Hindawi
Security and Communication Networks
Volume 2022, Article ID 5186376, 14 pages
https://doi.org/10.1155/2022/5186376

*Research Article*

# Cluster-Based Authentication Process in a Smart City

**Md. Rafiqul Islam**,[1] **Md. Nasim Adnan**,[2] **Md. Ezazul Islam**,[3] **and A B M Shawkat Ali**[4]

[1]*Computer Science and Engineering Discipline, Khulna University (KU), Khulna 9208, Bangladesh*
[2]*Department of Computer Science and Engineering, Jashore University of Science and Technology (JUST), Jashore 7408, Bangladesh*
[3]*Department of Computer Science, American International University-Bangladesh (AIUB), Dhaka 1229, Bangladesh*
[4]*Rajapore, Ghouradap, Jamalpur 2000, Bangladesh*

Correspondence should be addressed to Md. Rafiqul Islam; dmri1978@yahoo.com

Received 14 February 2022; Accepted 24 May 2022; Published 14 June 2022

Academic Editor: Habib Ullah Khan

Copyright © 2022 Md. Rafiqul Islam et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The widespread deployment of the Internet of Things in any smart city provides a regular flow of huge amount of data in server(s) that poses challenges for effective and efficient management to improve the quality of citizens' life. To maintain the privacy and security of these data, a proper and secured identification and authentication process is very essential. In this paper, we propose a cluster-based identification and authentication process for the users, edge servers, and service servers, which are engaged in storing, processing, and accessing data. The proposed identification and authentication process is secured due to some codes (values), which are not possible to compute except by the concerned entities. For the proposed trust evaluation method (which actually strengthens the proposed authentication process), we consider major components and their integration in the model very carefully so that the simulation results become credible. Hence, we hope that the simulation results will be useful for the readership. As a whole, the proposed approach has potentials of being implemented in real-time applications.

## 1. Introduction

World population has increased significantly in the last few decades, and most of the people are attracted to city "life" for job opportunities, improved health care, and education, to note a few. As a result, the number of city dwellers has increased at an unprecedented rate. According to statistics, the percentages of people living in cities were 29% and 50% in the years 1950 and 2008, respectively, which is predicted to reach 65% in 2040 [1]. With the rapid advancement and innovation of Information Technology (IT), the world is experiencing a paradigm shift in city infrastructures and services. The evolution of this new form of IT-based city is popularly known as "Smart City." In a smart city, digital technologies translate into better public services for inhabitants and better use of resources while impacting the environment less.

The smartness of a city depends on proper integration of numerous IoT devices with different service-oriented information systems. To make a city smart, sharing of information among the citizens and those service-oriented information systems is very important. For example, if a citizen suddenly becomes sick, s/he can ask for medical assistance through her/his smart phone. Then, an ambulance from the Emergency Medical Service will rush to her/his location to transfer her/him to a nearby hospital according to her/his need. Hence, interconnection among the citizens and service providers' information systems to realize different services is a central issue of a smart city.

We know that cloud computing involves the use of globally interconnected data centers. These data centers process a huge volume of IoT data from smart cities and respond as per user requirement. However, as the physical distance between the cloud and the end user is high, transmission latency and response time also increase accordingly [2]. Besides, IoT devices connected to distant data centers try to access sophisticated applications, which impose additional load on networks and contribute to higher latency [3]. Many IoT applications such as a Medical Assistance Management System need to send the location of a

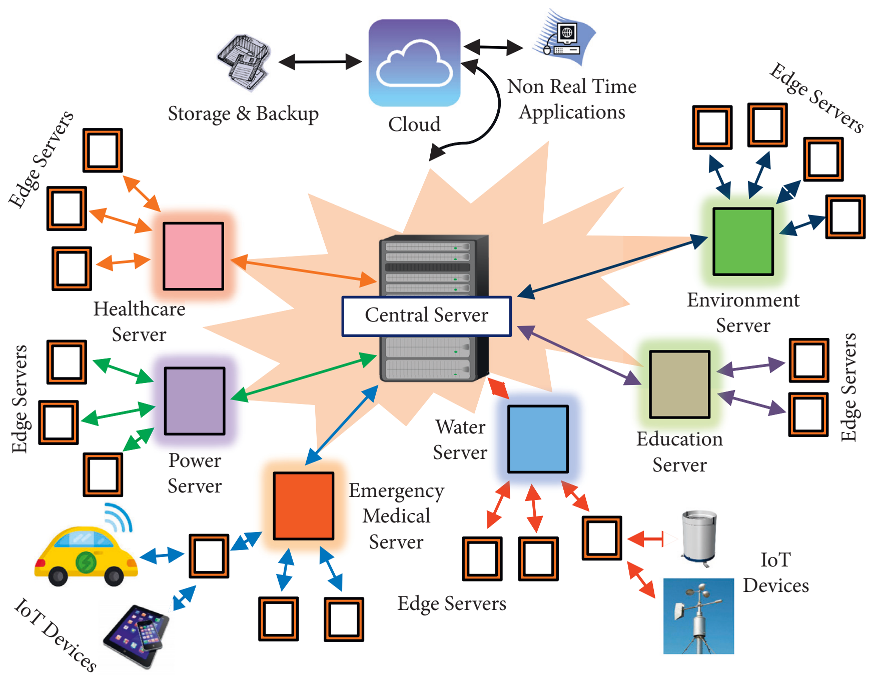


Figure 1: A smart city with different computing service facilities.

citizen, her/his current medical condition (e.g., a severe heart attack) with detailed medical records to the Medical Emergency Service of a smart city for assessment, so that an ambulance dedicated to her/his medical condition can reach to her/his location, provide paramedic service inside the ambulance, evaluate traffic conditions, and direct the ambulance in the best possible way to the nearest hospital of required specialty. However, the inherent limitations of cloud computing such as high latency, non-context-aware behavior, and no support for such mobility requirements pose serious limitations on the use of a real-time emergency service [4].

A large-scale network of IoT with device and network-level heterogeneity along with the ultralarge volume of data and events generated by "things" of IoT will require prohibitively high network bandwidth in the case of "IoT + - Cloud" scenario [3, 5]. Apart from these downsides, cloud computing will simply suffer from processing time inefficiency due to the large overhead of IoT data. Recent research efforts are investigating how to effectively exploit capabilities at the edge of networks to support the IoT in the context of the smart city [6]. In edge computing, massive data generated by different types of IoT devices of a smart city can be processed at the network edge instead of transmitting them to the distant cloud data centers owing to bandwidth and energy consumption concerns. Edge computing can provide services with faster response and greater quality, in comparison with cloud computing. Edge computing is more suitable to be integrated into the paradigm of a smart city to provide real-time services for a large number of end users.

Figure 1 depicts a smart city with dedicated servers for different city-oriented services. By using different IoT devices, users connect to those service servers through edge servers to obtain real-time services (cloud computing can also be used for providing non-real-time services). There can be millions of registered users for a particular server, and one user may be client(s) of multiple servers. Service to service and user to service communication are common scenarios in the context of a smart city. Most of the services can generally be accessed through IoT devices. In such a situation, user identification and authentication is a paramount security issue. It is important to note that edge computing can incorporate a hierarchy of servers with increasing computation capabilities for better service [7]. The hierarchy of servers can be organized into a cluster-based formation. The proposed cluster-based authentication process utilizes this capability of edge computing to manage the security challenges of a smart city more effectively. In a smart city, there will be thousands of users, who will access data, and it will be very difficult to observe every activity of a user. So, we consider cluster-based authentication to minimize the insider attacks, where a cluster member cannot access data of another cluster without going through her/his cluster master, who is a more trusted subject (user) than a cluster member. The clustering will decrease the processing overheads of authentication and authorization systems. The proposed authentication process is integrated with a trust evaluation method that helps to identify malicious activities of the user(s).

An increasing number of studies emphasize on the security, privacy, and risks within smart cities, highlighting the threats relating to information security and challenges for smart city infrastructure in view of management and processing of personal data [8–10]. Hence, privacy, data integrity, authenticity, and availability of information are major concerns for a smart city [11]. In this paper, we focus on to provide high availability of information and at the same time maintain privacy and data integrity. The contributions of the paper are as follows:

(i) We have shown a concept of data processing through edge servers and service servers controlled by the central server(s) in a smart city.

(ii) A robust trust evaluation process has been designed to evaluate the user trust before and after the registration to control the activities of the members as a first line of defense. This process is demonstrated using simulation results.

(iii) A secured cluster-based authentication process has been designed in the paradigm of edge computing. The secrecy of user information and authentication process is verified using AVISPA and Scyther.

(iv) A secured key management process has been verified using simulation results that ensure data confidentiality and integrity.

In this way, the paper contributes to realize an efficient ecosystem of IoT devices, data, and applications along with stringent security requirements. The remainder of this paper is organized as follows. In Section 2, we describe the proposed Trust Evaluation and Cluster-based Authentication Process. Section 3 presents comparison and discussion in detail. Finally, we render our concluding observations in Section 4.

## 2. Trust Evaluation and Cluster-Based Authentication Process

In the proposed cluster-based approach, we consider three types of servers such as edge server, which can ease the processing and storing overload of the other server; service server for each service (e.g., emergency healthcare service); and the central server for a smart city. Each user communicates with an edge server, which is located in the user's locality. A user can get service from any service server through cluster-based communication. All users of a service server will be registered with the help of the edge servers. Users will perform registration through the respective edge server.

*2.1. Trust Evaluation.* For user registration, at first, the "trust" of a user is evaluated. Trust has been treated as a soft security mechanism for protection or a means for security in many fields, ranging from economics, psychology, sociology, medicine, and technology [12]. Hence, it is nearly impossible to generalize the concept of trust. Even for a specific field, trust is very complicated as it can be influenced by many measurable and nonmeasurable attributes [13]. Trust can be classified into two types: one is direct trust and another is indirect trust [14, 15].

For registration in a service server, a direct trust can be gained if a user can present her/his National Identification Number (NID) or passport number or an equivalent authentic document issued by the government. On the other hand, if a user cannot present her/his NID/passport, for registration s/he must be endorsed by one or more existing user(s) having direct trust or a representative (such as a commissioner) from the user's locality. In this way, a user can gain indirect trust and be eligible for registration. Once a user is registered into one or more service server(s), her/his trust will be evaluated continuously according to her/his activities over time. Hence, in the process of trust evaluation, it is required to automatically collect information concerning trust decision attributes, evaluate the level of trust based on the values of trust decision attributes, and at times evaluate the entire trust evaluation process to make the proposed trust evaluation process more effective and updated [13].

The proposed trust evaluation process works as follows. At first, we define the levels of trust (LT) and their associated value ranges according to Table 1. When a user performs registration through direct trust, s/he is assigned "mid (M)" as the LT. Hence, s/he obtains a random value in the interval of [0.41, 0.50] as the initial trust value. Similarly, when a user performs registration through indirect trust, s/he is assigned "low (L)" as the LT. As a result, s/he obtains a random value in the interval of [0.31, 0.40] as the initial trust value. After a user obtains the initial trust value, her/his trust value is continuously updated based on a reward-penalty method. We use the following recurrence to assign ITV (Initial Trust Value) to a user $U_i$ who is registered to the system.

$$\mathrm{ITV} = \begin{cases} [0.41, 0.50], & \text{through direct trust,} \\ [0.31, 0.40], & \text{through indirect trust.} \end{cases} \tag{1}$$

For each unwanted or malicious activity by the user, her/his trust value will be decreased (as a penalty). Malicious activities for which the trust value is penalized or rewarded is called trust decision attributes (TDAs). Malicious activities such as unauthorized read, write, update, or delete of data are examples of some TDAs for which trust value can be decreased. According to the proposed trust evaluation process, TDAs for which the trust value is decreased are grouped into the following two broad categories:

(i) Data-level TDAs

(ii) Application-level TDAs

TDAs of the abovementioned groups and their penalty points are defined in Tables 2 and 3.

On the other hand, for a day-long authorized activity, the user's trust value will be increased (as a reward). For LTs ranging from low ($L$) to high (H), the user's trust value will be increased by 0.01 for each day-long authorized activity. However, for both warning (W) and very high (VH) LTs, the trust value will be increased by 0.001 for each day-long authorized activity. Trust value will not be gone beyond 1.00.

We now explain how the trust value of a user can be changed over time according to her/his activities and, as a result, how it impacts the LT. We suppose that a user is registered in a service server (the Emergency Medical Service Server) through direct trust; hence, the user's LT will be "mid" and a random number in the interval of [0.41, 0.50] will be assigned as the trust value. Let us assume that the

Table 1: Levels of trust and their associated value ranges.

| Levels of trust (LT) | Revocation list (RL) | Warning (W) | Low (L) | Mid (M) | High (H) | Very high (VH) |
|---|---|---|---|---|---|---|
| Value ranges | [0.00, 0.15] | [0.16, 0.30] | [0.31, 0.40] | [0.41, 0.50] | [0.51, 0.80] | [0.81, 1.00] |

Table 2: Penalties for data-level TDAs.

| Data-level attributes | Unauthorized read | Unauthorized write | Unauthorized update | Unauthorized delete | Unauthorized upload | Unauthorized download |
|---|---|---|---|---|---|---|
| Penalties | 0.1 | 0.2 | 0.2 | 0.3 | 0.2 | 0.2 |

Table 3: Penalties for application-level TDAs.

| Application-level attributes | Inappropriate share | Inappropriate use | Unauthorized installation | Unauthorized execution | Unauthorized upload | Unauthorized download |
|---|---|---|---|---|---|---|
| Penalties | 0.1 | 0.2 | 0.2 | 0.3 | 0.2 | 0.2 |

initial trust value (ITV) is 0.45. Now if s/he asks for medical assistance through an application deployed in the emergency medical service server, the emergency medical service will act accordingly. However, if it is identified that the service was called without any purpose, according to Table 3, the user's trust value will be reduced by 0.20 for "inappropriate use" of the application. The user's current trust value (CTV) will be 0.25, and as a result, the user's LT will change from "mid" to "warning" and an appropriate text warning will be sent to the user. After receiving the warning, if the user starts committing authorized activities, then for each day-long authorized activity the trust value will be increased by 0.001. In this way, the user can increase her/his LT. However, if the user does the same malicious activity with CTV 0.25, her/his CTV will again be reduced by 0.20 (changes of values for CTV can be formulated using equation (1)). As the user's CTV will be 0.05, the user's LT will be "revocation list" and her/his information will be put into the "revocation list" database so that s/he will be barred from not only using the Emergency Medical Service Server but also other sensitive service servers. Even if the user tries to obtain a new registration through indirect trust, s/he will be detected from the "revocation list" database and will be barred from obtaining a new registration. However, the user can again be assigned to the "warning" LT only if it is advised by an appropriate regulatory body. The proposed trust evaluation process can be further explained in Figure 2. In Figure 2, DTF (Database for Trust Evaluation) consists of tables of trust decision attributes, LTs and associated value ranges, and the revocation list, and DRU stands for Database for Registered Users.

$$\mathrm{CTV} = \mathrm{ITV} - \sum_{i=1}^{|\mathrm{TDA}|} \mathrm{pen}\left(\mathrm{TDA}_i\right) + \sum_{j=1}^{|D|} \mathrm{inc}. \tag{2}$$

Here, in (2) |TDA| is the number of trust decision attributes. $\mathrm{pen}(\mathrm{TDA}_i)$ indicates the penalty associated with the $i$-th TDA. $|D|$ is the number of day-long authorized activity; inc is the increment (as a reward) of CTV for each day-long authorized activity based on CTV. We define inc as follows:

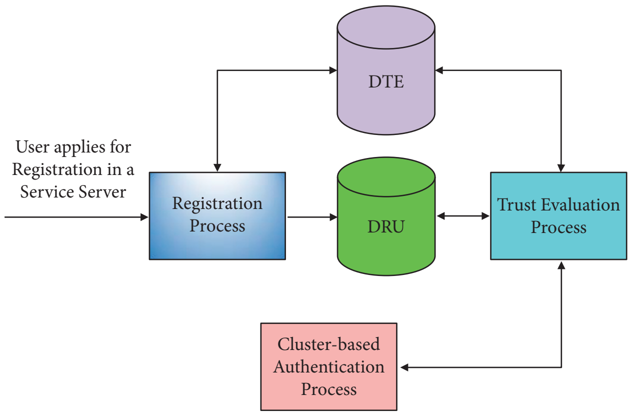

DTE
User applies for Registration in a Service Server
Registration Process
DRU
Trust Evaluation Process
Cluster-based Authentication Process

Figure 2: The process of trust evaluation.

$$\mathrm{inc} = \begin{cases} 0.01, & \text{if CTV} \in [0.31, 0.40], \\ 0.001, & \text{if CTV} \le 0.30 \text{ or CTV} \ge 0.81. \end{cases} \tag{3}$$

It is worth to mention that the whole process of trust evaluation is flexible. For example, penalties of TDAs can be reviewed regularly and can be updated as per relevancy to the trust decision process. Accordingly, if any TDA such as "unauthorized download" from the category of application-level attributes (Table 3) becomes irrelevant in the future, then its penalty will be set to a very small value or zero. If a new TDA appears to be relevant, it can be incorporated with an appropriate penalty. Even if a new category of TDAs evolves, they can be incorporated into the process of trust evaluation. Different data mining algorithms [16, 17] can be applied in the evolution of the trust evaluation process.

We now identify how effective is the proposed trust evaluation process in terms of protection against Denial of Service (DoS) attack and resistance to insider attack. In doing so, we simulate some scenarios as described in the following. The simulation program is developed using Python 3.8. For simulation, we consider three types of scenarios.

(i) Scenario 1: user first commits malicious activity and then plans to do day-long authorized activity

(ii) Scenario 2: user first does day-long authorized activity and then plan to commit malicious activity

(iii) Scenario 3: user starts at random whether to commit a malicious activity or do the day-long authorized activity

In Scenario 1, a user with direct trust obtains a random value in the interval of [0.41, 0.50] as ITV (initial trust value) according to equation (1). Then, s/he starts activities by first committing a malicious activity. The malicious activity is selected randomly from the set of data-level TDAs and application-level TDAs. As a result, the user's CTV (current trust value) will be reduced. To compensate the reduction, the user starts doing day-long authorized activity so that again user's CTV can be increased and again s/he can get an opportunity do some malicious activities. However, if the user goes into the "revocation list" in the database, her/his CTV will never be increased unless s/he is again assigned to the "warning" LT (level of trust) by an appropriate regulatory body.

From the simulation, we see that if a user with direct trust first commits a malicious activity and then plans to do day-long authorized activity (any number of days from 1 to 30), s/he will be able to commit only another malicious activity before going into the "revocation list" in the database. Here, in total two (02) malicious activities are committed, however, at the cost of nine (09) day-long authorized activities (see columns 2 and 3 of the first data row of Table 4). In this paper, we define maximum possible malicious activities (MPMA) as the number of malicious activities before going into the "revocation list." We also define the number of day-long authorized activities before going into the "revocation list" as the cost in days (CD). Here, MPMA = 2 and CD = 9. If the same user first does a day-long authorized activity (any number of days from 1 to 30) to increase CTV and then plan to commit malicious activity, s/he will be able to commit 4 malicious activities (MPMA), however, at the cost of 68 day-long activities (CD) (see columns 6 and 7 of the first data row of Table 4). If the same user starts by random (whether to commit a malicious activity or do day-long authorized activity), then MPMA = 4 and CD = 65 (see columns 10 and 11 of the first data row of Table 4).

An example of how MPMA and CD are calculated for simulation is described in the following. At first, a user with direct trust obtains a random value of 0.49 in the interval of [0.41, 0.50] as ITV. Then, for the case of "Scenario 1," malicious activity is selected randomly from the set of data-level TDAs and application-level TDAs. If "unauthorized download" gets selected from data-level TDAs, the associated penalty of 0.2 is deducted from the CTV of the user. As a result, CTV of the user reduces to 0.49–0.2 = 0.29 according to equation (2). At this point, MPMA = 1 (as one malicious activity has just been committed). The reduction of CTV also affects the user's LT (level of trust). With a CTV of 0.29, the user's LT will change from "mid" to "warning" and an appropriate text warning will be sent to the user. Now, if the user starts committing authorized activity, then for each day-long authorized activity the trust value will be increased by 0.001 (this increment value is assigned with the "warning" LT). For simulation, the number of day-long authorized activity is randomly selected either from 1 to 30 or from 1 to 60. Even if 60 is selected (meaning two month-long uninterrupted authorized activities), CTV is only increased to $0.29 + (0.001 \times 60) = 0.29 + 0.06 \equiv 0.30$. At this point, CD = 60. However, when the user commits the next random malicious activity (suppose with 0.2 penalties), CTV is reduced to 0.30–0.20 = 0.10. At this point, MPMA will be 2 (as two malicious activities have been committed in total). However, with CTV = 0.10, the user is placed into the "revocation list" database and will be barred from not only using the Emergency Service Servers but also other sensitive service servers. The user's CTV will never be increased unless s/he is again assigned to the "warning" LT, however, only if recommended by an appropriate regulatory body. In the same way, the simulation result also shows that in Scenario 1, the most likely value of MPMA is 2 (see Table 4). In Table 4, we interpret "any number of day-long authorized activities from 1 to 30" as "1 to 30" and "any number of day-long authorized activities from 1 to 60" as "1 to 60" for convenience.

From Table 4, we see that even for users with direct trust, and planning intelligently and patiently (user first do day-long authorized activity for any number from 1 to 60 and then plan to commit malicious activities), CD/MPMA is 31.28. This implies that even when the user(s) plan meticulously for committing malicious activities, they will not be able to repeat that within a month. Such a low frequency of malicious activities can occur even by mistake. Hence, it is easily understandable that the proposed trust evaluation process provides formidable protection against Denial of Service (DoS) attack and insider attack.

The calculation of CTV, which in turn determines the LT of a user, is performed in such a way that it is independent of the nature of applications or sensors (or any IoT devices). As detailed above, the calculation of CTV is directly dependent on data-level TDAs, application-level TDAs, and the current LT of a user. Therefore, the CTV of a user is directly dependent on the usage pattern of that particular user. This is why, in the scenario of a concurrent attack on an application, the CTVs of all participating users are reduced in the same manner.

*2.2. Cluster-Based Authentication.* Now, the proposed cluster-based authentication process comes into action. According to the proposed cluster-based authentication process, each service server gets a number generated by the central server and it is unique for each service server. Similarly, each edge server gets a number generated by the respective service server and each user of an edge server is assigned a number generated by the edge server. These numbers are called authentication numbers, which are used for generating keys and authentication token for each entity (user, edge server, and service server). Here, we have considered two types of cryptosystems such as symmetric (secret key) and asymmetric (public key) cryptosystems. In a secret

Table 4: Simulation of the proposed trust evaluation process.

| | Scenario 1 | | | | Scenario 2 | | | | Scenario 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 to 30 | | 1 to 60 | | 1 to 30 | | 1 to 60 | | 1 to 30 | | 1 to 60 | |
| Type of users | MPMA | CD | MPMA | CD | MPMA | CD | MPMA | CD | MPMA | CD | MPMA | CD |
| User with direct trust | 2 | 9 | 2 | 42 | 4 | 68 | 287 | 8977 | 4 | 65 | 37 | 954 |
| User with indirect trust | 1 | 0 | 1 | 0 | 3 | 53 | 44 | 1368 | 4 | 44 | 12 | 326 |

key cryptosystem, a secret key is used for both encryption and decryption. In an asymmetric cryptosystem, private and public keys are used. Figure 3 shows an edge server with its registered users. The respective server generates the authentication number using a cryptographic hash function such as SHA-512 or any other strong hash function. The authentication number generation process is given below.

(1) For the central server, ANcs = H (IDcs ‖ RN ‖ TScs-an) generated by itself
(2) For a service server, ANss = H (IDss ‖ ANcs ‖ TSss-an) generated by the central server
(3) For an edge server, ANes = H (IDes ‖ ANss ‖ TSes-an) generated by the respective service server
(4). For a user, ANu = H (IDu ‖ Pwd ‖ ANes ‖ TSu-an) generated by the respective edge server

The authentication numbers of all registered users are stored in the respective edge servers. All the symbolic notations used in this article are defined in Table 5. All the authentication numbers are stored in the server, which generates them and the respective entity obtains her/his/its authentication number from her/his/its controlling server. Here, an authentication number is much secured since it is generated using a one-way hash function such as SHA-512. Hence, it is not possible to compute an authentication number of a user without knowing the authentication number of the respective edge server and the authentication number of an edge server cannot be computed without knowing the authentication number of the respective service server. A user can get access to the respective edge server using her/his IDu, Pwd, and ANu. For secured communication, each entity will have a secret key. The secret key generation process is as follows:

(1) For a user, Ksec-u = H (IDu ‖ ANu ‖ ESsec ‖ TSu-k) generated by the respective edge server
(2) For an edge server, Ksec-es = H (IDes ‖ ANes ‖ SSsec ‖ TSes-k) generated and stored by the respective service server
(3) For a service server, Ksec-ss = H (IDss ‖ ANss ‖ CSsec ‖ TSss-k) generated and stored by the central server
(4) For the central server, Ksec-cs = H (IDcs ‖ ANcs ‖ CSsec ‖ TScs-k)

For the security of the secret key of each entity (user, edge server, service server, and central server), secrecy of the respective generating entity plays an important role. For example, in case of the secret key of a user (*Ksec-u*), even if *ANu* (authentication number of the user) is compromised no one can compute *Ksec-u* without knowing the *ESsec*

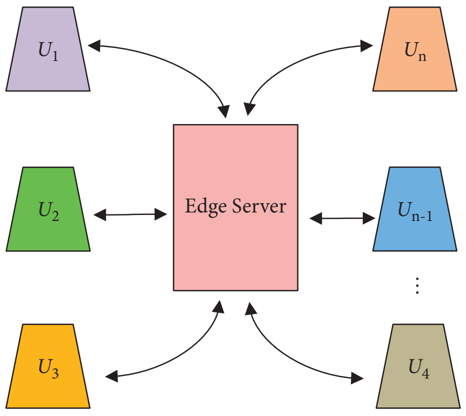


Figure 3: An edge server with its registered users.

(secret key of the edge server). Similarly, other secret or private keys cannot be computed without knowing the secret key of the respective generating entity. Dialogues of the different entities in process of secret key generation are shown in Table 6.

In the proposed approach, each service server and its related edge server constitute a cluster. Each edge server is a cluster member, and the respective service server is a cluster master in the cluster. The secret key of a cluster member is used for secured communication between a member of a cluster and its master (service server). Member to member communication is performed using public key cryptography [18]. However, the user to user communication can be done through the help of the respective edge server. For cluster-based communication, each entity will use token for authentication purposes. The generation process of authentication token is as follows:

(1) For a user, ATu = H (IDu ‖ Ksec-u ‖ ESsec ‖ TSu-at) generated by the respective edge server
(2) For an edge server, ATes = H (Ides ‖ Ksec-es ‖ SSsec ‖ TSes-at) generated by the respective service server
(3) For a service server, ATss = H (IDss ‖ Ksec-ss ‖ CSsec ‖ TSss-at) generated by the central server

Figure 4 shows a cluster where all edge servers are members of the cluster and the respective service server is the cluster master. A cluster member can be authenticated by the cluster master whenever it is necessary. If there are *r* (more than one) service servers in a smart city, there will be *r* clusters for secured communication.

Here, a user gets services directly from the edge server where s/he has done her/his registration. At first, a user logs in to the edge server using her/his ID, Pwd, and

Table 5: Symbols with their definitions.

| Symbol | Definition |
|---|---|
| ‖ | Concatenation |
| ES, SS, CS | Edge server, service server, and central server |
| IDu, IDes, IDss, IDcs | ID of a user, edge server, service server, and central server, respectively |
| Pwd | Password of a user |
| ATu, ATes, ATss | Authentication Token for a user, edge server, and service server, respectively |
| RNcs | Random number of the central server |
| ANu, ANes, ANss, ANcs | Authentication number of user, edge server, service server, and the central server, respectively |
| ESsec, SSsec, CSsec | Secret key of edge server, service server, and central server, respectively |
| TSu, TSes, TSss, TScs | Time stamp for the user, edge server, service, and central server, respectively. Times tamp means date and time. In a time stamp an, k and AT represent authentication number, key, and authentication token, respectively. For example, TSu-an is the time stamp for generating the authentication number of a user |
| Ksec-u, Ksec-es, Ksec-ss | Private key of the user, edge server, and service server, respectively |

Table 6: Dialogues of different entities.

| User | Edge server | Service server | Central server |
|---|---|---|---|
| 1. Requests ANu | 1.1 Sends requested ANu | | |
| 2. Requests with (IDu + ANu) for Ksec-u | 2.1 Generates Ksec-u | | |
| | 3. Requests ANes | 3.1 Sends ANes | |
| | 4. Requests with (IDes + ANes) for Ksec-es | 4.1 Generates Ksec-es | |
| | | 5. Requests ANss | 5.1 Sends ANss |
| | | 6. Requests with (IDss + ANss) for Ksec-ss | 6.1 Generates Ksec-ss |
| | | | 7. Generates Ksec-cs |

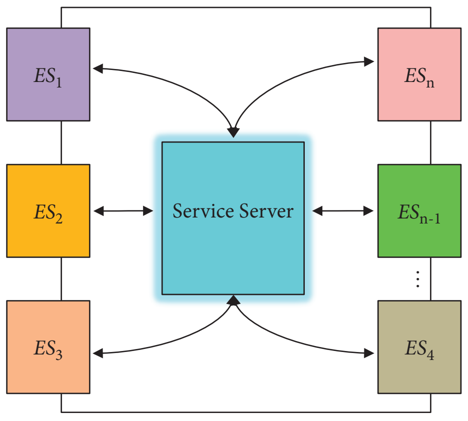


Figure 4: A cluster of edge servers and the respective service server.

authentication token (AT). For a strong authentication process, AT is used here. A user can get service from another edge server where s/he has not registered. This process will be performed in the following way. Suppose a user $U_i$ has performed registration in an edge server $ES_i$ where her/his credential is stored. If s/he wants to get service from an edge server $ES_j$ (where the user $U_i$ has not performed her/his registration), in such case, in the log-in process the edge server asks the user to enter her/his ID, Pwd, and the ID of the edge server $ES_i$. The server $ES_j$ sends an encrypted message (where plain text is ID, Pwd of the user, and the ID of $ES_j$ with a nonce which is used to authenticate the message) to $ES_i$ to authenticate the user $U_i$. Next, $ES_i$ authenticates the user and sends an encrypted response to $ES_j$. Here, all the edge servers in a cluster use public key cryptosystem such as RSA or elliptical curve cryptography for secret messaging. The above scenario is depicted in Figure 5. In Figure 5, CTes denotes ciphertext, which is generated using the private key of $ES_j$ where plain text is ID and Pwd of the user, ID of $ES_j$, and a nonce (*n1*). $ES_i$ decrypts the message using the public key of $ES_j$ and finds the authentication request to authenticate the user $U_i$. $ES_i$ sends an authentication message (*AMes*) with *n1* to $ES_j$. In this way, $ES_j$ authenticates the user $U_i$.

Now, when an edge server itself wants to get service from the service server of the same cluster (cluster master), the edge server is authenticated by the service server. The edge server sends an encrypted message using the secret key of the edge server where plain text is ID and AT of the edge server. After receiving the message, the service server will authenticate the edge server and accept the request for the service if the requesting edge server is an authentic one.

When an edge server wants to get service from a service server (cluster master) of a different cluster, the edge server sends the request to the cluster master of its cluster with its ID and AT and the ID of the server from which it wants to get

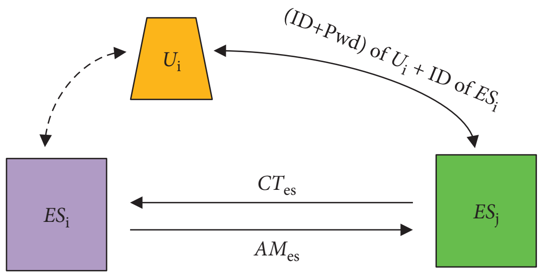


FIGURE 5: Authentication of a user from a different edge server.

service. Suppose an edge server $ES_e$ (cluster member of cluster $e$) wants to get service from $SS_g$ (cluster master of cluster g). Let the cluster master of cluster $e$ is $SS_e$. In this case, $ES_e$ encrypts its ID and AT and sends a request message to $SS_e$ to get the service from $SS_g$. Then, $SS_e$ opens the message and finds ID and AT, and authenticates $ES_e$. If $ES_e$ is an authentic member, then $SS_e$ sends an encrypted message using its private key and a nonce. $SS_g$ receives the message and opens it using the public key of $SS_e$. In this way, $ES_e$ can reach out $SS_g$ for the desired service. This scenario is depicted in Figure 6.

In the cluster-based authentication process, there exists another cluster where members are all the service servers and cluster master is the central server or an authentication server connected with the central server (see Figure 7). This cluster is mainly used to authenticate the service server when interservice server communication is needed. Suppose a service server $SS_b$ wants to get service from the service server $SS_c$. In this situation, the $SS_c$ uses its private key and sends an encrypted message to the central server to authenticate $SS_b$. The central server will then send a message to $SS_b$ to send his ID and AT. Then, $SS_b$ sends its ID and AT as a ciphertext using its secret key. The central server opens it and authenticates $SS_b$ if its AT is in the central server.

Secured communication between two service servers can be performed using public cryptography such as RSA [19] or elliptical curve cryptography. According to Chatzigiannakis et al. [20], elliptical curve cryptography is a better alternative to RSA as elliptical curve cryptography requires less computational resources. In the paradigm of the proposed approach, the majority of devices will have enough resources required. Suppose an edge server $ES_p$ is a member where the cluster master is $SS_c$ and $ES_p$ wants to get service from another service server $SS_g$. In this case, $ES_p$ sends a request message to $SS_g$ with the ID of $SS_c$. Then, $SS_g$ sends the encrypted request message of $ES_p$ using the public key of $SS_c$. $SS_c$ opens it and finds the ID and AT of the edge server $ES_p$ and proceeds accordingly.

Verification of phases 3, 3.1, 3.2, and 3.3 from Table 7 is simulated using AVISPA and Scyther separately. For the phases mentioned, the secrecy of messages is verified using both AVISPA and Scyther. In addition, AVISPA also checks the authentication of messages. We achieve the following goals when the protocol from mentioned table is simulated using AVISPA and Scyther.

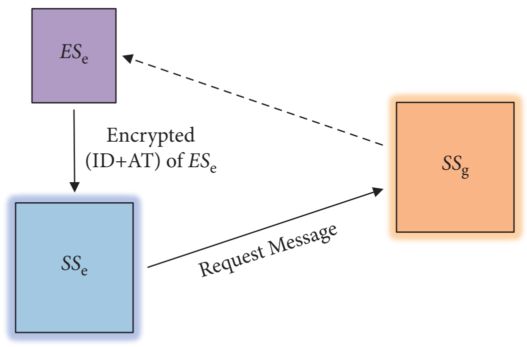


FIGURE 6: Communication between an edge server and a service server of a different cluster.

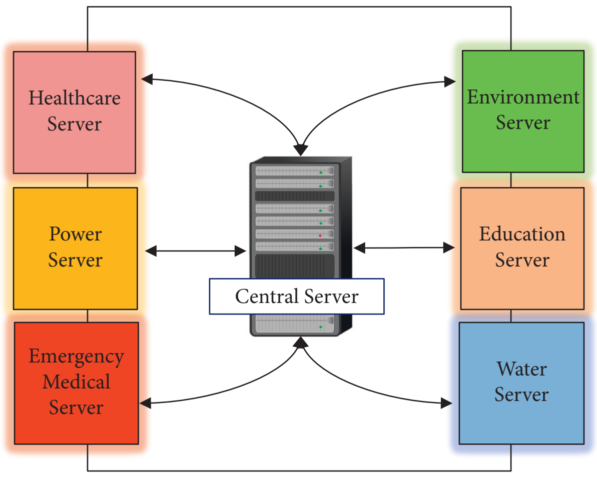


FIGURE 7: A cluster of service servers and the central server.

(1) Secrecy of ID, Pwd, and *AMes* using AVISPA and Scyther when $ES_j$ and $ES_i$ pass messages between them

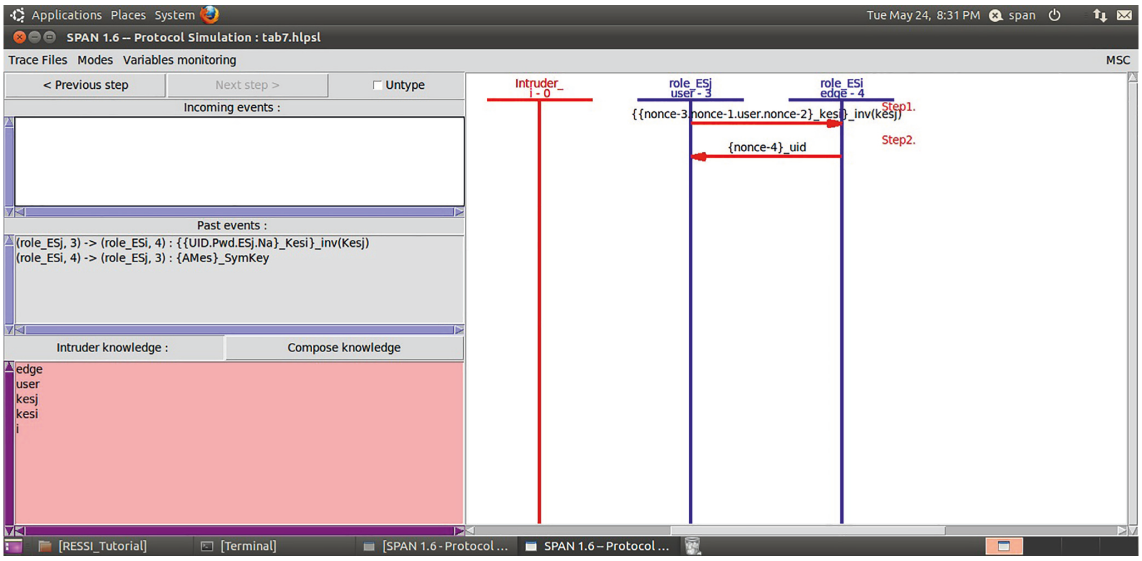
(2) Authentication of the sender using AVISPA when ID and Pwd of user sent from $ES_j$ from $ES_i$

(3) No attack by an intruder when checking authentication using AVISPA when (ID + Pwd) of user are sent by $ES_j$

When $ES_j$ sends *PEes* to $ES_i$, it is encrypted using the public key of $ES_i$ and the private key of $ES_j$. The target is to ensure the authenticity of the sender and maintain the secrecy of the messages within the cipher. Figures 8 and 9 show that AVISPA and Scyther verify the secrecy of plain texts (ID and Pwd), respectively, inside *PEes* cipher. At this phase, the receiver receives the cipher and decrypts using the public key of the sender and the private key of the receiver. Authentication of the sender at the same phase is maintained, which is verified by AVISPA, and Figure 8 shows the result. In the same figure, AVISPA also verifies that due to this authentication process no intruder can attack. On the other hand, when $ES_i$ sends *AMes* to $ES_j$, the secrecy of AMes is maintained in both sender and receiver ends. At this phase, the sender encrypts the message using a symmetric key

Table 7: Dialogue for user authentication.

| User (Ui) | Edge server $(ES_i)$ | Edge server $(ES_j)$ |
|---|---|---|
| 1. Login to $ES_i$ with (ID, Pwd, AT) | 1.1 Finds AT and authenticates $U_i$ | |
| 2. Sends log-in request to $ES_j$ | | 2.1 Asks (ID + Pwd) of $U_i$ and ID of $ES_i$ |
| 2.2. Sends (ID + Pwd) of $U_i$ and ID of $ES_i$ | | 2.3 Received necessary information |
| | | 3. Encrypts (ID + Pwd) of user and ID of $ES_j$ with a nonce using the public key of $ES_i$ and private key of $ES_j$ and sends *PEes* to $ES_i$ to authenticate $U_i$ |
| | 3.1 Decrypts PE the public key of $ES_j$ and private key of $ES_i$ and finds an authentication request | |
| | 3.2 Authenticates $U_i$, encrypts *AMes* by symmetric key ($ES_i$, $ES_j$), and sends encrypted message *AMes* to $ES_j$ | 3.3 *AMes* is received and decrypts by symmetric key ($ES_i$, $ES_j$) and authenticate $U_i$ |



```
(role_ESj, 3) -> (role_ESi, 4) : {{UID.Pwd.ESj.Na}_Kesi}_inv(Kesj)
(role_ESi, 4) -> (role_ESj, 3) : {AMes}_SymKey
```

```
edge
user
kesj
kesi
i
```

Figure 8: Simulation result of the protocol in Table 7 using AVISPA verifies secrecy and authentication.

shared by the sender and receiver. AVISPA and Scyther both verify this result in Figures 8 and 9.

Verification of phases 1, 1.1, 2, and 2.1 in Table 8 is simulated using AVISPA and Scyther separately. For all the phases mentioned, the secrecy of messages is verified using both AVISPA and Scyther. In addition, AVISPA also checks the authentication of messages. We achieve the following goals when the protocol from the mentioned table is simulated using AVISPA and Scyther.

(1) Secrecy of *ATes* and *PEss* using AVISPA and Scyther when $ES_e$, $SS_e$, and $SS_g$ pass messages among them

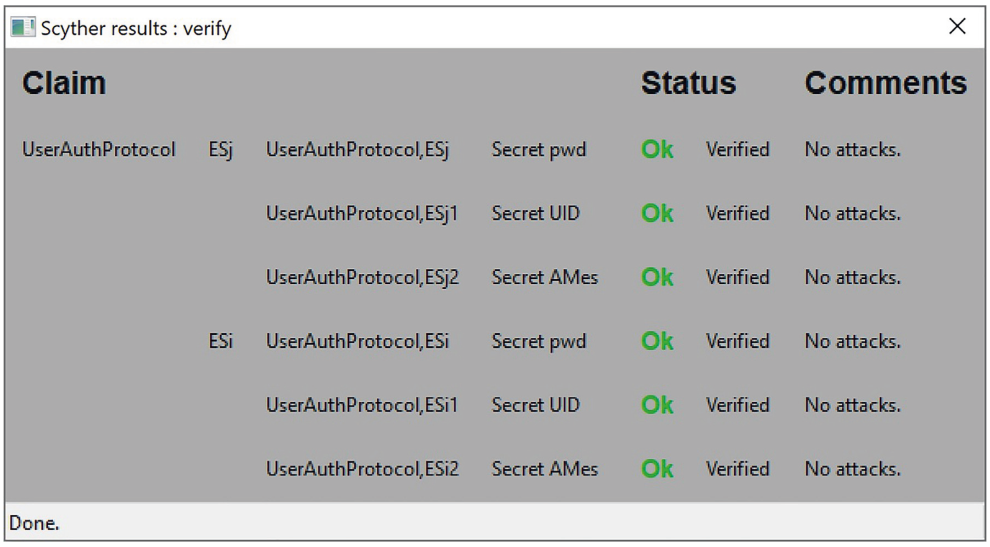
(2) Authentication of the sender using AVISPA when *PEss* is sent from $ES_e$ to $SS_g$

(3) No attack by intruder occurs when checking authentication using AVISPA when *PEss* is sent by $ES_e$

When $ES_e$ sends a service request to $SS_e$, the secrecy of the message checked from sender and receiver sides, both AVISPA and Scyther in Figures 10 and 11, respectively, show that secrecy of *ATes* is maintained. At this phase, the message is encrypted using a symmetric key shared between sender and receiver. On the other hand, the secrecy of *PEss* is maintained when $ES_e$ sends an encrypted message to $SS_g$. At this phase, the sender encrypts the message using the private key of $ES_e$ and $SS_g$ decrypts the cipher using the public key of $SS_e$. This result of secrecy is verified by AVISPA and Scyther, which are shown in above mentioned figures. In addition, authentication of the sender of PEss also verified by AVISPA, which is depicted in Figure 11.

## 3. Comparison and Discussion

In this paper, based on some predefined comparison criteria, we present a qualitative comparison between the proposed cluster-based authentication process and some other similar processes that involves authentication. We consider the following contenders in the comparison spectrum.



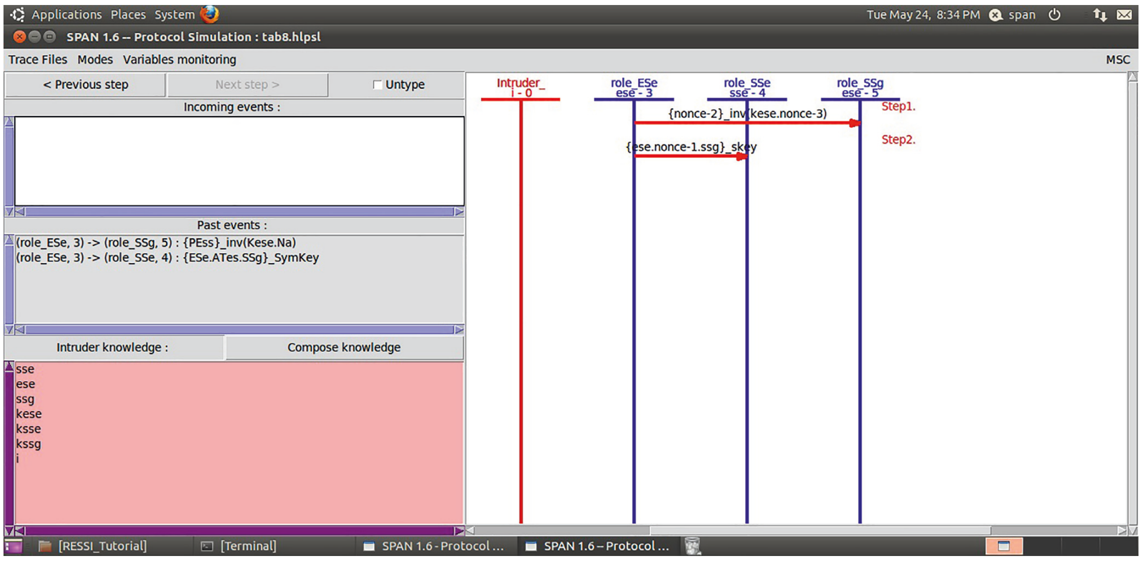
```
Scyther results : verify

Claim                                                      Status              Comments
UserAuthProtocol  ESj  UserAuthProtocol,ESj   Secret pwd   Ok    Verified      No attacks.
                       UserAuthProtocol,ESj1  Secret UID   Ok    Verified      No attacks.
                       UserAuthProtocol,ESj2  Secret AMes  Ok    Verified      No attacks.
                  ESi  UserAuthProtocol,ESi   Secret pwd   Ok    Verified      No attacks.
                       UserAuthProtocol,ESi1  Secret UID   Ok    Verified      No attacks.
                       UserAuthProtocol,ESi2  Secret AMes  Ok    Verified      No attacks.
Done.
```

Figure 9: Simulation result of the protocol in Table 7 using Scyther verifies secrecy.

Table 8: Dialogue for getting service from the different cluster master.

| Cluster member ($ES_e$) | Cluster master ($SS_e$) | Cluster master ($SS_g$) |
|---|---|---|
| 1. Sends encrypted service request using E ((IDes ‖ ATes), ID of SSg) by symmetric key ($ES_e$, $SS_e$) | 1.1 Decrypts the encrypted request and finds IDes, ATes, and authenticate $ES_e$ | |
| 2. Creates an encrypted message (PEss) using the private key of $SS_e$ and a nonce | | 2.1 Receives encrypted message and decrypts using the public key of $SS_e$<br>2.2 Accepts service request |



```
Past events :
(role_ESe, 3) -> (role_SSg, 5) : {PEss}_inv(Kese.Na)
(role_ESe, 3) -> (role_SSe, 4) : {ESe.ATes.SSg}_SymKey

Intruder knowledge :
sse
ese
ssg
kese
ksse
kssg
i
```

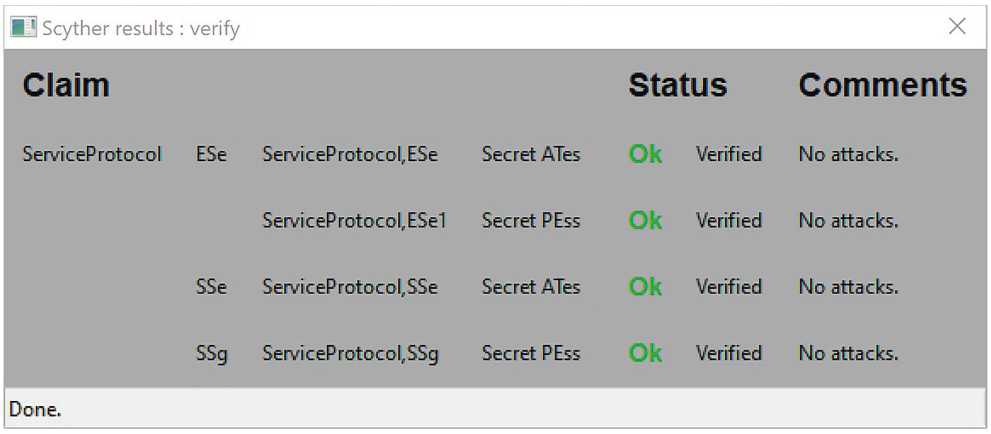
Figure 10: Simulation result of the protocol in Table 8 using AVISPA verifies secrecy and authentication.

(i) Contender 1: secure remote user authenticated key establishment protocol for smart home environment [21]

(ii) Contender 2: effective authentication for restricting unauthorized user [22]

(iii) Contender 3: a secure remote user authentication scheme for smart cities e-governance applications [23]

(iv) Contender 4: a secure IoT architecture for smart cities [24]



```
Scyther results : verify

Claim                                                              Status          Comments

ServiceProtocol   ESe   ServiceProtocol,ESe    Secret ATes   Ok   Verified   No attacks.

                        ServiceProtocol,ESe1   Secret PEss   Ok   Verified   No attacks.

                  SSe   ServiceProtocol,SSe    Secret ATes   Ok   Verified   No attacks.

                  SSg   ServiceProtocol,SSg    Secret PEss   Ok   Verified   No attacks.

Done.
```

Figure 11: Simulation result of the protocol in Table 8 using Scyther verifies secrecy.

From Table 9, we see that the proposed cluster-based authentication process uses more secured public key encryption system with a framework of key hierarchy. Besides, together with trust evaluation, the proposed cluster-based authentication process can provide a shield against DoS and Insider Attack.

We now evaluate the proposed cluster-based authentication process in terms of execution time(s). Table 10 shows the mean execution times of generating authentication numbers (ANcs, ANss, ANes, and ANu), secret keys (Ksec-u, Ksec-es, Ksec-ss, and Ksec-cs) and authentication tokens (ATu, ATes, and ATss) using PHP as a scripting language. We conduct five rounds of executions for each of the authentication numbers, secret keys, and authentication tokens, and then calculate the mean of execution times, where values of IDs (e.g., IDcs, IDss, IDes and IDu) and password are assumed to be six characters long for the abovementioned generation processes. For the same generation processes, Apache server time is used for time stamp (e.g., TSu, TSes, TSss, and TScs) values as we use PHP for above experiment. PHP script generates the abovementioned time stamp values within in Apache server using an Intel Core i5 CPU with 2.30 GHz processor and 8 GB RAM. We use PHP benchmark tool to record the execution times in microseconds. Among the first set that shows authentication number generation times; generation of ANss takes the highest time, where the fastest one is ANcs. In the second set, which shows secret key generation times, close competitors are Ksec-u and Ksec-cs with almost equal time. In the same set, the slowest time is taken by Ksec-es to be generated. In the third set, ATss takes the slowest time among three where ATes is the fastest to be generated. All these keys mentioned in Table 10 are used in variety of dialogues, next paragraphs show detailed comparisons.

Next, we compare the secret key generation times using variety of hashing algorithms required for "dialogues of different entities," which is shown in Figure 10. We use MD5, SHA256, SHA384, SHA512, and Snefru as hashing algorithms to generate secret keys (Ksec-u, Ksec-es, and Ksec-ss) as shown in Table 6. We calculate the total time required to generate secret keys (shown in Table 6) using each of the hashing algorithms (see Figure 12). Our experiment shows that SHA384 and SHA512 perform the best (takes less time) among all the algorithms used, whereas SHA256 and Snefru take more time in execution, and MD5 takes moderate time compared to all others.

Figure 13 shows the comparison of total time taken for both encryption and decryption using variety of key length of RSA and elliptic curve cryptography (ECC). We use these two asymmetric algorithms for "dialogue for user authentication." We execute all encryption and decryption shown in Table 7 with 1024-, 2048-, and 3072-bit keys separately for RSA, and for the same operations using ECC, we use 160-, 224-, and 256-bit keys. These are the comparable key sizes of these two asymmetric algorithms considering security strengths [25]. For example, 256-bit ECC key will provide same level of security as 3072-bit key of RSA. The figure shows total time required for encryption and decryption with RSA always faster than ECC in all the test cases above. It is shown in [25] that key generation time of ECC is faster than RSA as ECC uses shorter keys to provide same level of security strength. Hence, if encryption and decryption take place more frequently than the key generation process, then use of RSA is a good choice. However, when the situation is opposite, the use of ECC is a better choice.

Figure 14 shows the comparative result of total execution times for encryption and decryption required for the steps shown in Table 8. These encryption and decryption processes require combination of asymmetric and symmetric keys both. We calculate the total time for two combinations asymmetric and symmetric keys: one set contains RSA and AES with variety of key length and another set contains ECC and AES, where RSA and ECC are asymmetric algorithms and AES is a symmetric one. We use variety of key lengths for each of the sets. RSA with 512-bit key and AES with 128-bit key combination is the fastest in terms of execution time than all other pairs of RSA and AES combination, whereas the combination of RSA (2048) and AES (256) is the slowest in terms of execution time among all the combinations. On the other hand, combination of ECC and AES with variety of key lengths encrypts and decrypts even faster than all the

Table 9: Comparison between the proposed cluster-based authentication process and some other similar processes.

| Comparison Criteria | Contender 1 | Contender 2 | Contender 3 | Contender 4 | The proposed cluster-based authentication process |
|---|---|---|---|---|---|
| 1. Applicability in smart city paradigm | Yes | Yes | Yes | Yes | Yes |
| 2. Applicability in edge computing paradigm | No | No | No | No | Yes |
| 3. Applicability in device level | No | Yes | No | No | No |
| 4. Use of authentication token | No | No | No | No | Yes |
| 5. Use of one time password | No | Yes | No | No | No |
| 6. Hierarchical key management | No | No | No | Yes | Yes |
| 7. Type of encryption | Symmetric key | Symmetric key | Symmetric key | Symmetric key | Public key |
| 8. Algorithm detail | AES-CBC | AES-512 | AES-512 or AES-128 | AES-512 or AES-128 | RSA or Elliptical Curve Cryptography |
| 9. Protection against denial of service (DoS) attack | Yes | No | Yes | No | Yes |
| 10. Resistance to insider attack | No | No | Yes | No | Yes |
| 11. Trust evaluation on user activity | No | No | No | No | Yes |
| 12. Low latency with high availability | No | Not applicable | No | No | Yes |

Table 10: Mean execution time.

| Numbers, keys, and tokens | Mean execution time |
|---|---|
| ANcs | $1.46 \times 10^{-5}$ |
| ANss | $3.24 \times 10^{-6}$ |
| ANes | $2.34 \times 10^{-6}$ |
| ANu | $3.00 \times 10^{-6}$ |
| Ksec-u | $2.96 \times 10^{-6}$ |
| Ksec-es | $2.72 \times 10^{-6}$ |
| Ksec-ss | $2.81 \times 10^{-6}$ |
| Ksec-cs | $2.96 \times 10^{-6}$ |
| ATu | $2.77 \times 10^{-6}$ |
| ATes | $2.57 \times 10^{-6}$ |
| ATss | $2.96 \times 10^{-6}$ |

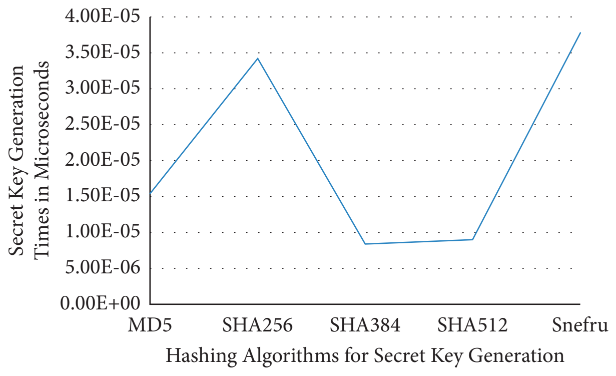


Figure 12: Total secret key generation times using variety of hashing algorithms for "dialogues of different entities."

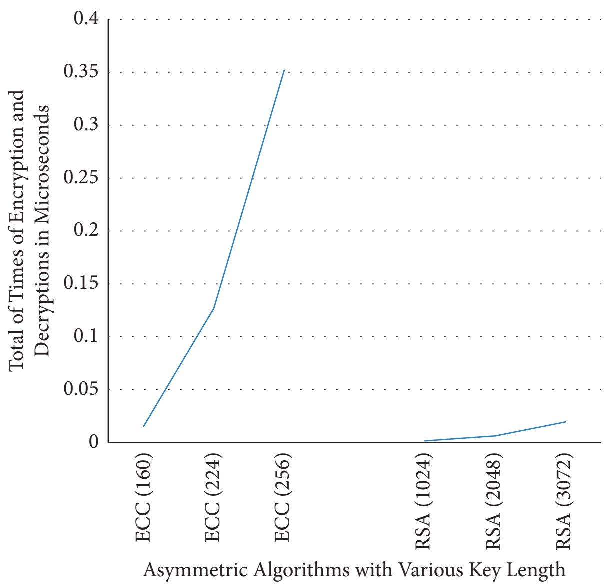


Figure 13: Secret key generation times using variety of key length (1024, 2048, and 3072 bits) in RSA for asymmetric encryption measured for "dialogue for user authentication."

combinations of RSA and AES. The combination of ECC and AES has a decreasing trend in total encryption and decryption time as the key length increases for each execution. For example, ECC with 256-bit key and AES with 256-bit key show the fastest total of encryption and decryption time.

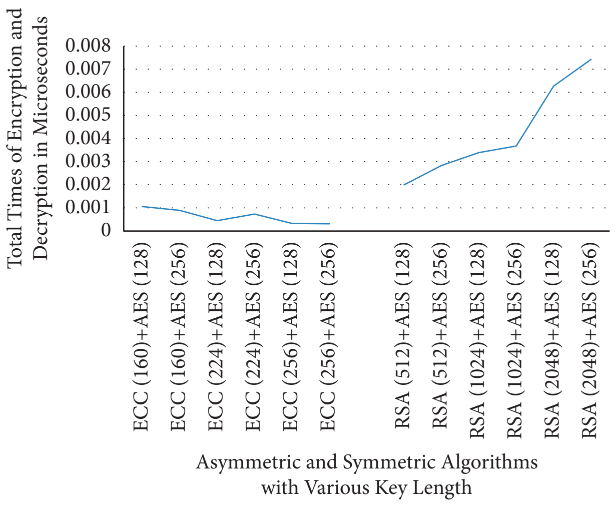


Figure 14: Total encryption and decryption times using variety of key lengths of RSA (asymmetric), ECC (asymmetric), and AES (symmetric) for "dialogue for getting service from the different cluster master."

## 4. Conclusion

In this paper, we have proposed a cluster-based authentication process for the users, edge servers, and service servers, which are engaged in storing, processing, and accessing data. All the necessary processes have been presented using necessary figures and tables. A trust evaluation process is used to evaluate the user trusts before and after the registration to control the activities of the members in the system. The proposed trust evaluation process can be further augmented through fusion of soft information with already existing hard information. Soft information for the proposed trust evaluation process can be collected from three different kinds of platforms such as authority's complaint platform, users' review platform, and social media platform. Moreover, simulation of the proposed trust evaluation process has not been conducted for time-advance mechanism using a large number of alternative system configurations. In the future, we intend to extend our work by exploiting the aforementioned scopes. Also, we shall extend our work by comparing with some similar ones such as [26,27].

The proposed authentication process is secured enough in the context of a smart city. As such, the authentication token of a user is simulated using the secret key and secret of the generating entity (edge server). Even if the secret key is compromised, authentication token cannot be generated without the secret of the edge server. Similarly, authentication tokens of other entities are secured according to the proposed method. We have done simulations of the protocol using AVISPA and Scyther. The simulation results have shown that the sessions in the process are safe.

## Data Availability

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 upon reasonable request.

## Conflicts of Interest

All authors declare that they have no conflicts of interest.

## References

[1] T. Harwood, "Smart city infographic," 2020, https://www.postscapes.com/anatomy-of-asmart-city/.
[2] J. Kaur, A. Agrawal, and R. A. Khan, "Security issues in fog environment: a systematic literature review," *International Journal of Wireless Information Networks*, vol. 27, no. 3, pp. 467–483, Sep 2020.
[3] M. Peng and K. Zhang, "Edge computing technologies for internet of things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.
[4] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: a comprehensive survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10200–10232, 2020.
[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.
[6] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.
[7] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: state of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.
[8] E. Ismagilova, L. Hughes, P. Nripendra, K. Yogesh, and Y. Dwivedi, "Security, privacy and risks within smart cities: literature review and development of a smart city interaction framework," *Information Systems Frontiers*, vol. 89, 2020.
[9] R. A. Khan, S. U. Khan, and H. U. M. Khan, "Systematic mapping study on security approaches in secure software engineering," *IEEE Access*, vol. 9, pp. 19139–19160, 2021.
[10] R. A. Khan, S. U. Khan, and H. U. M. Khan, "Systematic literature review on security risks and its practices in secure software development," *IEEE Access*, vol. 10, pp. 5456–5481, 2022.
[11] A. S. Elmaghraby and M. M. Losavio, "Cyber security challenges in smart cities: safety, security and privacy," *Journal of Advanced Research*, vol. 5, no. 4, pp. 491–497, 2014, Cyber Security.
[12] Y. Ruan, A. Durresi, and L. Alfantoukh, "Trust management framework for internet of things," in *Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 1013–1019, Crans-Montana, Switzerland, March 2016.
[13] P. Wang and P. Zhang, "A review on trust evaluation for internet of things," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, vol. 16, pp. 34–39, Brussels, BEL, June 2016.
[14] A. Boukerche and Y. Ren, "A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks," in *Proceedings of the 5th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN '08*, pp. 88–95, Association for Computing Machinery, New York, NY, USA, November 2008.

[15] Y. Liu, K. Li, Y. Jin, Y. Zhang, and W. Qu, "A novel reputation computation model based on subjective logic for mobile ad hoc networks," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 547–554, 2011.

[16] M. N. Adnan, P. A. Forest and M. Z. Islam, Forest PA: c," *Expert Systems with Applications*, vol. 89, pp. 389–403, 2017.

[17] M. Bewong and M. Z. Islam, "BDF: a new decision forest algorithm," *Information Sciences*, vol. 569, pp. 687–705, 2021.

[18] S. Schechter, T. Parnell, and A. Hartemink, "Anonymous authentication of membership in dynamic groups," in *Financial Cryptography Financial Cryptography*, M. Franklin, Ed., vol. 3, pp. 184–195, Springer Berlin Heidelberg, Berlin, Germany, 1999.

[19] S. Nisha and M. Farik, "Rsa public key cryptography algorithm - a review," *Proceedings of the IEEE*, vol. 6, no. 7, pp. 187–191, 2017.

[20] I. Chatzigiannakis, A. Vitaletti, and A. Pyrgelis, "A privacy-preserving smart parking system using an iot elliptic curve based security platform," *Computer Communications*, vol. 89-90, pp. 165–177, 2016.

[21] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure remote user authenticated key establishment protocol for smart home environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 391–406, 2020.

[22] A. Patil, D. Rana, S. Vichare, and C. Raut, "Effective authentication for restricing unauthorized user," in *Proceedings of the International Conference on Smart City and Emerging Technology*, pp. 1–4, ICSCET), Mumbai, India, January 2018.

[23] G. Sharma and S. Au, "A secure remote user authentication scheme for smart cities e-governance applications," *Journal of Reliable Intelligent Environments*, vol. 3, no. 3, pp. 177–188, 2017.

[24] S. Chakrabarty and D. W. Engels, "A secure iot architecture for smart cities," in *Proceedings of the 13th IEEE Annual Consumer Communications Networking Conference*, pp. 812-813, CCNC), Las Vegas, NV, USA, January 2016.

[25] V. Gupta, S. Gupta, S. Chang, and S. Douglas, "Performance analysis of elliptic curve cryptography for ssl," in *Proceedings of the 1st ACM Workshop on Wireless Security, WiSE '02*, pp. 87–94, New York, NY, USA, September 2002.

[26] S. Khan, S. Nazir, and H. Ullah Khan, "Smart object detection and home appliances control system in smart cities," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 895–915, 2021.

[27] H. Ullah Khan, M. Kamel Alomari, S. Khan et al., "Systematic analysis of safety and security risks in smart homes," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 1409–1428, 2021.