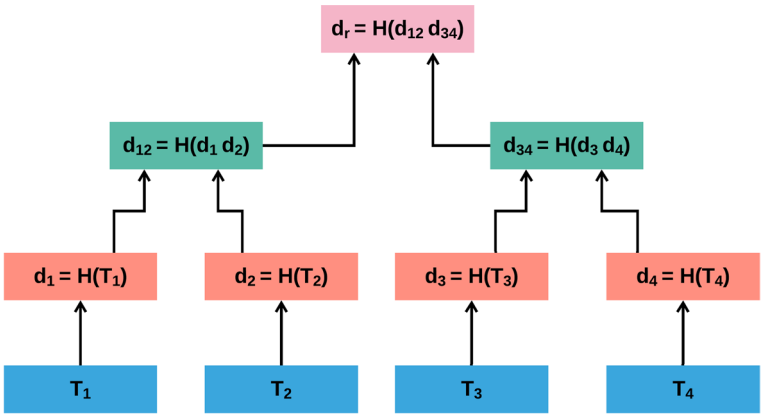

Fig. 26 Merkle Tree

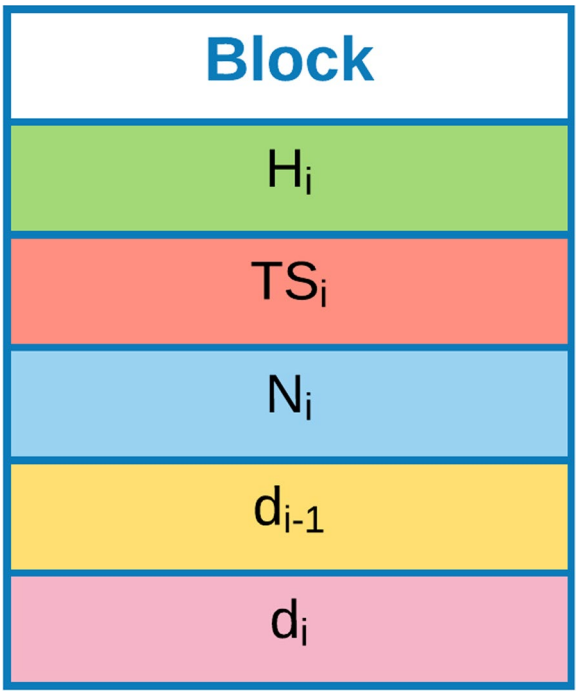

Fig. 27 Structure of a block

case, edge server plays the role of a miner. Edge device chooses miner and then it approaches the main server to add the new data block to respective blockchain ledger. Main server authenticates an edge server with the help of ARBAC module; at the same time, main server validates the new data to be added based on some predefined requirements or rules set by the system admin. Based on these requirements or rules, relevant edge server conducts checking of data prior to adding the data block into the blockchain ledger. In this phase, edge server checks whether proper data come from the relevant edge device or not; if checking is successful, then it proceeds to main server. Now, as the edge server is authenticated by the main server, targeted ledger for the new block must be chosen by the main server. Proof of work is done by main server based on some predefined requirements set. Thus, main server and relevant edge server come to a consensus based on proof of work. So, the new block is added to the relevant ledger upon consensus.

$$\begin{aligned} &\textit{Authentication of ES} > \textit{data validation by MS} > \\ &\textit{PoW by MS} > \\ &\textit{Consensus by ES and MS} > \textit{New Block Added} \end{aligned}$$

Sequence for proof of work and consensus achieved can be described as above, where first step is to authenticating the edge server (ES) by the main server (MS) and at last new block is added to relevant ledger by the main server (MS).

5.2 Distributed system for blockchain

As data in the immutable blockchain ledgers grows simultaneously, each of the ledgers will grow a long way which is almost impossible by a single system to handle. On the other hand, proof of work for achieving consensus when finding the right place to add new block and verifying the whole chain (blockchain ledger) take huge overhead. No data can be deleted from any blockchain ledger as all the blocks are interlinked, so new version of data or block can be added. For the purpose of load balancing the overhead and smooth operation, whole blockchain data storage is designed in distributed manner. How devices interact themselves to store data into the main server is as follows.

5.3 System functionality

The proposed framework in this paper includes five groups of behaviors: SmartContract (SC), EdgeDevice

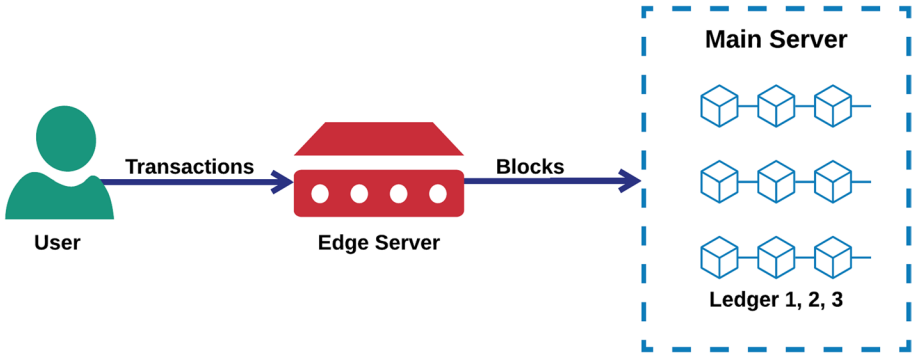

Fig. 28 Interaction of user with ledgers in main server

(ED), EdgerServer (ES), ForensicUser (FU), and MainServer (MS) as depicted in Fig. 29. Blockchain facilitates SC that mediates the communication between ED and ES. ED plays the role of a peripheral device of the whole framework that utilizes SC for all communications to ES. ED finds the relevant ES in the area to submit the activity log through SC. ES preregisters itself with the MS for all future communications to collect activity log data from ED upon authentication and send it to MS. MS further authenticates requests from ES to establish direct communication to ED for adding new data into the ledger within MS. ES also takes part in consensus with MS to validate and add new block of data. Finally, MS itself decides the type of ledger according to the types (example in Table 7) of activity log data. In addition, MS also preregisters FU and authenticates it on request.

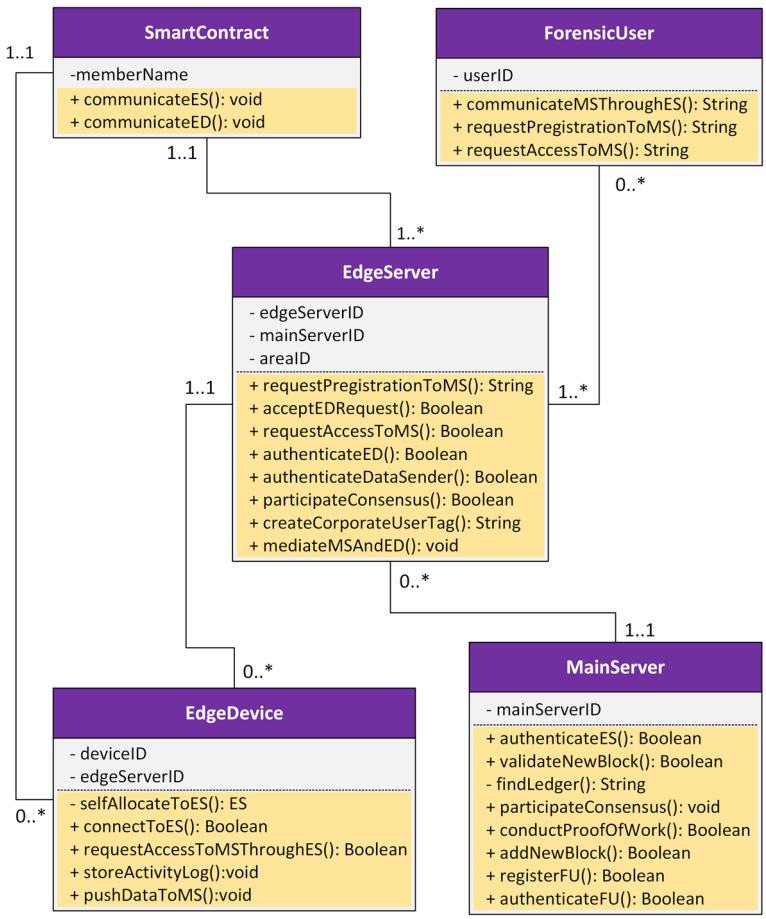

Fig. 29 Class diagram that shows system functionality

Above algorithm 1 shows that blockchain-based SC initializes separate lists of ED, ES, and area in lines 1 to 3 and waits for a request from ED in line 4 to connect it to relevant ES. In the next line, SC authenticates the requester. Upon successful authentication, SC finds the relevant ES for ED, sends activity log to proper ES, set the ES as miner, and finally connects the ED to ES in lines 7 to 10. The next section discusses the security aspects of the framework.

1: $EDList \leftarrow[\,]$
2: $ESList \leftarrow[\,]$
3: $AreaList \leftarrow[\,]$
4: Awaits request from i_{th} ED
5: Authenticate $EDList[i]$
6: **for** $h \leftarrow 1$ to $length(ESList)$ **do**
7: Find relevant h_{th} ED from $ESList$
8: Sends activity log from $EDList[i]$ to $ESList[h]$
9: Set $ESList[h]$ as a miner
10: Connects $EDList[i]$ with ES
11: **end for**
12: Goto 1

Algorithm 1. Process of SC in edge network

6 Discussion on security aspects

Some propositions and their proofs based on a few security use cases from our experiment along with a comparison of security features are described below:

6.1 Security propositions

Proposition 1: Mixed type of verification and authentication system provides better security

Proof: In our proposed framework, verification and authentication are conducted using conventional id, password, and asymmetric keys. As such, when a corporate user wants access to main server the user is verified using its credentials (username, password) and the edge server signs the TAG_{ck} using its own private key. Later main server verifies the signature of the edge server using the public key of edge server. Where public key is a publicly available key, but private key is not. User credential-based login approach, asymmetric cryptography, and new concepts of key generations together form a strong verification and authentication system.

Proposition 2: Time-based key generation process ensures randomization of secret keys

Proof: For all the secret key generation processes mentioned in this article, are randomized based on timestamp of the respective devices. For instance, TS_{i}, TS_{ess}, TS_{tag}, TS_{mss} and TS_{ET} all are timestamps of respective devices required for computing secret keys.

Proposition 3: The proposed approach prevents fraud

Proof: Multiple levels of security key generations processes, use of existing id-password-based credentials, and use of asymmetric key ensure the prevention of any kind of unauthenticated access.

Proposition 4: Ensures integrity of secret key

Proof: In all of the phases, most of the secret key computations are backed up by hash function. For this, implementation SHA-1 is used with variety of salt values.

Proposition 5: The proposed framework prevents impersonation attack

Proof: All computed secret keys are encrypted and sometimes hashed if it is required to transmit within multiple devices, so any intruder cannot get it beneficiary if it is impersonated. Even if any key is required to be stored, it is done in a secured manner. Such as, main server encrypts the C-ET with the public key of edge server when sending it over the network to edge server.

Proposition 6: Proper distribution of appropriate objects to subjects

Proof: In our framework, object refers to data and subject refers to user requesting data access. ARBAC is the module which decides the object allocation to proper users according to user roles and privileges, which helps to ensure selective data access for the users of the system, that also reduces system execution overhead because only the allocated data is accessed by the user rather than accessing whole data.

Proposition 7: Data integrity due to immutability of blockchain

Proof: Implementation of blockchain ensures data integrity because data within blockchain ledger are unchangeable. Blockchain is immutable, which ensures that data within blockchain ledger is not changeable. Each block of blockchain holds the hash of current and previous block hash. If any block of the blockchain is to be altered, the whole chain of data is required to be changed, which is impossible.

6.2 Comparison of security features

We compare Wang et al. [4], Jangirala et al. [5], Bonnah et al. [6], and Sharma et al. [2] with the security features in our proposed framework in Table 9. It shows that the proposed framework provides even better security and scalability than its close contender in [5] as we combine secret key, asymmetric key, and hash function together in edge computing using consortium blockchain.

7 Conclusion

The main contribution of this research work is the proposed authentication process that efficiently authenticates all users and devices with newly developed code (value) generation processes. Combination of lightweight consensus and consortium blockchain in edge network

Table 9 Comparison of the proposed framework with other state of the art methods

Security features	[4]	[5]	[6]	[2]	Our framework
Man-in-the-middle attack	Yes	Yes	Yes	No	Yes
Impersonation attack	Yes	Yes	Yes	Yes	Yes
Mutual authentication	Yes	Yes	Yes	Yes	Yes
User anonymity	Yes	Yes	Yes	Yes	Yes
Integrity of secret keys	No	No	Yes	Yes	Yes
Formal security verification in AVISPA	No	Yes	Yes	Yes	Yes
Blockchain-enabled	Yes	Yes	Yes	No	Yes
Type of security algorithm	Asymmetric	Bitwise XOR, Oneway Hash	Asymmetric	Oneway Hash	Asymmetric, Oneway Hash
Resistance to denial-of-service attack	No	Yes	No	Yes	Yes
Tamper Resistant Data Storage	Yes	Yes	Yes	No	Yes
Session key integrity	Yes	Yes	No	Yes	Yes
Insider attack	No	Yes	No	Yes	Yes
Contributors to Scalability	Edge Computing, Consortium Blockchain	Edge Computing	Edge Computing	None	Edge Computing, Lightweight Consensus, Consortium Blockchain

enhanced scalability of the proposed framework and integrity of authentication keys. Furthermore, it enriched the endpoint device authentication, so any unwanted device cannot add a new block of data to the blockchain data storage. On the other hand, ARBAC module is our hybrid implementation composed of role and attribute based access control system for our research which played a vital role to the overall work to control forensic user access. In our experiment, the proposed approach prohibits users from modifying any data stored in blockchain data storage for several test cases. That ensures data immutability which is an inherent feature of blockchain. In system design section, authentication of users are verified using AVISPA when sharing secret information among them assuming that intruder may interfere the communication. When transmitting secrets among users, result analysis phase shows that highly confidential citizen data are accessed only by authenticated forensic users with appropriate roles assigned by admin-level users, where no intruders can alter data.

Appendix

Tables 1 and 3 show the details of all the notations used.

Acknowledgements

Md. Ezazul Islam is supported by the Henry Sutton PhD Scholarship stipend and Research Training Program (RTP) fee-offset scholarship through Federation University Australia.

Authors' contributions

All authors have contributed equally. Authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Operations of proposed solution (algorithms) are implemented and verified using some existing verification tools, where no data are required for the experiments. So, only computer scripts (source code) for the experiments are available. The source codes of this study are available from the corresponding author on reasonable request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 4 July 2022 Accepted: 1 July 2023
Published online: 25 July 2023

References

1. K. Awuson-David, T. Al-Hadhrami, M. Alazab, N. Shah, A. Shalaginov, Bcfl logging: An approach to acquire and preserve admissible digital forensics evidence in cloud ecosystem. Futur. Gener. Comput. Syst. **122**, 1–13 (2021). https://doi.org/10.1016/j.future.2021.03.001. https://www.sciencedirect.com/science/article/pii/S0167739X21000807

2. G. Sharma, S. Kalra, A secure remote user authentication scheme for smart cities e-governance applications. J. Reliab. Intell. Environ. **3**(3), 177–188 (2017)
3. I. Yaqoob, K. Salah, R. Jayaraman, Y. Al-Hammadi, Blockchain for healthcare data management: opportunities, challenges, and future recommendations. Neural Comput. & Applic. **34**(14), 11475–11490 (2022)
4. J. Wang, L. Wu, K.K.R. Choo, D. He, Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure. IEEE Trans. Ind. Inform. **16**(3), 1984–1992 (2019)
5. S. Jangirala, A.K. Das, A.V. Vasilakos, Designing secure lightweight blockchain-enabled rfid-based authentication protocol for supply chains in 5g mobile edge computing environment. IEEE Trans. Ind. Inform. **16**(11), 7081–7093 (2019)
6. E. Bonnah, J. Shiguang, Decchain: A decentralized security approach in edge computing based on blockchain. Futur. Gener. Comput. Syst. **113**, 363–379 (2020)
7. P. Sanda, D. Pawar, V. Radha, Blockchain-based tamper-proof and transparent investigation model for cloud vms. J. Supercomput. **78**(16), 17891–17919 (2022). https://doi.org/10.1007/s11227-022-04567-4
8. J. Chen, X. Ran, Deep learning with edge computing: A review. Proc. IEEE **107**(8), 1655–1674 (2019)
9. H.N. Noura, O. Salman, A. Chehab, R. Couturier, Distlog: A distributed logging scheme for iot forensics. Ad Hoc Netw. **98**, 102,061 (2020)
10. Y. Chen, Y. Lu, L. Bulysheva, M.Y. Kataev, Applications of blockchain in industry 4.0: A review. Inf. Syst. Front. 1–15 (2022)
11. A. Ellervee, R. Matulevicius, N. Mayer, in *ER Forum/Demos*, A comprehensive reference model for blockchain-based distributed ledger technology (2017), pp. 306–319. https://ceur-ws.org/Vol-1979/. https://ceur-ws.org/Vol-1979/paper-09.pdf
12. E. Koutsoupias, P. Lazos, F. Ogunlana, P. Serafino, in *The World Wide Web Conference*, Blockchain mining games with pay forward (2019), pp. 917–927. https://doi.org/10.1145/3308558.3313740
13. S. Rouhani, R. Belchior, R.S. Cruz, R. Deters, Distributed attribute-based access control system using permissioned blockchain. World Wide Web **24**(5), 1617–1644 (2021)
14. E. Androulaki, J. Camenisch, A.D. Caro, M. Dubovitskaya, K. Elkhiyaoui, B. Tackmann, in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, Privacy-preserving auditable token payments in a permissioned blockchain system (2020), pp. 255–267. https://doi.org/10.1145/3419614.3423259
15. D. Kirli, B. Couraud, V. Robu, M. Salgado-Bravo, S. Norbu, M. Andoni, I. Antonopoulos, M. Negrete-Pincetic, D. Flynn, A. Kiprakis, Smart contracts in energy systems: A systematic review of fundamental approaches and implementations. Renew. Sust. Energ. Rev. **158**, 112013 (2022)
16. A.K. Samanta, B.B. Sarkar, N. Chaki, A blockchain-based smart contract towards developing secured university examination system. J. Data Inf. Manag. **3**(4), 237–249 (2021)
17. F.H. Al-Naji, R. Zagrouba, *Cab-iot: Continuous authentication architecture based on blockchain for internet of things* (J. King Saud Univ.-Comput. Inform, Sci, 2020)
18. S. Xu, Z. Zhang, M. Kadoch, M. Cheriet, A collaborative cloud-edge computing framework in distributed neural network. EURASIP J. Wirel. Commun. Netw. **2020**(1), 1–17 (2020)
19. M.A. Uddin, A. Stranieri, I. Gondal, V. Balasubramanian, Blockchain leveraged decentralized iot ehealth framework. Internet Things **9**, 100,159 (2020)
20. S. Joshi, S. Stalin, P.K. Shukla, P.K. Shukla, R. Bhatt, R.S. Bhadoria, B. Tiwari, Unified authentication and access control for future mobile communication-based lightweight iot systems using blockchain. Wirel. Commun. Mob. Comput. **2021** (2021). https://www.hindawi.com/journals/wcmc/2021/8621230/
21. I. Ali, S. Sabir, Z. Ullah, Internet of things security, device authentication and access control: a review. arXiv preprint arXiv:1901.07309 (2019)
22. Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, J. Wan, Smart contract-based access control for the internet of things. IEEE Internet Things J. **6**(2), 1594–1605 (2018)
23. Q.M. Rajpoot, C.D. Jensen, R. Krishnan, in *IFIP Annual Conference on Data and Applications Security and Privacy*, Integrating attributes into role-based access control (Springer, 2015), pp. 242–249
24. V.C. Hu, D. Ferraiolo, R. Kuhn, A.R. Friedman, A.J. Lang, M.M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone et al., Guide to attribute based access control (ABAC) definition and considerations (draft). NIST Spec. Publ. **800**(162), 1–54 (2013)
25. P. Kamboj, S. Khare, S. Pal, User authentication using blockchain based smart contract in role-based access control. Peer Peer Netw. Appl. **14**(5), 2961–2976 (2021)
26. S.L. Ludin, P. Laghate, M.J. Stevens, F.R. Shotton, J. Hatala. Multi-domain configuration handling in an edge network server (2017). US Patent 9,769,238. https://patents.google.com/patent/US11146615B2/
27. S.N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, A. Bani-Hani, Blockchain smart contracts: Applications, challenges, and future trends. Peer Peer Netw. Appl. **14**(5), 2901–2925 (2021)
28. L. Gong, R.M. Needham, R. Yahalom, in *IEEE Symposium on Security and Privacy*. Reasoning about belief in cryptographic protocols, vol. 1990 (Citeseer, 1990), pp. 234–248
29. Y. Boichut, T. Genet, Y. Glouche, O. Heen, in *2nd Conference on Security in Network Architectures and Information Systems (SARSSI 2007)*, Using animation to improve formal specifications of security protocols (2007), pp. 169–182. https://www.sciencedirect.com/science/article/pii/S1571066106001897
30. V. Lozupone, Analyze encryption and public key infrastructure (pki). Int. J. Inf. Manag. **38**(1), 42–44 (2018)
31. C. Easttom, The rsa algorithm explored. Int. J. Innov. Res. Inf. Secur. **4**(1) (2017). http://www.ijiris.com/volumes/Vol4/iss1/01.JAIS10082.pdf
32. S. Dramé-Maigné, M. Laurent, L. Castillo, H. Ganem, Centralized, distributed, and everything in between: Reviewing access control solutions for the IoT. ACM Comput. Surv. (CSUR) **54**(7), 1–34 (2021)
33. M. Russo, N. Šrndić, P. Laskov, Detection of illicit cryptomining using network metadata. EURASIP J. Inf. Secur. **2021**(1), 1–20 (2021)
34. C. Nartey, E.T. Tchao, J.D. Gadze, B. Yeboah-Akowuah, H. Nunoo-Mensah, D. Welte, A. Sikora, Blockchain-IoT peer device storage optimization using an advanced time-variant multi-objective particle swarm optimization algorithm. EURASIP J. Wirel. Commun. Netw. **2022**(1), 1–27 (2022)
35. S. Zhang, J.H. Lee, Analysis of the main consensus protocols of blockchain. ICT Express **6**(2), 93–97 (2020)

Publisher's Note