Vol. 08, No. 3 (2026) 1743-1750, doi: 10.24874/PES08.03.026

Proceedings on Engineering Sciences

www.pesjournal.net

EDGE COMPUTING FOR CONTEXT-AWARE ACCESS CONTROL IN IT SYSTEMS

Nitesh Kumar Jangid[1]
Pankaj Dadheech
Mukesh Kumar Gupta

Received 18.05.2026.
Revised 02.07.2026.
Accepted 29.07.2026.

Keywords:

Context-aware access control, Cloud computing, Sensors, Hashing, IoT, IoT cloud services

A B S T R A C T

Edge computing enables local computation close to the sensing environment. This computation may be helpful for context-aware solutions required for IT Systems. An IT System is considered more secure if a context-based role is assigned to a user. Location is one of the interesting contexts for the security of IT Systems. The proposed solution utilizes edge computing to calculate the Zone_token for the group of workstations. The input parameters for Zone_token computation are local parameters like temperature, humidity, image, etc. The computation of the Zone_token enables the computation of the PC-token for each workstation on the edge gateway. The PC-token stored on the edge gateway is updated in the cloud IT System using IoT Cloud Services like AWS IoT and AWS Greengrass Core. The user may require different roles in different zones based on login from zone-specific workstations. An IoT app deployed on each workstation is used to grab the PC-token from the edge gateway so that it may be submitted during login, along with id, pwd to ensure the zone may be decided by the IT System and the corresponding role may be assigned to the user.

© 2026 Published by Faculty of Engineering

1. INTRODUCTION

Public cloud providers, like Amazon Web Services, offers on-demand availability of various resources like compute, storage, database, networking, etc. over the internet. IT systems are being deployed in public clouds from private data centres to reduce IT CAPEX and OPEX costs. However, outside enterprise attacks are multifold for IT systems deployed in the public cloud. Context-aware access control can reduce these attacks.

In the context-awareness, the user may experience different roles based on the location context. For example, the user may be allowed to perform Create-Read-Update-Delete (CRUD) operations from zone-1 and read operations from zone-2, and no operations are allowed in zone-3, as given in Figure 1.

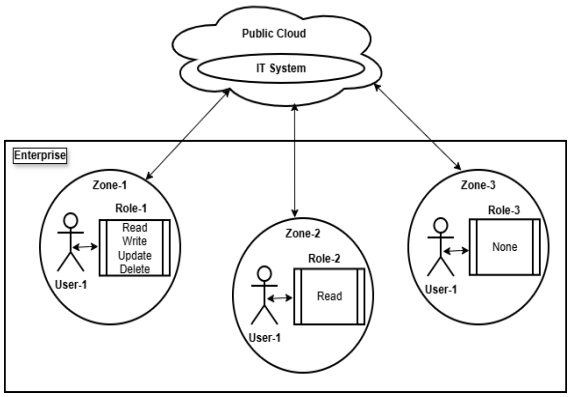


Figure 1. Various roles of user in different zones

[1] Corresponding author: Nitesh Kumar Jangid
Email: nitesh.research@gmail.com

In an enterprise, it is common for users to access the IT system through multiple stable workstations in different locations, i.e., "the user is mobile, but the workstations are stable". Due to the absence of built-in sensors in stable workstations, the user mobility context cannot be traced. However, the growing IoT network in the enterprise may create opportunities to enable context awareness for workstations.

Since Role-Based Access Control(RBAC) is not sufficient due to a lack of context handling, such as time, location, etc. The authors explored existing context-aware access control methods in the research literature. The LRBAC model by Zhang et al. (2006) proposed just-in-time user role activation based on the user's physical location on a mobile device. The LBAC model by Ardagna et al. (2006) proposed location verification services based on SLA parameters, confidence, and timeout to allow user granted or denied access based on user location and credentials. The LAAC protocol by Cho et al. (2006) proposed location-based access for mobile devices based on a location key derived from the overlapping access beacon's information. The DRBAC model by Prayogi et al. (2007) proposed granting roles to the user in a session based on the user's location structure information. The TSRBAC model by Chen et al. (2010) proposed using temporal and spatial entities to constrain permissions on objects. The Location-based access control using smart phones by Zhang et al. (2012) proposed location registration, authentication, authorization, and a location verification mechanism for mobile transactions using smart phones. The LoTAC framework by Androulaki et al. (2014) proposed integrating cloud providers and location infrastructure to enforce location- and time-based access control on cloud data to mitigate data breach incidents. The Special Connector Model by Satoh (2017) proposed a framework for context-aware services executed in pervasive computing environments and cloud computing platforms. The Smart-AC framework by Kashmar et al. (2019) proposed a general access control framework to overcome the limitations of current access control methods to tackle various IT cases. The GSTRBAC model by Lail (2021) proposed restricting cloud resources based on time and location to ensure cloud security. The research work reviewed by Bertino and Kirkpatrick (2009) compares the various location-based authentication and access control methods and identifies open research directions. Cleeff et al. (2010) proposed a theoretical framework for LBAC evaluation, usefulness criteria, usage scenarios and future research directions. Routray and Bera (2023, 2024) proposed a two-phase decryption process using ciphertext-policy attribute-based encryption (CP-ABE) to ensure that a valid user context can send instructions to IoT devices through the cloud. The Wang et al. (2024) research review compare the various static and dynamic access control technologies for IoT-based big data to protect the personal information. The Fugkeaw et al. (2025) proposed a hybrid RBAC-ABAC model to enable dynamic and context-aware access control across diverse and geographically dispersed teams. The above existing works related to context-aware access control for mobile devices only. The GPS receivers for indoor localization were used by Nirjon et al. (2013) by deploying a directional antenna and signal processing technique. Also, the authors Jangid and Gupta (2022) presented a research poster at MobiCom 2022, Australia, on their initial findings and a book chapter by Jangid et al. (2025) in the Lecture Notes in Networks and Systems book series. The following research gaps were identified:

1. Context-aware access control was proposed for mobile devices only.
2. The GPS receiver alone is not sufficient for indoor localization due to inaccuracy.

In the proposed method, all workstations are organized into several zones, and each zone is identified through a Zone_token generated through environmental cryptography. The environmental cryptography uses zone local environmental parameters like temperature, humidity, image, sound, etc, to generate a Zone_token, which is the hashed value of the concatenated local environmental parameters. Hash generation is a cryptographic technique, and when the input parameters are environment-sensed values, this technique is called environmental cryptography. Equation 1 shows the generation of an environmental token.

$$\text{Environment-Token} = \text{Hash}(\text{sensors reading}) \quad (1)$$

Users who want to log in to the IT system must provide UID, PWD, and PC_token so the system can assign a role based on the zone identified through PC_token. The PC_token is generated through an edge device deployed in an enterprise with AWS Greengrass Core. An edge device allows AWS Lambda computation to generate PC_token based on the Zone_token and the PC MAC address. The PC_token with its zone-id is updated into the IT application database using AWS IoT Core cloud services. Since Zone_token is generated periodically at a set interval, both Zone_token and PC_token are dynamic in the proposed system.

2. PROPOSED WORK

The proposed work is explained through new technological development, mathematical modelling, experiments, results and discussion.

2.1 New Technological Development

The proposed work is based on new technological developments such as AWS IoT Cloud Service and Raspberry Pi edge device. The details of both technologies are provided below:

AWS IoT: AWS IoT is a cloud-based platform that enables IoT (Internet of Things) devices to securely communicate with applications and other cloud services, such as DynamoDB and EC2. In our proposed system, we use AWS IoT as the communication layer between the edge device and the AWS cloud. The edge device operates as the IoT client and runs a Python-based application using the AWS IoT Device SDK. The application publishes device-related information using the MQTT protocol by establishing a secure connection with AWS IoT Core. AWS IoT Core provides the device connectivity, message brokering, authentication, authorization, and routing capabilities required for this communication.

Once the Python application starts, it establishes a TLS-secured connection with the AWS IoT Core device endpoint. The connection uses a private key and a root CA certificate. TLS protects communication between the edge device and AWS IoT Core, while the IoT policy associated with the certificate determines which operations the client is authorized to perform. After successful authentication and authorization, the edge device acts as an MQTT publisher. The application publishes the generated JSON message to the MQTT topic such as sdk/test/python.

In addition to delivering messages to MQTT subscribers, AWS IoT Core can process MQTT messages through the Rules Engine. Rules are evaluated against MQTT topic streams and can filter, transform, or route incoming data to other AWS services. AWS IoT rules can route data to services such as Amazon DynamoDB, Amazon S3, Amazon SNS, Amazon SQS, AWS Lambda, Amazon Kinesis, and other AWS services. We have used Amazon DynamoDB to store the data. The rules extract the relevant information and fields from the MQTT message, including PC_token and Zone-ID, and forward them to DynamoDB. The proposed DynamoDB structure uses PC_token as the partition key, allowing different PC_token records to be stored as separate items

Raspberry Pi: The Raspberry Pi is a small, low-cost single-board computer (SBC) that operates on Raspberry Pi OS, a Linux-based operating system (OS) developed to provide accessible computing capabilities in a small physical form. It is not just a microcontroller but is a fully functioning Single-Board Computer. It can execute a complete operating System and multiple applications concurrently.

The Raspberry Pi can function as an IoT endpoint, edge computer, or gateway depending on the system architecture. In a simple IoT implementation, sensors are connected directly to the Raspberry Pi, which periodically collects measurements and processes them using an application. The resulting data can then be transmitted to a remote server or cloud platform through Ethernet or wireless networking.

The ability to execute applications locally makes Raspberry Pi suitable for edge computing. In traditional cloud-centric architectures, devices may transmit large amounts of data to centralised servers for processing. With edge computing, part of the computational workload moves closer to the data source. A Raspberry Pi can therefore perform tasks such as:

- Sensor data acquisition
- Data filtering and preprocessing
- Local event detection
- Protocol conversion
- Temporary data storage
- Local decision-making
- Computer vision processing
- Communication with cloud services

Raspberry Pi can be integrated with cloud IoT platforms to form a complete device-to-cloud architecture. In such a system, the Raspberry Pi represents the physical computing layer, while the cloud platform provides device management, secure communication, data processing, storage, and application integration.

2.2 Mathematical Modelling

The proposed work uses IoT and IoT Cloud Services, as explained in a book by Kurniawan (2018), to create context awareness for stable workstations. As shown in Figure 2, a group of stable workstations are organized into multiple zones. A Zone_token for each zone is generated by sensing local environmental parameters such as temperature, humidity, sound, and images through various sensors. For example, a token for zone-1 is created using sensors S_1, S_2, S_3, and S_4, and an IoT Device. IoT Device compute Zone_token and sends it to the edge device deployed with AWS Greengrass Core.

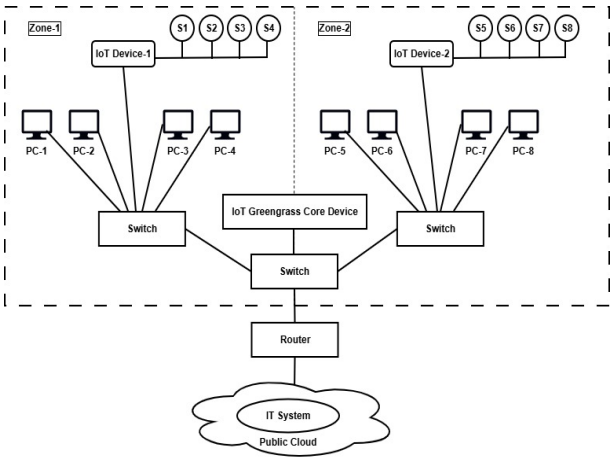


Figure 2. System Design with Zones, Sensors, IoT Devices and Edge Device Deployment

The Zone_token is computed as given in equation 2:

$$Zone_token = Hash(concat(S_1, S_2, S_3, ..., S_n)) \quad (2)$$

At any instance of time, the generated Zone_token is unique among all zones. An IoT application deployed on each stable workstation communicates with AWS IoT Greengrass Core and calls the AWS Lambda function with the workstation MAC address as a parameter. Based on the workstation MAC address and associated Zone_token, the edge device computes PC_token and returns it to the workstation on demand. The PC_token is computed as given in equation 3:

$$PC_token = Hash(concat(Workstation\ MAC\ Address\ , Zone_token)) \quad (3)$$

Some configuration is required on the Edge Device, such as Zone-ID, Zone_Token, and the Workstation MAC Address. The Zone_token for the Zone-ID may be recomputed after a set interval to make the system dynamic. Once the Zone_token is generated, it updates the PC_token and Zone-ID in the IT System deployed in the public cloud. The IT system also stores each user's roles in different zones. A user who wants to log in to the IT system generates a PC_token using the IoT application and enters UID, PWD, and PC_token during login, with an optional biometric parameter.

2.3 Experiment and Results

The proposed system data flow approach is shown in Figure 3 below. The IoT Zone device computes the Zone_token from various sensor inputs and submits it to the edge gateway. The edge gateway updates the Zone_token in the configuration table for the zone ID. Also, PC_tokens in that zone ID are updated into the configuration table on the edge gateway for zone PCs. Updated PC_tokens are synced into the IT System configuration database through the AWS IoT cloud service. The user launches the IoT app deployed on a PC to get PC_token, and the same needs to be supplied during login along with user ID and password. Based on the supplied PC_token, the system identifies the zone ID and assigns the role for that zone only.

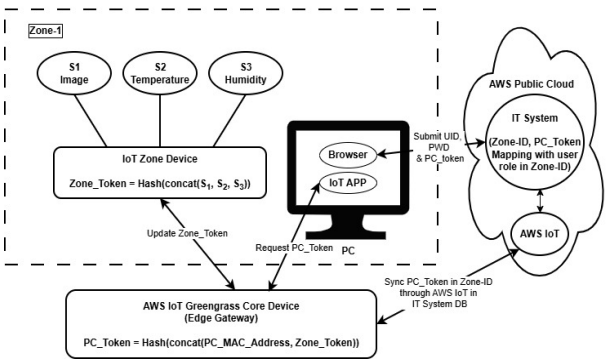


Figure 3. Proposed System Data Flow Approach

The system is planned to be deployed in a phase-wise manner. In the first phase, mathematical modelling and development of the test device are planned to test the Zone_token and PC_token. In the second phase, the system will be tested in a real environment, and the AWS IoT cloud service shall be integrated into the IT System using an edge device and a zone device. This paper covers only the work completed in the first phase. The second phase is in progress.

LOC-MAC-HASH Zone Device is developed to sense temperature, humidity and image of the zone to generate the Zone_token. The device can generate a PC_token for the supplied workstation MAC address. The device is built using a Raspberry Pi, an LCD Display, Temperature and Humidity Sensor, and a Camera Module, as shown in Figure 4 and Figure 5. A USB Keyboard is required to input the user's choice and the workstation MAC Address. The software is developed using Python. A case for the device is developed using a 3D printer. The Raspberry Pi (R Pi) has 40 GPIO pins for interfacing.

Circuit Description:

DHT11-Temperature and Humidity Sensor pin connectivity is given below:

1. \+ pin is connected to the R Pi GPIO pin 2
2. Out pin is connected to the R Pi GPIO pin 7
3. – pin is connected to the R Pi GPIO pin 9

Raspberry Pi Camera Module 2 is connected to the CSI Camera connector.

The JHD204A 20x4 Alphanumeric LCD display is connected to the I2C LCD Interface module to interface with the Raspberry Pi. The I2C LCD Interface module pin connectivity is given below:

1. SCL pin connected to the R Pi GPIO pin 5
2. SDA pin is connected to the R Pi GPIO pin 3
3. VCC pin is connected to the R Pi GPIO pin 4
4. GND pin is connected to the R Pi GPIO pin 6



Figure 4. LOC-MAC-HASH Zone Device generating Zone_token

A Python program is embedded in the device, which performs the following activities in the local environment in sequence:

1. Sense the temperature
2. Sense the humidity
3. Capture the image
4. Generate the Zone_token
5. Prompt the user to enter the MAC address
6. Generate the PC_token on the MAC address entry

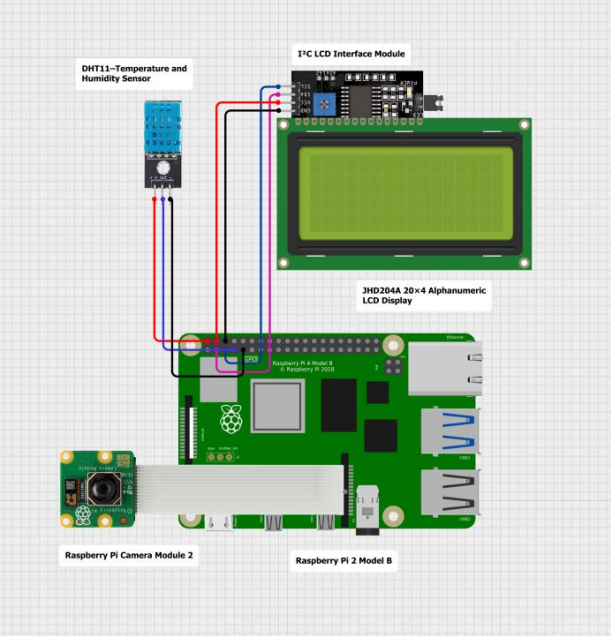


Figure 5. LOC-MAC-HASH Zone Device Circuit Diagram

The program uses the SHA3-256 algorithm to generate the tokens. The Zone_token and PC_token are generated using the LOC-MAC-HASH Zone device for 4 locations and one workstation at each location as per Table 1.

The following code segment in python is used to create Zone_token and PC_token.

Python Code for Zone_token

```
def getImage():
        image = picam2.capture_array()
        ret, buffer = cv2.imencode('.png', image)
        if not ret:
                print("Failed to encode image")
                exit()
        image_bytes = buffer.tobytes()
        return image_bytes

def encodeHash():
        humidity,temperature=
        Adafruit_DHT.read_retry(sensor, pin)

        dthTempValue = f"{temperature}"
        dthHumidityValue = f"{humidity}"
        image = getImage()
        hasher = hashlib.sha3_256()
        hasher.update(dthTempValue.encode())
        hasher.update(dthHumidityValue.encode())
        hasher.update(image)
        zone_token = hasher.hexdigest()
```

Python Code for PC_token

```
hasher = hashlib.sha3_256()
hasher.update(zone_token.encode())
hasher.update(mac_address.encode())
pc_token = hasher.hexdigest()
```

Table 1. Details of Zone_token and PC_token generated for 4 workstations located in 4 different zones.

Zone#	Zone_token	PC MAC Address	PC_token
Zone-1	17f85e43ba1abc815a7f76abf3b135c7cdaee0700782c111d1b77235b33013ad	3C:2C:30:1A:B4:7F	5c54dbdbb30d54fd8183fdd63832bd3674ebe85cd1b4faf0ae80c391490c0fa2
Zone-2	19ee059c5d679d874562f5635f5d5bad7cd3171bdd64d4ecd577148a5de727e0	D6:79:89:6E:95:B3	5de3db17eb32c031c35a3a1d6da286d73712b770729c3422590e5d050f3be091
Zone-3	fb8a708523b673e8d9bf0be2bf9fc2c06b366150fd70fef7c5894da53010bc9e	C0:25:A5:48:8E:B8	8842ffa86e2e209d4d5fd8e9574da55e908c5db5c51b41c48c9c9cab57ed2272
Zone-4	65aeac5a3a37ca005eab21719630888fa4fc95bc1686360ee35d96ae6727292a1	3A:7F:2C:91:BD:E8	f9a5505d91eab2c8568c7921195dab99be4f2f2a9218dc40b8adc05ea5a39f86

In Table 1 above, it is observed that 4 PCs deployed in 4 different zones will get different PC_token and Zone_token. Workstation MAC address allocation to a specific zone is in the edge device configuration file. This may create the possibility of the following cases being implemented:

1. One workstation in one Zone: If security is to be provided to only one workstation assigned to a user, then the user can access the system role from one PC only.
2. Group of workstations in Zone: If security is to be provided to a group of workstations where a group of people can work in a collaborative manner across different workstations.
3. Virtual Boundary: If security is to be provided to logically divide the groups of workstations into different zones in the same physical room/hall.

4. Physical Boundary: If security is to be provided to the groups of workstations located in different rooms. Each room is considered as a Zone.

Example:
Proposed configuration in MySQL DB: In the IT System, a user may have different roles in different zones as mentioned in Table 2 on entity X.

Table 2. Role permissions on entity X.

Role Name	Entity	Permissions
R1	X	Create
R1	X	Read
R1	X	Update
R1	X	Delete
R2	X	Read

A user may have the following roles in zones as mentioned in Table 3.

Table 3. User role in zones.

User Name	Zone ID	Role
U1	1	R1
U1	2	R2

Proposed configuration in DynamoDB through AWS IoT Core: The following information mentioned in Table 4 is synced into the DynamoDB from the edge device.

Table 4. PC_token and Zone-ID in DynamoDB

PC_token	Zone ID
5c54dbdbb30d54fd8183fdd63832bd3674ebe85cd1b4faf0ae80c391490c0fa2	1
5de3db17eb32c031c35a3a1d6da286d73712b770729c3422590e5d050f3be091	2

Proposed configuration in edge device MySQL DB: Zone-ID and Zone_token are stored in the edge device as mentioned in Table 5. PC-token is generated from Zone_token and MAC Address whenever Zone_token is updated as given in Table 6.

Table 5. Zone-ID and Zone_token in edge device

Zone-ID	Zone_token
1	17f85e43ba1abc815a7f76abf3b135c7cdaee0700782c111d1b77235b33013ad
2	19ee059c5d679d874562f5635f5d5bad7cd3171bdd64d4ecd577148a5de727e0

Table 6. PC_token generated for MAC Address

Zone-ID	MAC Address	PC_token
1	3C:2C:30:1A:B4:7F	5c54dbdbb30d54fd8183fdd63832bd3674ebe85cd1b4faf0ae80c391490c0fa2
2	D6:79:89:6E:95:B3	5de3db17eb32c031c35a3a1d6da286d73712b770729c3422590e5d050f3be091

The edge device will sync zone-id and PC_token to DynamoDB in the AWS cloud through the AWS IoT Core Cloud Service.

During login, the user will provide userid, password and PC_token:

Case-1: When user will provide PC_token 5c54dbdbb30d54fd8183fdd63832bd3674ebe85cd1b4faf0ae80c391490c0fa2, the IT system will detect zone-id as 1 from Dynamo DB and role R1 is assigned to user on entity X in IT System.

Case-2: When the user provides PC_token 5de3db17eb32c031c35a3a1d6da286d73712b770729c3422590e5d050f3be091, the IT system detects zone-id as 2 from DynamoDB, and role R2 is assigned to the user on entity X in the IT System.

2.4 Discussion

It is observed that Zone_token and PC_token are unique for each zone and workstation, respectively. Based on the PC_token and zone-id supplied during login to the IT system, a role may be assigned to the user accordingly. A combination of sensing parameters creates a unique Zone_token at different times. The proposed solution coined the term environmental cryptography for context-aware access control for IT systems. The environmental hash is a good source for indoor localisation without complex and costly hardware. The length of the environmental hash is a good parameter for handling the large number of workstations in a globally distributed enterprise. The solutions not only allow physically separated zones but also allows the creation of virtual zones for groups of workstations. The solution can partition the Wi-Fi access point area into multiple virtual zones for a group of workstations. The solution may be adopted by any number of IT systems, as the generated hashes are confined to each IT system and no hash collisions occur across IT systems. Since the proposed solution is dynamic, a hash collision in an IT system can be handled by re-sensing. It is also observed that images are a good source to generate a unique hash for each sensing. Image sensing is a unique parameter for each location in the universe, which may vary for the same location based on changing light conditions. Computing Zone_token within a finite time interval (10 minutes or 20 minutes) will create different Zone_token and PC_token for the same zone and workstation. Since the system is dynamic enough, the security of the system is high compared to a static system.

3. CONCLUSION

The proposed environmenal cryptography uses hashing to generate Zone_token through local parameters like temperature, humidity, image, etc. This new approach of indoor localization creates an opportunity for IT

system security using context-awareness for user mobility on stable workstations. AWS IoT Greengrass is creating the possibility of local computation of Zone_token and PC_token and easily provides a separate channel to sync PC_token with Zone ID in IT systems deployed in the public cloud through AWS IoT Core Cloud Service. IoT-based context-awareness for IT systems security is modelled to improve the IT system design.

It is observed that Zone_token are unique to each zone and is generated at finite time intervals to ensure system dynamics. Subsequently, the PC_token is generated as a unique, dynamic token. Since a dynamic system is more secure than a static system, the proposed system has potential for adoption in production in the near future. The availability of IoT networks in the enterprise provides a low-cost solution for novel security for stable workstations. In future, the research shall be extended to mobile devices as well.

Acknowledgement: We sincerely acknowledge the Department of IT and Communication, Government of Rajasthan, Swami Keshvanand Institute of Technology, Management and Gramothan and Rajasthan Technical University, Kota for their continuous encouragement for this research work.

References:

Androulaki, E., Soriente, C., Malisa, L., & Capkun, S. (2014). Enforcing location and time-based access control on cloud-stored data. 34th IEEE Annual International Conference on Distributed Computing Systems, 637–648. https://doi.org/10.1109/ICDCS.2014.71

Ardagna, C. A., Cremonini, M., Daimiani, E., Vimercati, S. D. C. D., & Samarati, P. (2006). Supporting location-based conditions in access control policies. International ACM Symposium on Information, Computer and Communications Security (ASIACCS '06), 212–222. https://doi.org/10.1145/1128817.1128850

Bertino, E., & Kirkpatrick, M. (2009). Location-aware authentication and access control - concepts and issues. International Conference on Advanced Information Networking and Applications (AINA), 10–15. https://doi.org/10.1109/AINA.2009.50

Chen, H., Wang, S., Wen, J., Huang, Y., & Chen, C. (2010). A generalized temporal and spatial role-based access control model. *Journal of Networks, 5*(8), 912–920. https://doi.org/10.4304/jnw.5.8.912-920

Cho, Y., Bao, L., & Goodrich, M. T. (2006). LAAC: a location-aware access control protocol. 3rd International Conference on Mobile and Ubiquitous Systems – Workshops (MobiQuitous). 1–7. https://doi.org/10.1109/MOBIQW.2006.361757

Cleeff, A. V., Pieters, W., & Wieringa, R. (2010). Benefits of location-based access control: a literature study. International Conference on Cyber, Physical and Social Computing and Green Computing and Communications, 739–746. https://doi.org/10.1109/GreenCom-CPSCom.2010.148

Fugkeaw, S., Rattana, A., Jongvijitkun, C., & Nongya, P. (2025). AC-devchain: an efficient context-aware access control in devsecops using blockchain. 17th International Conference on Knowledge and Smart Technology (KST), 346–352. https://doi.org/10.1109/KST65016.2025.11003315

Jangid, N. K., & Gupta, M. K. (2022). Location-aware IT system security using IoT in multizone. 28th International Conference on Mobile Computing And Networking (MobiCom '22), 847–849. https://doi.org/10.1145/3495243.3558258

Jangid, N. K., Dadheech, P., & Gupta, M. K. (2025). LOC-MAC-HASH: A novel approach to generate environmental tokens for IT system security. AI Technologies for Information Systems and Management Science. *Lecture Notes in Networks and Systems, 1479*, 33–44. https://doi.org/10.1007/978-3-031-95017-9_4

Kashmar, N., Adda, M., Atieh, M., & Ibrahi, H. (2019). Smart-AC: a new framework concept for modeling access control policy. *Procedia Computer Science, 155*, 417–424. https://doi.org/10.1016/j.procs.2019.08.058

Kurniawan, A. (2018). Learning AWS IoT. Packt Publishing, Birmingham, UK. ISBN: 9781788396110

Lail, M. A. (2021). Towards cloud-based software for incorporating time and location into access control decisions. 26th International Symposium on Access Control Models and Technologies (SACMAT '21), 55–57. https://doi.org/10.1145/3450569.3464395

Nirjon, S., Liu, J., Priyantha, B., & Dejean, G. (2013). High-sensitivity cloud-offloaded instant GPS for indoor environments. 11th International Conference on Embedded Networked Sensor Systems (SenSys '13), 1–2. https://doi.org/10.1145/2517351.2517429

Prayogi, A. A., Park, J., & Hwang, E. (2007). Selective role assignment on dynamic location-based access control. International Conference on Convergence Information Technology (ICCIT 2007), 2136–2141. https://doi.org/10.1109/ICCIT.2007.255

Routray, K., & Bera, P. (2023). Context-aware attribute based access control for cloud-based SCADA systems. International Workshop on Enhanced Network Techniques and Technologies for the Industrial IoT to Cloud Continuum (IIoT-NETs '23), 35–40. https://doi.org/10.1145/3609389.3610569

Routray, K., & Bera, P. (2024). Fog-assisted dynamic IoT device access management using attribute-based encryption. 25th International Conference on Distributed Computing and Networking (ICDCN '24), 346-352. https://doi.org/10.1145/3631461.3631466

Satoh, I. (2017). Spatial connector: mapping access control models for pervasive computing and cloud computing. *Procedia Computer Science*, *110*, 174–181. https://doi.org/10.1016/j.procs.2017.06.075

Wang, T., Fang, W., & Zhang, W. (2024). Access control for IoT-based big data: a state-of-the-art review. 6th International Conference on Big Data Engineering (BDE '24), 81–87. https://doi.org/10.1145/3688574.3688586

Zhang, F., Kondoro, A., & Muftic, S. (2012). Location-based authentication and authorization using smart phones. 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 1285–1292. https://doi.org/10.1109/TrustCom.2012.198

Zhang, H., He, Y., & Shi, Z. (2006). Spatial context in role-based access control. Information Security and Cryptology – ICISC 2006. *Lecture Notes in Computer Science*, *4296*, 166–178. https://doi.org/10.1007/11927587_15

Nitesh Kumar Jangid
Department of Computer Engineering, Swami Keshvanand Institute of Technology, Management and Gramothan,
Jaipur, 302017,
India
nitesh.research@gmail.com
ORCID 0000-0003-2588-0254

Pankaj Dadheech
Department of Computer Engineering, Swami Keshvanand Institute of Technology, Management and Gramothan,
Jaipur, 302017,
India
pankajdadheech777@gmail.com
ORCID 0000-0001-5783-1989

Mukesh Kumar Gupta
National Informatics Centre, Ministry of Electronics and Information Technology
New Delhi, 110003,
India
drmukesh5766@gmail.com
ORCID 0000-0002-4907-0259