Wireless Personal Communications
https://doi.org/10.1007/s11277-022-10017-4

Distributed PEP–PDP Architecture for Cloud Databases

Gaurav Deep[1] · **Jagpreeet Sidhu**[1] · **Rajni Mohana**[1]

Accepted: 29 August 2022

Abstract

Cloud computing allows accessing data from anywhere; Cloud databases play an important role in storing requests for access management. These requests require authorization management which has become a crucial area in access control. The request-response paradigm plays an important role in the PEP–PDP architecture. Many applications are available in literature based on the centralized PEP–PDP architecture. In this architecture, performance degrades with the increase in requests. Failure of PDP increases while handling requests from multiple PEPs. The proposed work extends the existing centralized PEP–PDP architecture to distributed architecture with PEP side caching to achieve scalability. In the proposed architecture, all PEPs employ side caching to improve efficiency. Various simulations and validation checks are performed to validate the architecture. Simulation results show proposed architecture is significantly efficient in handling large requests in contrast to existing single PEP-PDP and multiple PEP-single PEP architectures.

Keywords Insider threats · Policy enforcement point · Policy decision point · Cyber-physical space · Policy access point

1 Introduction

The influence of Cloud Computing has reached every field of life. Everybody is interested in getting benefits from Cloud Computing to reduce the initial investment cost; wider reachability, traffic management, and resource management as per demand are significant benefits [1]. Many old technologies are becoming part of Cloud Computing [2, 3], which leads to an increase in the amount of data travelling worldwide per second [4]. Therefore, data management plays a vital role in maintaining data privacy [5, 6]. Hypervisors [7] help run cloud computing functions at the back end. There are many live examples available,

---

✉ Jagpreeet Sidhu
jagpreet.pu@gmail.com

Gaurav Deep
deepgaurav48@pbi.ac.in

Rajni Mohana
rajni.mohana@juit.ac.in

[1] Computer Science and Engineering & Information Technology, Jaypee University of Information Technology, Waknaghat Solan, Himachal Pradesh, India

such as Twitter [8], online web-based education [9], Smart Power grids [10], Blockchain [11], etc. Many new fields of research have come up, such as cloud and its security [12, 13], super clouds [14], Mobile cloud computing [15], Databases on the cloud [16], etc.

Cloud Service Provider is responsible for every service they provide to their clients. According to the statistics, 68% of organizations feel vulnerable to insider attacks performed by their employees [17–19]. An insider threat is a threat from an employee of the Cloud service provider. The user's data and information are in the safe custody of the Cloud Service Provider under his responsibility [20]. This issue becomes critical in Cloud Computing as several heterogeneous and homogenous hypervisors operate parallel [21]. Multiple hypervisors are parallel running on the cloud. User data migrate from one hypervisor to another as demand increases or decreases. Keeping data in a safe state becomes significantly tricky. Here, the role of access control comes.

Access Control is a security function that protects shared resources against unauthorised access. The distinction between authorised and unauthorised access is made according to an Access Control Policy [22]. Access Control consists of authentication and authorisation. Authentication is verifying an entity's identity, given its credentials. The entity could be in the form of a person, a computer, a device, or a group of network computers [23].

Various authentication policies are available; single-factor authentication policy to multi-factor authentication policy consists of multiple factors in deciding on an authentic user [24, 25]. An Authorisation represents the right granted to a user to exercise an action (e.g., read, write, create, delete, and execute) on particular objects [23]. Authorization policies can be written in role-based access control (RBAC) [26, 27], attribute-based access control (ABAC) [28, 29], extensible access control markup language (XACML) [30, 31], etc.

A cloud service provider (CSP) receives user requests for accessing data. Once the user gets authenticated, employees managing servers request access authorization policy from the policy enforcement point (PEP). PEP provides an access authorization policy if available; otherwise, the request is transferred to the policy decision point (PDP). PDP allows or rejects the access request according to authorization policies (AP) decided by the service provider (SP) [32–34].

The threat detection architecture of Yaseen et al. [35] suggests using multiple PEPs and a single PDP. This paper suggests improvements in the area of scalability and fault-tolerance of the existing multiple PEP (MPEP) -single PDP architecture (SPDP) [35]. Existing algorithms [36, 37] can find dependencies in PEPs and PDP. The proposed work extends existing algorithms to distributed architecture with PEP side caching. Applications of a single PEP-PDP architecture in multiple areas are discussed in Sect. 2. This Section also discusses research gaps. Section 3 proposes improvements in identified key. Section 4 presents simulation results in contrast to various parameters in the existing architecture. Finally, Sect. 5 concludes the work.

2 Background

PDP plays an essential role in making a final decision on authorization policies. The requirement of managing user authorization is essential for user management [38–40]. The single PEP-PDP architecture works effectively for the fewer number of requests (limited numbers).

In [41], authors have proposed attribute-based encryption on decentralized multi-agent systems working on a common goal. An agent is a particular entity that is independent in collecting data, processing it, and sending it on channels. The advantage of using a multi-agent system is its fault tolerance even after some nodes got failed. In this proposed system, attribute authority trusted centres are registered on the authentication servers by sharing a secret key. Then all the authentication and authorization procedures are carried out between them. The same procedures are followed between attribute authority and agents.

Access policies are available in XACML.PDP evaluates digital policies and Meta policy and decides whether to allow access to an object or not. Firstly, PAP creates new policies; secondly, PIP provides the required data to PDP for making decisions; in the end, PEP implements PDP's rules. The decentralized multi-agent system lags the synchronization speed of the multiple agents.

In [42], the authors have taken privacy control of patient records available at multiple places. The patient record is available at doctors, nurses, insurance companies, pharmacies, and relatives. Authors have stressed using electronic patient health records in public and private clouds to better access control patient records in EPHR. Their work mainly focuses on two levels (1) protecting the shared data between multiple parties and protecting the patient's private data available at his immediate doctors.

Authors have proposed privacy protection based on the healthcare system's access control model in the hybrid cloud. In this model, access to required data is generated by the access requester and transferred to PEP, where PEP gets more details about the subject, action, and environment. This request is transferred to PDP for deciding on an access request according to applicable policies. PAP creates access policies, which get stored in the repository. PAP also contains an access policy and privacy policy. Limitations of the proposed model include dealing with an emergency.

In [43], the authors have proposed an access control framework for cyber-physical space. TAAC model (topology-aware access model) is proposed for better access control. The TAAC model is an extension of the RBAC model. TAAC model integrates physical and cyber access control, making it an adaptive model that also adjusts the user's privileges. This paper also proposes secure policy enforcement, which helps in mitigating insider attacks. According to the historical behavioural data and the current access request, the risk value of the users is calculated. This way, malicious users are restricted from accessing the system.

Access control, framework enforcement, and admin modules work in the cyber-physical space. PEP receives the requests in the first module and sends them to PDP. PDP acts according to the policy stored in PAP. Topology attributes are stored in PIP, and risk attributes are stored in the risk module, which helps the PDP make decisions. In administration, module policies are specified in PAP according to the trust value of each user, stored in the trust repository. Policy constraints are managed in the policy constraint management module. This proposed model does not handle multiple cyber access spaces in a smart city.

In [44], the authors have proposed a new control approach to increase the agility of the electricity grid. Policy-based network management for better management of power and energy networks is suggested. To configure the PBNM system for productive use, the authors have proposed to use text mining to derive connection parameters at the LV level. It also uses Volt-VAr optimization to tune the connection settings at each DER to manage the voltage across all the networks.

This paper also suggests policy-based network management voltage control validation using the PEP-PDP architecture. In this system-specific policies are stored in a policy repository, which PDP further fetches. PDP checks the power factor against the applied policy. Whenever PDP finds bounds available in the policy violated, the system would trigger policy obligation to get updated VVC from the 3-OPF tool.

In [45], the authors have proposed a framework for cloud healthcare recommender service. The healthcare service works on data received from the internet of healthcare things. This service prevents it from data theft and other types of modifications; data is concealed two times before actual storage in the cloud. In this framework, a personal gateway is proposed at the patient side, which does the first level of concealment task.

After concealing, data gets transferred to the concerned fog node. This fog node applies attribute-based encryption on the encrypted health profile received from the personal gateways. This concealment process also covers multiple profiles in every group. The security authority centre is a third party responsible for generating certificates for fog nodes and gateways.

Fog-based middleware is proposed to protect the privacy of the patient's health data, which increases trust. Learning agents built patients' privacy policies according to their IOHT data.PIP works as a privacy preference unit. Policy enforcement point as a privacy checker extracts the encoded rules to make self-acting decisions. Policy agents in the middleware act as PDP controls users' health data disclosure to external services. An agent performs first-level concealment locally, and a global concealment agent does second-level concealment. Overall, trust is calculated by the trust agent.

In [46], the authors have discussed the role of user privacy in an android operating system based on mobile phones. Android applications, on average, request 11.4 permissions, out of which 5.12 directly affect privacy. This number increases with the count of android applications, resulting in more privacy violations.

Overcoming this problem, the authors have proposed a decision support system for writing high-level policies, where the non-technical user can write policies. This DSS is based on content-based recommendations. In these characteristics of the user, searches are saved locally. New objects are immediately proposed to the user based on characteristics stored locally. This system performs well if user behaviour does not change—a learning period is required for a system to perform well. The proposed architecture caper consists of DSS, policies, PEP, and PDP.

In this proposed architecture, whenever any application requests access to one of the user's private information, the request is received by PEP. PEP converts into XACML V3 format and sends it further to PDP. It checks for available policy decisions for acceptance or rejection. If PDP denies the request, the system lacks information regarding the user's preferences. This request is sent to DSS and further asks the users to allow the request or deny it. The new XACML V3 role is created and updated in the policy database if the user allows it. The DSS gets matured enough when the request score reaches a predetermined threshold value. The proposed system can manage user privacy, among other applications, comfortably and better.

In [47], the authors have discussed the fall detection issue of patients by the German data protection law. In the proposed architecture, 12 essential requirements are analyzed to fulfil the requirements of four stakeholder groups, i.e., hospital operators, hospital staff, patients, and legal stakeholders. These 12 essential requirements include detecting falls, intimating nearest nursing staff, preventing misuse of information, and confirming emergencies to provide two-way communication between patients and concerned nursing staff members.

The proposed system works in default mode, assessment mode, and investigation mode. Almost all the cameras are working; the fall detection algorithm processes video data to detect any fall event. No human intervention is allowed; as soon as the algorithm detects any fall event system enters into the second mode, i.e., assessment mode. In this second mode, an alarm message is broadcasted to nearby nursing staff as soon as the algorithm detects fall events from the camera video. The area of alarm message expands if no response is received. All other alarm messages were cancelled when the nursing staff accepted the alarm.

The anonymized video is shown to the nurse for conforming to the emergency. The investigation mode starts after the nurse confirms the emergency from the anonymized video in the last mode of the system. Access to the video stream of the associated camera in an unmodified form is provided to the nurse, which helps decide the need for any medical equipment.

Bi-directional communication between the patient and nurse is provided for better handling of the situation. PEP, PIP, and PDP control the system. Events are intercepted by PEP and forwarded to PDP, which evaluates them against available policies in the PIP unit. The proposed system can provide the best possible solution in detecting falls detection.

In [48], the authors have proposed a COPS-based IPv6 traceback algorithm. In this proposed work, the authors have also proposed a traceback architecture. In this architecture, once the IDS detects a DDOS attack happens on the victim, it generates a request in its local PDP for enforcing a policy. PDP sends a message to all PEPs to check for message paths travelling to the victim through them. PEP, which sends the positive response, PDP starts identifying the out path of the packet from every sender. Simulations were carried out in NS2 and proved the proposed algorithm's effectiveness in identifying the attacker in Ipv6 based network.

In [49], the authors have proposed privacy protection for fog computing and internet of things data based on a Blockchain. In this work, IoTs collected data is forwarded to corresponding edge nodes. The edge node then creates access control attributes and strategies for a specific source and stores them in the Blockchain node associated with the edge node.

This node generates the corresponding chaotic and MLNCML sequences and stores them in a chaotic sequence coding library, which helps encryption. Whenever PEP receives a request for accessing data, it creates an attribute-based access request and sends it to PDP. It decides in consultation with PAP to allow or reject the request.

In [50], the authors have proposed a mechanism to Provide advanced remote medical treatment services through pervasive environments. In this proposed work, body sensors take key parameters and transmit them to the hospital via Mobile phone if the patient is outside his home or via a wireless router in the case of home. If the Doctor is required to check its parameters, PEP-PDP acts as an Authorization architecture, which checks the request's legitimacy. This PEP-PDP architecture manages access to Medical records. Encouraging results were achieved on the test bed.

In [51], the authors have proposed Network-Level Access Control Policy Analysis and Transformation mechanism. This proposed work removes the requirement to apply authorization policy at multiple places such as firewalls, routers, proxies, and application servers by applying at a single place, i.e. at Firewall. PEP-PDP architecture is implemented at Firewall. The proposed work has been able to work by removing the condition clause being a hyper-rectangle and performs well and found applications in other areas such as in identity- and role-based access control applications.

In [52], the authors have proposed a privacy-aware authorization engine for collaborative environments. In this work, PEP-PDP architecture is proposed when Business networking between multiple organizations involves persons working at multiple levels, and information flows between them, violating company policies and legal rules. A request is generated and sent to PEP to control information flow or access between various organizations. PEP acts according to the authorization policy available or may ask for a new one from PDP. In Business networking, every organization is supposed to manage information flow between through PEP-PDP architecture for better control.

Various applications of Single PEP-PDP architecture is shown in Table 1.

In [35], the authors have stressed the importance of insider threats, which are the most vulnerable threat than the outsider threat; whereas an insider knows various protocols and policies of the system, an outsider can steal from the windows to which it can have access. Furthermore, usage of the cloud increases day by day, which requires more insiders to manage it. Current access control management uses the request-response mechanism, which consists of the PEP-PDP architecture. In this architecture, final decisions are taken by PDP and enforced by PEP. Copies of decisions taken are stored inside the PEP cache, ultimately increasing its decision-making efficiency. When any request is received at PEP, it searches for a similar one in the side cache. Whenever a similar request is found, the decision to that request is enforced.

In this paper, the authors have shown how Insiders have an edge in performing data theft in cloud relational databases. Insiders' knowledge base also helps guess data based on various data dependencies in cloud relational databases. Authors have also shown in this paper that cloud relational databases cannot detect attacks based on various inferences.

As shown in Fig. 1, the insider threat aware PEP-side caching architecture (ITACA) is proposed to deal with various insider threats. This architecture's working is also shown in Fig. 2. in the sequence diagram. In this ITACA various PEPs with side caching are connected with a single PDP. Request from insider to access data is received by PEP. Suppose PEP has a similar request decision copy in its cache or can ask its neighbour PEPs at the next level. If a previous decision copy is found, the request is passed to DCP. Each PEP member consists of a dependency-check point. This DCP unit checks every user request for any presence of dependency between existing issued requests. If DCP clears it, then the decision to grant access is issued. Otherwise, if DCP detects some threat, then the request is passed to PDP for further evaluation.

Request to PDP is also sent in the case when no copy of previous decisions is available at the PEP level or with neighbouring PEP. After receiving a request from PEP, the risk level of granting access to the request is analysed based on a knowledgebase of an insider, dependencies among data items, and the lifeline of data items. The authors already gave algorithms for evaluating these parameters in [36], 37]. After checking the risk value, ITDU provides access/denial to the received request. Authors in their work have shown that associating multiple PEPs with side caching with a single PDP is much better than the single PEP-PDP architectures it increases efficiency in handling requests many times.

Multiple PEP-single PDP architectures can handle 1500 maximum requests. It can be noted down from the previous architectures that PDP is working as a stressed architecture when multiple requests are coming from multiple PEPs. There is a limitation on the number of PEPs that a single PDP can manage, which makes the chances of failures high. If at any moment PDP got failed, the whole system would collapse. Therefore, there is a need to extend the existing system on many parameters when handling more requests. On the other hand, chances of improvements are there on front of the number of requests transferred to PDP when the percentage of dependency increases, the number of possible threats when requests increase, and many other parameters.

Table 1 Applications areas of single PEP-PDP architecture

Applications/authors	Nyrkov et al. [41]	Son et al. [42]	Cao et al. [43]	Ryan et al. [44]	Elmisery et al. [45]	Oglaza et al. [46]	Krempel et al. [47]	Amin et al. [48]	Liu et al. [49]	Vassis et al. [50]	Basile et al. [51]	Gogoulos et al. [52]
Decentralized multi-agent systems	**X**											
Electronic patient health record		**X**										
Access control in cyber-physical space			**X**									
Increasing the agility of the electricity grid				**X**								
Cloud-based healthcare recommender service					**X**							
Managing android permissions						**X**						

Table 1 (continued)

Applications/authors	Nyrkov et al. [41]	Son et al. [42]	Cao et al. [43]	Ryan et al. [44]	Elmisery et al. [45]	Oglaza et al. [46]	Krempel et al. [47]	Amin et al. [48]	Liu et al. [49]	Vassis et al. [50]	Basile et al. [51]	Gogoulos et al. [52]
Fall detection system for hospitals							**X**					
Ipv6 Trace back algorithm								**X**				
Privacy protection for fog computing and internet of things									**X**			
Remote medical service through a pervasive environment										**X**		
Network-Level Access Control Policy Transformation											**X**	

Table 1 (continued)

Applications/authors	Nyrkov et al. [41]	Son et al. [42]	Cao et al. [43]	Ryan et al. [44]	Elmisery et al. [45]	Oglaza et al. [46]	Krempel et al. [47]	Amin et al. [48]	Liu et al. [49]	Vassis et al. [50]	Basile et al. [51]	Gogoulos et al. [52]
Privacy-aware authorization engine for collaborative environments												**X**

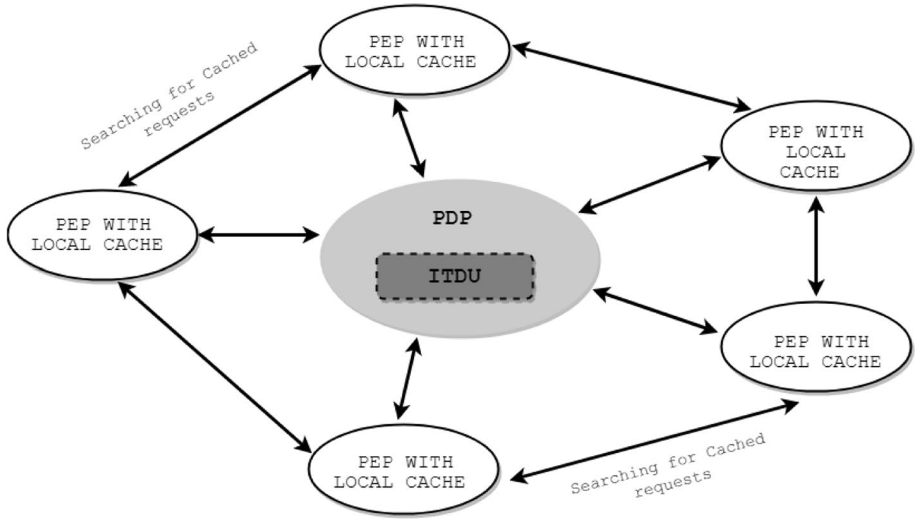


Fig. 1 Insider threat-aware PEP-Side Caching Architecture [35]

3 Proposed Multiple PEP- Multiple PDP Architecture

The scalability issue of the architecture of Yaseen et al. [35] is handled in the proposed distributed authorisation architecture. The proposed architecture is an extension of Yaseen et al. [35]. In distributed authorisation architecture, it is scaled up with multiple PEPs are PDPs to work together. Simulation work was carried out on multiple Virtual machines with test bed configurations as listed in Table 2.

Multiple blocks work in parallel to each other, each consisting of a single PDP and multiple PEP, as shown in Fig. 3. Multiple PDPs can communicate with each other via inter PDP communication network. One block consists of multiple PEP and a single PDP. The working of this block is replicated. In a block, when an insider sends a Data access request to any PEP, the request receiving PEP checks for the availability of Authorisation policy in its local cache. When it is found, dependencies between the user's already-provided data and the requested data are checked in the dependency checkpoint at PEP for threat. If no threat is found, the Authorisation policy allows data to be fetched from resources and provided to the insider.

In another case, when authorisation policy is unavailable at insider request receiving PEP, request for required Authorisation policy is broadcasted to all the neighbouring PEPs asking them to look into their respective local caches. When any PEP provides an Authorisation policy to requesting PEP, it is again checked in the dependency checkpoint. In either case, if the Authorisation policy is unavailable at the PEP level or the neighbouring PEPs level request is forwarded to PDP asking for the concerned Authorisation policy. As the number of insider requests increases, this architecture can be scaled up.

Figure 4 Shows each PEP receives requests from an insider, running with its associated PDP in parallel to all other PDPs. Each PEP performs the same tasks as Fig. 1, but in parallel to each other. When a PDP fails, its associated PEPs are distributed to remaining PDPs for fault-tolerant. Scalability is achieved because the proposed architecture can handle large requests. Multiple PEP-PDP blocks can be configured efficiently for better control

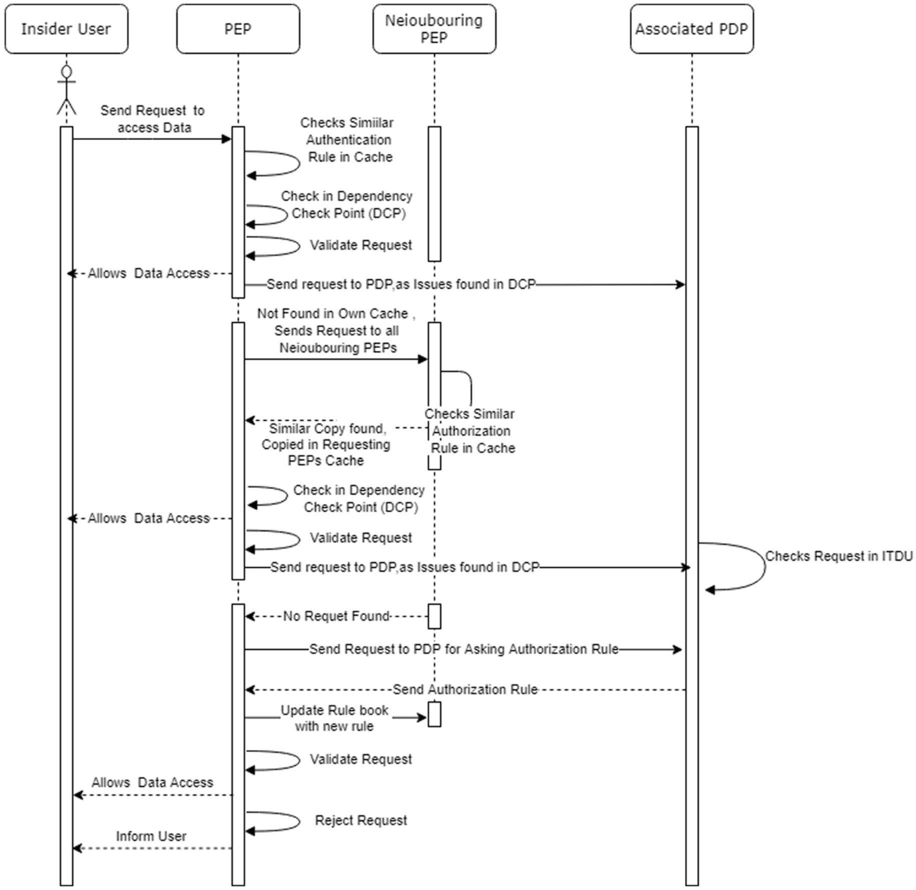


Fig. 2 Sequence diagram of multiple PEP – Single PDP architecture

over insider threats. Algorithm 1 defines the distributed PEP-PDP environment with side caching to prevent insider threats. We have assumed a dependency checkpoint, and the Internal threat unit is working well [36], 37]

In the proposed algorithm, an access request from an insider is received by PEP (any) if that PEP has any heuristic decision similar to the request. Dependency checkpoint checks for dependency between the current request and the previous request. Dependency checkpoint checks for violations of data access according to authorizations. If a similar heuristic decision request is unavailable, all its neighbouring PEPs are searched; if found, it is sent to a dependency checkpoint; otherwise, it is transferred to its associated PDP and associated PDP checks for possible threats at the dependency checkpoint and internal threat detection unit. If no threat is found, access to data is allowed, and the decision is communicated to all its associated PEPs; otherwise, the request gets rejected.

Table 2 Distributed architecture test bed configuration

Role	VMs	Number	Bandwidth
PDP	Octa Core, 8 GB RAM	4	100 Mbit/s
PEP	Octa Core, 8 GB RAM	24	100 Mbit/s

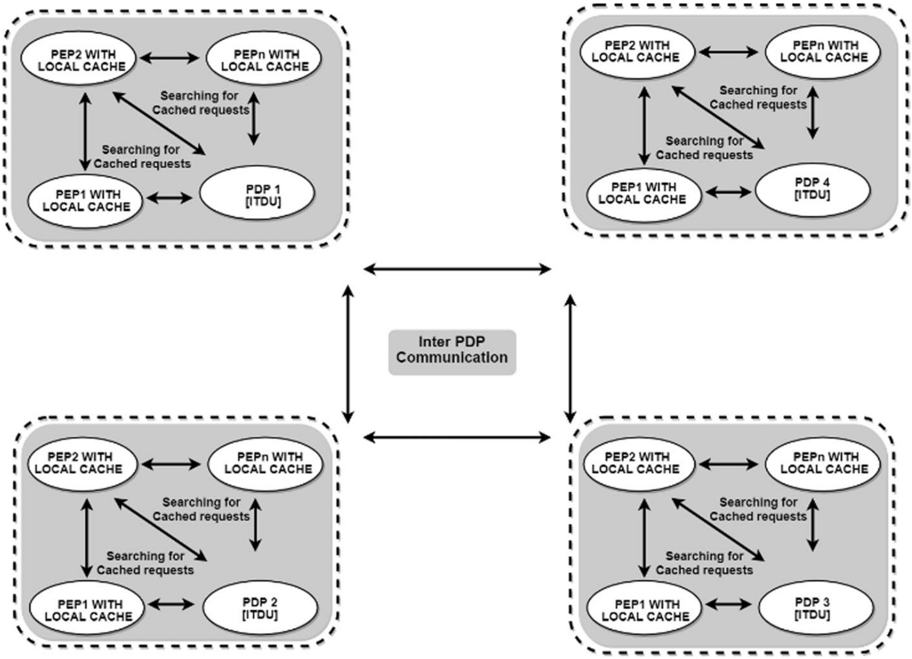


Fig. 3 .Distributed insider threat-aware PEP-side caching architecture

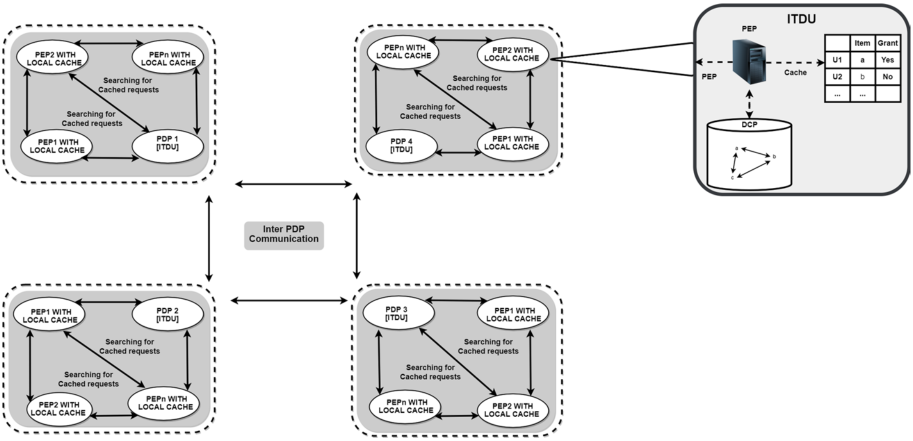


Fig. 4 Proposed distributed PEP- PDP architecture

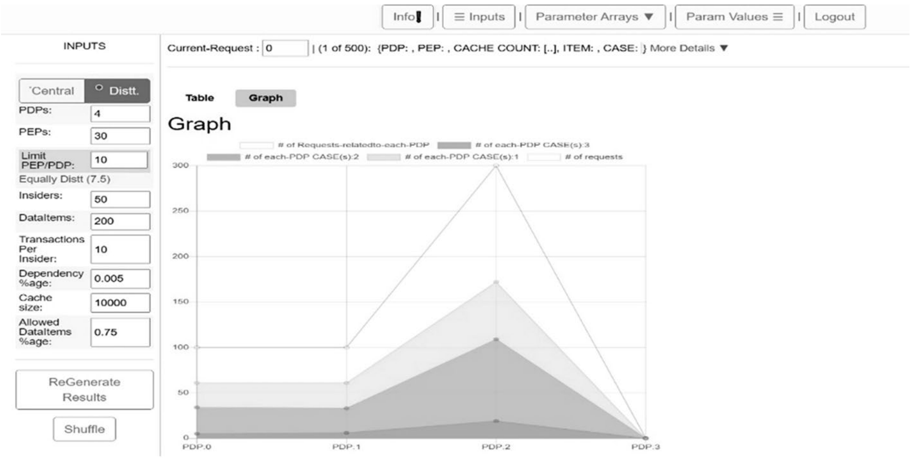
```
Algorithm 1:Distributed PEP-PDP environment with side caching for prevention of insider threat

Input: An insider alice access request Q for a data item D, received by PEP in one of the blocks.
Output: Access decision (Grant or Reject)
STEP 1: If Q (alice, D) does exist in PEP caches, then
            Send a request to DCP to check dependencies
               If D can be combined with K to infer information, then
                  If  Alice has a cached value of K, then
                     Forward alice request to the associated PDP to check the possible threat
                     Else
                     No threat found, re-issue the cache response for alice request to D
                   End If
               End If
           Else
            If Q (Alice, D) does  not exist in PEP caches, then
              Send Q (Alice, D) to all neighboring PEP respective cache,
              heuristic decision copy when found, send it to PEPs DCP for threat evaluation
            Else
             No decision copy, send Q (alice D) to the associated PDP
            End if
           End if
STEP 2: If alice request is received at the associated PDP, then
          Send a request to ITDU
            If ITDU decides that there is no threat exists, then
              Alice is allowed to get their requested D; all CPEPs receive associated PDP
              decision and corresponding CPEP allows the user to get his requested D.
            Else
              Alice is not allowed to get their requested D; the associated PDP rejects the
              request. The response gets updated in all CPEPs
            End If
        End If
```

4 Simulation and Results

Figure 5. illustrates the simulation runs of the proposed distributed PEP-PDP architecture. It consists of PEPs, PDPs, caches, and dependency checkpoints. Simulation runs can be varied based on various parameters (no. of PDPs, PEPs, insiders, data items, transactions, dependency percentage, cache size, and allowed data items percentage).

The proposed architecture consists of multiple blocks working parallel to each other, where each block consists of multiple PEP and a single PDP. As the number of blocks has increased improvement in handling the same number of requests with respect to time in



Current-Request : 0 | (1 of 500): {PDP: , PEP: , CACHE COUNT: [..], ITEM: , CASE: } More Details ▼

○ Central ● Distt.

PDPs: 4
PEPs: 30
Limit PEP/PDP: 10
Equally Distt (7.5)
Insiders: 50
DataItems: 200
Transactions Per Insider: 10
Dependency %age: 0.005
Cache size: 10000
Allowed DataItems %age: 0.75

Graph

Fig. 5 Simulation run

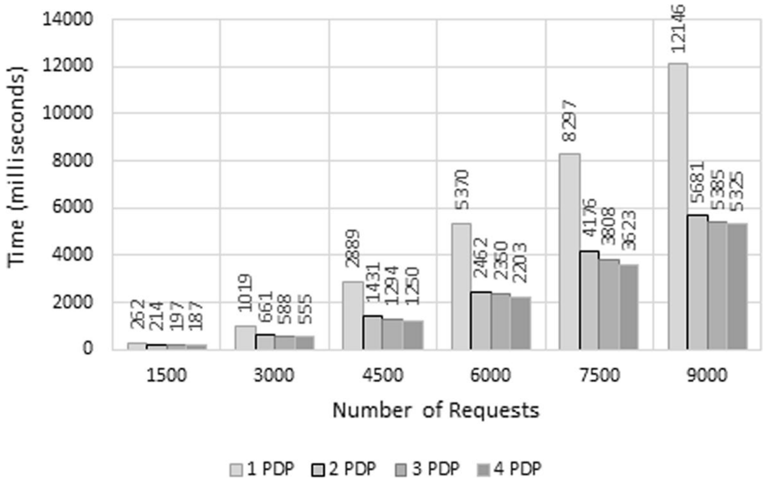


Fig. 6 Response time of existing architecture and proposed architecture

milliseconds has been seen, as shown in the figure, as compared to the architecture proposed by Yaseen et al. [35].

It can be observed from Fig. 6. and Table 3. that the proposed architecture takes half time to handle the same number of requests in contrast to [35].

The improvement in scalability has resulted in other parameters also, such as:-

4.1 Requests Handled W.R.T Data Dependency.

The requests generated by insiders is received at PEP for data request. Requested data may consist of dependencies, which violates authorization rules. This request is further

Table 3 Response time of existing 1 PDP architecture and proposed architecture

Number of requests	1 PDP	2 PDP	3 PDP	4 PDP	5 PDP	6 PDP
1500	262	214	197	187	179	168
3000	1019	661	588	555	524	505
4500	2889	1431	1294	1250	1196	1150
6000	5370	2462	2350	2203	2139	2046
7500	8297	4176	3808	3623	3502	3432
9000	12,146	5681	5385	5325	5239	5100

Table 4 Variable number of dependencies in various PDP architectures

Percentage of dependencies	5	10	15	20	25	30	35	40	45
Requests handled in 1PDP architecture	1454	1472	1474	1474	1474	1474	1474	1474	1474
Requests handled in 2 PDP architecture	2260	2309	2344	2374	2374	2374	2374	2374	2374
Requests handled in 3 PDP architecture	3816	3828	3841	3842	3845	3845	3845	3845	3845
Requests handled in 4 PDP architecture	5270	5277	5298	5309	5330	5330	5300	5330	5330

passed to PDP for further evaluation. The total number of requests generated by insiders was calculated by (number of insiders * transactions per insider).

Table 4. illustrates the variable number of dependencies (in percentage) in various PDP architectures in contrast to [35] which can work in coordination and symmetric. It is observed from Fig. 7 that 2 PDP architecture no. of requests got stable when several dependencies were 20% in contrast to 3 and 4 PDP architectures at 25%. The behaviour is analyzed and concludes to significant improvement which handles more requests with the help of a dependency checkpoint.

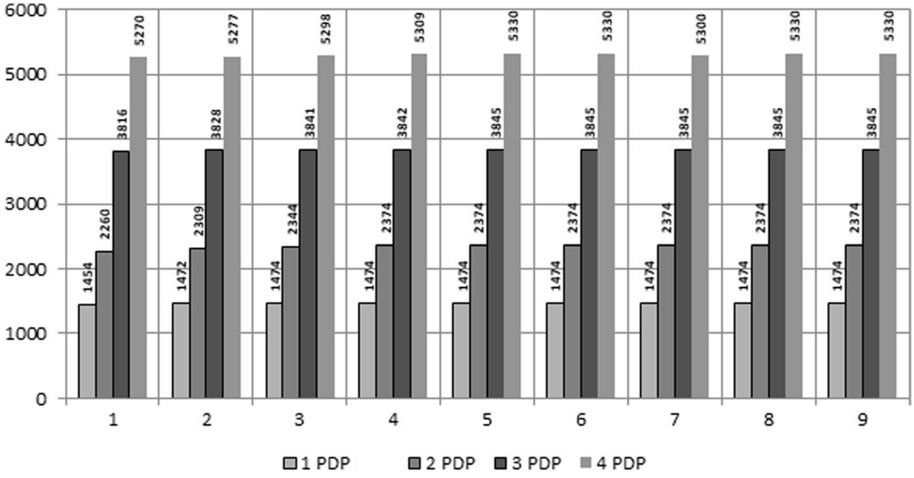


Fig. 7 Graph showing the variable number of dependencies in various PDP architectures

Table 5 Number of possible threats in various PDP architectures

Percentage of dependencies	1 PDP architecture	2 PDP architecture	3 PDP architecture	4 PDP architecture
5	56	62	66	78
10	77	67	70	83
15	78	75	72	89
20	77	80	77	89
25	77	85	80	89
30	79	85	87	89
35	79	85	87	92
40	79	84	87	90
45	78	84	87	90

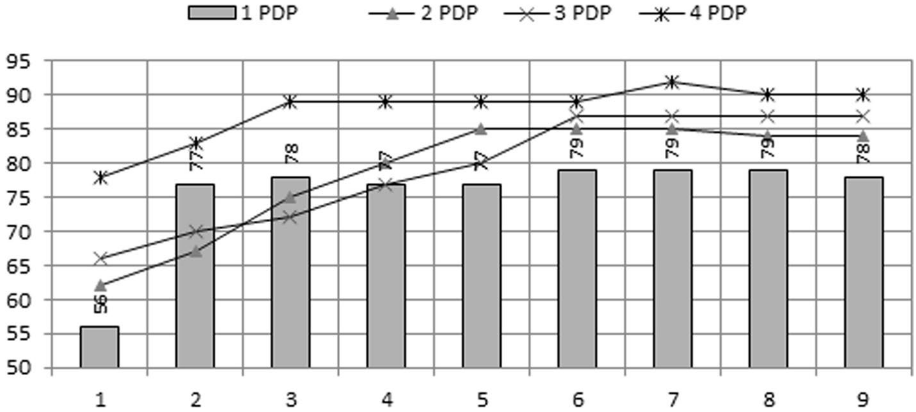


Fig. 8 Graph showing the number of possible threats in various PDP architectures

4.2 Possible threats detected w.r.t data dependency.

The number of possible threats was detected using a dependency checkpoint at PEP. It can be observed from the simulation data as in Table 5 and Fig. 8. As the percentage of dependency increases (in percentage), the number of possible threats increases. The result shows that total requests generated in the simulation architecture are evenly distributed among each sector of multiple PEPs and their associated PDP.

The number of requests generated is much higher than the centralized PDP architecture, but the threat number is not rising. The number of possible threats becomes stable when the percentage of dependencies reaches the 25–30. It means the percentage of dependencies plays a significant role in finding possible threats.

4.3 Requests Passed to PDP W.R.T Requests Per Insider

Simulation results in Table 6. and Fig. 9. shows that as the number of requests per insider increases, requests to PDP also increase to find out possible threats. The results confirm that our proposed architecture can handle more requests than the architecture proposed in

Table 6 Details of requests passed to PDP in various PDP architectures

Requests/Insider	1 PDP	2 PDP	3 PDP	4 PDP
10	505	785	1291	1776
30	1478	2297	3813	5302
50	2450	3877	6368	8863
70	3370	5481	9006	12,405
90	4270	6999	11,425	15,867
110	5190	8606	13,921	19,452

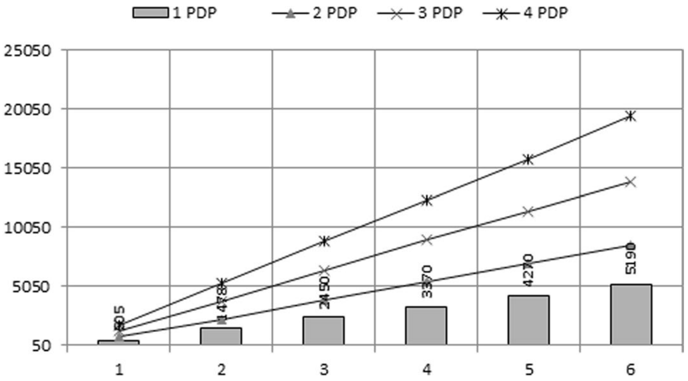


Fig. 9 Graph of requests passed to PDP in various PDP architectures

[35]. As the number of requests per insider increases at the level of 110, the number of requests reaching PDP in the 2 PDP architecture almost reached double that of the architecture proposed in [35]; requests reach almost three times in the 3 PDP architecture and almost four times in 4 PDP architecture. It means that our proposed architecture is much more efficient than the architecture proposed in [35]

4.4 Requests passed to PDP w.r.t number of data items available

As the number of data items increases, the percentage of requests towards PDP increases, as shown in Fig. 10.

It can be seen clearly from the simulation results in Table 7 and Fig. 10 that the number of requests handled in the architecture proposed in [35] is much less than our proposed architecture; that is why it increases at other values of the number of data items. The total number of requests generated by insiders was calculated by (number of insiders * transactions per insider).

4.5 First hit versus collaborative hits

Collaborative means the collection of all the hits of all three cases. There are three cases where the decision to make a request takes place. The first case is when the corresponding PEP receives the request, and the corresponding PEP has a decision copy similar to the

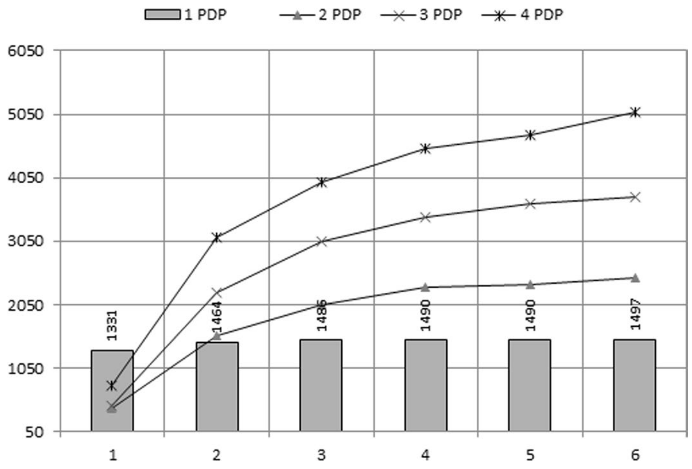


Fig. 10 Graph of percentage requests passed to PDP in various PDP architectures

Table 7 Details of percentage requests passed to PDP in various PDP architectures

Number of data items	1PDP	2 PDP	3 PDP	4 PDP
50	1331	424	450	746
150	1464	1555	2233	3101
250	1486	2054	3054	3982
350	1490	2319	3426	4500
450	1490	2364	3634	4713
550	1497	2464	3749	5084

Table 8 Details of the first hit versus collaborative hits of various PDP architectures

X-axis	2	10	20	30	40
first hit 1 PDP	53	13	7	5	4
collaborative hit 1 PDP	53	95	104	104	104
first hit 2 PDP	71	67	65	63	60
collaborative hit 2 PDP	71	374	663	946	1188
first hit 3 PDP	83	78	76	74	73
collaborative hit 3 PDP	83	384	480	690	841
first hit 4 PDP	87	80	77	72	65
collaborative hit 4 PDP	87	351	364	545	662

request with it, also known as the first hit. In this case, there is no need to go anywhere to access the decision copy. Second, the corresponding PEP does not have the decision copy; it asks all its neighbouring PEPs; if available, the decision copy is available to the corresponding PEP. Finally, in the last third case, the decision copy is not available at the corresponding PEP, nor at its neighbouring PEPs; the request gets transferred to its associated PDP where a decision to this is taken, nor a copy of the decision transferred to the corresponding PEP where the request is first received.

It can be viewed from the simulation results in Table 8 and Figs. 11 and 12 that the results of the first hit and collaborative hits are much better than the architecture proposed in [35] as compared to our proposed architecture. In our architecture number of PEPs, PDPs are more in numbers, which leads to better results.

4.6 Static Test with Analysis of Variance Test (ANOVA)

Various static methods are available, like the Z test and T-test, but they are applicable only for two group values. The Chi-square test finds the expected value between three

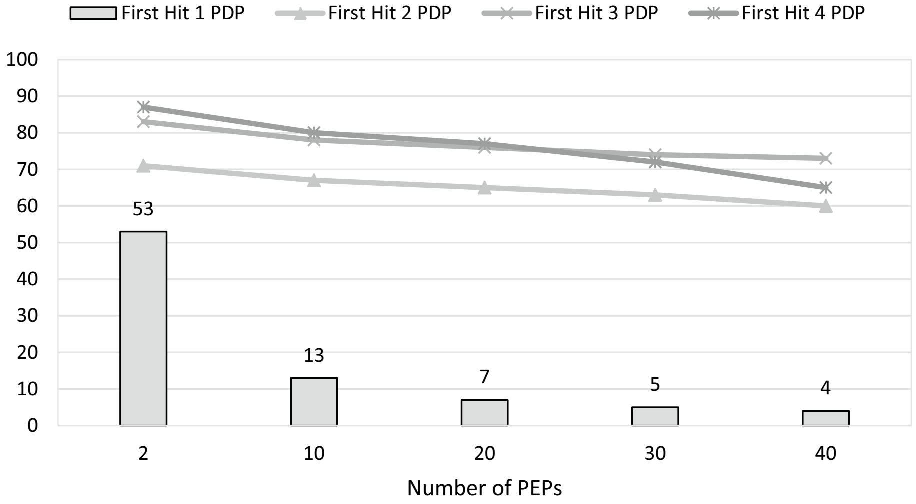


Fig. 11 Graph of the first hit various PDP architectures

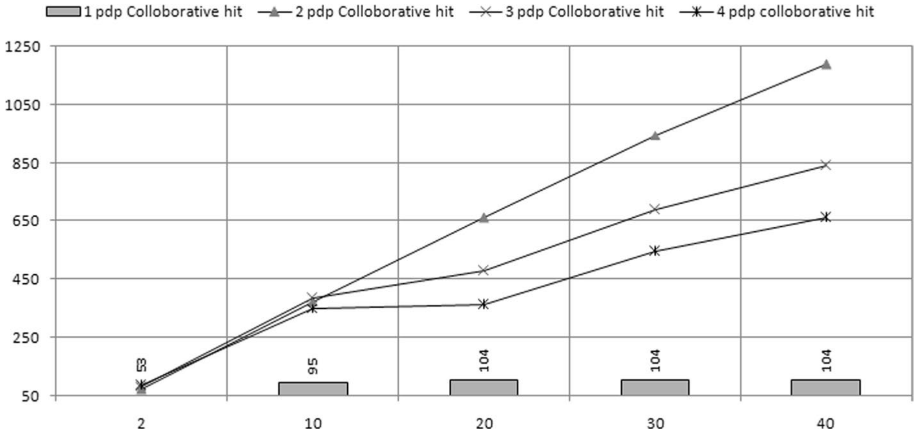


Fig. 12 Graph of collaborative hits of various PDP architectures

Table 9 Assumptions of Null and Alternative Hypothesis for ANOVA test

	4.1 Requests handled at a variable percentage number of dependencies	4.2 Detection of a number of possible threats at a variable percentage number of dependencies	4.3 Increase in requests Per insider to requests passed to PDP	4.4 Percentage requests passed to PDP with an increase in data items	4.5 First hit versus collaborative hits	
Null hypothesis	For the Same percentage of dependencies existing and proposed architecture behave the same	For the same percentage of dependencies existing and proposed architecture behave the same	For the same set of requests/insiders, existing and proposed architecture behave the same	For the same set of the number of data items for the existing and proposed architecture behave the same	For the same set of the first hit for a number of PEPs for the existing and proposed architecture behave the same	For the same set of the collaborative hit for a number of PEPs for the existing and proposed architecture behave the same
Alternate hypothesis	For the Same percentage of dependencies existing and proposed architecture are behaving differently because with the increase in number of PDPs Performance is improving	For the same percentage of dependencies existing and proposed architecture are behaving differently because with the increase in number of PDPs Performance is improving	For the same set of requests/insider existing and proposed architecture are behaving differently because with the increase in number of PDPs Performance is improving	For the same set of the number of data items for the existing and proposed architecture are behaving differently because with the increase in number of PDPs Performance is improving	For the same set of the first hit for number of PEPs for the existing and proposed architecture are behaving differently because with the increase in number of PDPs Performance is improving	For the same set of the collaborative hit for number of PEPs for the existing and proposed architecture are behaving differently because with the increase in number of PDPs Performance is improving

Table 10 ANOVA test results for requests handled at variable percentage number of dependencies

Groups	Count	Sum	Average	Variance
Row 1	9	13,244	1471.555556	43.77777778
Row 2	9	21,157	2350.777778	1658.444444
Row 3	9	34,552	3839.111111	105.3611111
Row 4	9	47,774	5308.222222	563.1944444
ANOVA				

Source of variation	SS	df	MS	F	*P*-value	F crit
Between Groups	76,991,092.53	3	25,663,697.51	43,300.04735	8.46E-58	2.901119584
Within Groups	18,966.22222	32	592.6944444			
Total	77,010,058.75	35				

or more groups. These tests are not valid here as we are interested in analyzing variance between the group values from the paper [35] and multiple group values from our proposed work. Therefore, the analysis of variance test (ANOVA) is used to determine whether there is any statistical difference between the means of three or more independent groups [53–55]. We have chosen null and alternate hypotheses for different proposed architecture variants for each measured point, as shown in Table 9.ANOVA tests were conducted at a significance level of 5%.

Result for Anova tests is shown below in Tables 10, 11, 12, 13, 14 and 15.

It can be observed from the Table 10. that the value of F-ratio is more than the value of F crit, which interprets that the Alternate hypothesis has been Accepted, which states that with the increase in the number of PDPs, the performance of the system is improving for the same set of percentage of Dependency.

It can be observed from the Table 11. that the value of F-ratio is more than the value of F crit, which interprets that the Alternate hypothesis has been Accepted, which states that with the increase in the number of PDPs, the performance of the system is improving for the same set of percentage of Dependency.

It can be observed from the Table 12. that the value of F-ratio is more than the value of F crit, which interprets that the Alternate hypothesis has been Accepted, which states that with the increase in the number of PDPs, the performance of the system is improving for the same set of Requests/Insider.

It can be observed from the Table 13. that the value of F-ratio is more than the value of F crit, which interprets that the Alternate hypothesis has been Accepted, which states that with the increase in the number of PDPs, the performance of the system is improving for the same set of Number of Data Items.

It can be observed from the Table 14. that the value of F-ratio is more than the value of F crit, which interprets that the Alternate hypothesis has been Accepted, which states that with the increase in the number of PDPs, the performance of the system is improving for the same set of the First hit for Number of PEPs.

It can be observed from the Table 15. that the value of F-ratio is more than the value of F crit, which interprets that the Alternate hypothesis has been Accepted, which states that with the increase in the number of PDPs, the performance of the system is improving for the same set of the collaborative hit for Number of PEPs.

We are able to find only two papers in which authors have tried to use multiple PDPs to make the system better than the single PDPs, as shown in Table 16.

In paper [56], the authors have proposed a two-stage clustering approach with PDPs working sequentially on authorization policies. In this approach, when a request is received and transferred by the request dispatcher, it is further sequentially received by multiple sub-PDPs, at each sub-PDP request is matched with the authorization policy. In paper [32], the authors have proposed a 4PDP4E toolset to protect users data travelling online. This toolset was proposed for data protection directives given by the European Union. This toolset,4 PDPs were used for risk management, requirement engineering, model-driven design, and system assurance in the systems development lifecycle. In this, different PDP architectures work for different requirements in coordination.

In the proposed architecture, we have used 4 PDPs in simulation, scaled up to the maximum level of 8 PDPs. The proposed architecture is tested statistically using the

Table 11 ANOVA test results for detection of number of possible threats at variable percentage number of dependencies

Groups	Count	Sum	Average	Variance		
Column 1	9	680	75.55555556	54.52777778		
Column 2	9	707	78.55555556	75.77777778		
Column 3	9	713	79.22222222	69.94444444		
Column 4	9	789	87.66666667	19		
ANOVA						
Source of variation	SS	df	MS	F	*P*-value	F crit
Between Groups	728.75	3	242.9166667	4.43177499	0.010278	2.90112
Within Groups	1754	32	54.8125			
Total	2482.75	35				

Table 12 ANOVA test results for Increase in Requests per Insider to Requests passed to PDP

Groups	Count	Sum	Average	Variance		
Column 1	6	17,263	2877.166667	3,059,876.167		
Column 2	6	28,045	4674.166667	8,585,431.367		
Column 3	6	45,824	7637.333333	22,443,362.67		
Column 4	6	63,665	10,610.83333	43,661,092.57		
ANOVA						
Source of variation	SS	df	MS	F	*P*-value	F crit
Between Groups	207,846,098.8	3	69,282,032.93	3.564359837	0.03253569	3.098391212
Within Groups	388,748,813.8	20	19,437,440.69			
Total	596,594,912.6	23				

Table 13 ANOVA test results for Percentage Requests Passed to PDP with Increase in Data Items

Groups	Count	Sum	Average	Variance		
Column 1	6	8758	1459.666667	4100.266667		
Column 2	6	11,180	1863.333333	604,440.6667		
Column 3	6	16,546	2757.666667	1,577,157.067		
Column 4	6	22,126	3687.666667	2,549,030.667		
ANOVA						
Source of variation	SS	df	MS	F	*P*-value	F crit
Between Groups	17,706,988.5	3	5,902,329.5	4.986414146	0.00961568	3.098391212
Within Groups	23,673,643.33	20	1,183,682.167			
Total	41,380,631.83	23				

Table 14 ANOVA test results for the first hit

Groups	Count	Sum	Average	Variance		
Row 1	5	460	92	490.5		
Row 2	5	3242	648.4	197,158.3		
Row 3	5	2478	495.6	85,007.3		
Row 4	5	2009	401.8	47,829.7		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	826,759.75	3	275,586.5833	3.335533125	0.046029131	3.238871517
Within Groups	1,321,943.2	16	82,621.45			
Total	2,148,702.95	19				

Table 15 ANOVA test results for collaborative hits

Groups	Count	Sum	Average	Variance		
Row 1	5	460	92	490.5		
Row 2	5	3242	648.4	197,158.3		
Row 3	5	2478	495.6	85,007.3		
Row 4	5	2009	401.8	47,829.7		
ANOVA						
Source of Variation	SS	df	MS	F	*P*-value	F crit
Between Groups	826,759.75	3	275,586.5833	3.335533125	0.046029131	3.238871517
Within Groups	1,321,943.2	16	82,621.45			
Total	2,148,702.95	19				

Table 16 Comparison with existing work where multiple PDPs used

	The clustering-based request travels from one PDP to another	Different PDP architectures work for different requirements in coordination with each other	Working independently to each other, although connected also, can be scaled up or down easily for the same purpose for better control
Fan Deng et al. [56]	**X**		
Carvalho et al. [32]		**X**	
Proposed Architecture			**X**

ANOVA test, and the value of p is less than 0.05, which says existing architectures and proposed architectures are working the same. In this proposed architecture, even though they work in parallel, they are also connected. The proposed architecture is best suitable for access management in DBaaS, where requests migrate to different clouds. Each PDP architecture works on different clouds but in coordination with each other.

5 Conclusion

Cloud computing for data storage and retrieval has become a basic necessity for every individual due to its significant cost-saving, security, flexibility, mobility, insight, increased collaboration, quality control, and other numerous benefits [57]. Many cloud-based companies are essential in managing access rights to cloud relational databases [58]. On the other side, the problem of insider threat has become a significant threat to data security. Literature discusses various request-response paradigms based on the PEP-PDP architecture. Yaseen et al. [35] have suggested one of the combinations in which multiple PEPs were used with a single PDP to handle several Requests, but the major limitation of this architecture is it is not able to handle a large number of requests from the insiders, and single PDP becomes a stressed member of the system. Failure of PDP leads to the failure of the whole system. Scalability can achieve up to some level.

To improve the PEP-PDP architecture suggested by Yaseen et al. [35], this paper proposed a distributed PEP-PDP architecture for insider threat-aware access control for cloud relational databases. In the proposed architecture problem of scalability, PDP failure is resolved by introducing multiple PDPs and a set of associated PEPs. All PDPs work in parallel, although connected. During simulation and results analysis, it is found that the proposed architecture is working satisfactorily. The proposed architecture can handle many more requests than existing ones and can work in parallel in multiple clouds. The number of pending requests can distribute among other remaining PDPs. Scalability can be achieved in the proposed architecture as per requirement. In future, the proposed architecture can be extended to include Load balancing to counter the failure of any PDP.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability All data generated or analysed during this study are included in this published article.

Code Availability The code developed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Modi, C., Patel, D., Borisaniya, B., Patel, A., & Rajarajan, M. (2013). A survey on security issues and solutions at different layers of Cloud computing. *The Journal of Supercomputing, 63*(2), 561–592.
2. Hsu, C. H., Ma, J., & Obaidat, M. S. (2014). Dynamic intelligence towards merging cloud and communication services. *Information Systems Frontiers, 16*(1), 1–5.
3. Flahive, A., Taniar, D., & Rahayu, W. (2013). Ontology as a Service (OaaS): A case for sub-ontology merging on the cloud. *The Journal of Supercomputing, 65*(1), 185–216.
4. Petrov, C. (2021). *25+ Impressive big data statistics for 2021*. https://techjury.net/blog/big-data- statistics/#gref
5. Petters, J. E. F. F. (2020, September 28). *Data privacy guide: definitions, explanations and legislation*. https://Www.Varonis.Com/Blog/Data-Privacy/.
6. Al-Gburi, A., Al-Hasnawi, A., & Lilien, L. (2018). Differentiating security from privacy in internet of things: a survey of selected threats and controls. In *Computer and network security essentials*, (pp. 153–172). Springer, Cham.
7. Sharma, S., Singh, S., Singh, A., & Kaur, R. (2016). Virtualization in cloud computing. *International Journal of Scientific Research in Science, Engineering and Technology, 2*, 181–186.
8. Devi, V. A. & Nayyar, A. (2021). Evaluation of geotagging twitter data using sentiment analysis during COVID-19. In *Proceedings of the second international conference on information management and machine intelligence*, (pp. 601–608). Springer, Singapore.
9. Daradoumis, T., Bassi, R., Xhafa, F., & Caballé, S. (2013). A review on massive e-learning (MOOC) design, delivery and assessment. In *2013 eighth international conference on P2P, parallel, grid, cloud and internet computing* (pp. 208–213). IEEE.
10. Mehmi, S., Verma, H. K., & Sangal, A. L. (2014). Smart grid cloud for Indian power sector. In *International conference on recent advances and innovations in engineering (ICRAIE-2014)* (pp. 1–6). IEEE.
11. Bhatia, R. (2020). Interoperability solutions for blockchain. In *2020 International conference on smart technologies in computing, electrical and electronics (ICSTCEE)* (pp. 381–385). IEEE.
12. Singh, M. & Singh, S. (2014). Review of implicit security mechanisms for cloud computing. International *Journal of Computer Applications*, *106*(17).
13. Verma, A. & Kaushal, S. (2011). Cloud computing security issues and challenges: a survey. In *International conference on advances in computing and communications* (pp. 445–454). Springer, Berlin, Heidelberg.
14. Shikare, D. & Shetty, S. D. (2014). Supercloud–need, issues and challenges. *International Journal of Engineering Research*, *3*(5).
15. Sharma, R., Kumar, S., & Trivedi, M. C. (2013). Mobile cloud computing: Bridging the gap between cloud and mobile devices. In *2013 5th international conference and computational intelligence and communication networks* (pp. 553–555). IEEE.
16. Mackenzie, R. J. (2021). *4 ways that the cloud is changing research*. https://Www.Technologynetworks.Com/Informatics/Lists/4-Ways-That-the-Cloud-Is-Changing-Research-315952.
17. Gurucul Solutions Pvt Ltd. (n.d.). 2020 insider threat survey report. https://Gurucul.Com/2020-Insider-Threat-Survey-Report. Retrieved June 5, 2021, from https://gurucul.com/2020-insider-threat-survey-report.
18. Tabrizchi, H., & Rafsanjani, M. K. (2020). A survey on security challenges in cloud computing: Issues, threats, and solutions. *The Journal of Supercomputing, 76*(12), 9493–9532.
19. Liu, L., De Vel, O., Han, Q. L., Zhang, J., & Xiang, Y. (2018). Detecting and preventing cyber insider threats: A survey. *IEEE Communications Surveys & Tutorials, 20*(2), 1397–1417.
20. Ramachandran, M., & Chang, V. (2016). Towards performance evaluation of cloud service providers for cloud data security. *International Journal of Information Management, 36*(4), 618–625.
21. Barrowclough, J. P., & Asif, R. (2018). Securing cloud hypervisors: A survey of the threats, vulnerabilities, and countermeasures. *Security and Communication Networks, 2018*, 1–20.
22. Sandhu, R. S., & Samarati, P. (1994). Access control: Principle and practice. *IEEE Communications Magazine, 32*(9), 40–48.
23. Kizza, J., & Kizza, F. M. (2008). *Access control, authentication, and authorization* (pp. 180–208). IGI Global.
24. Sail, S. & Bouden, H. (2018). A multi-factor authentication scheme to strength data-storage access. In International conference on big data, cloud and applications (pp. 67–77). Springer, Cham.
25. Anakath, A. S., Rajakumar, S., & Ambika, S. (2019). Privacy preserving multi factor authentication using trust management. *Cluster Computing, 22*(5), 10817–10823.

26. Chen, H. C. J., Violetta, M. A., & Yang, C. Y. (2013). Contract RBAC in cloud computing. *The Journal of Supercomputing, 66*(2), 1111–1131.
27. Xu, J., Yu, Y., Meng, Q., Wu, Q., & Zhou, F. (2020). Role-based access control model for cloud storage using identity-based cryptosystem. *Mobile Networks and Applications*, 1–18.
28. Morisset, C., Willemse, T. A., & Zannone, N. (2019). A framework for the extended evaluation of ABAC policies. *Cybersecurity, 2*(1), 1–21.
29. Rana, S., & Mishra, D. (2020). Efficient and secure attribute based access control architecture for smart healthcare. *Journal of Medical Systems, 44*(5), 1–11.
30. Ahmadi, S., Nassiri, M., & Rezvani, M. (2020). XACBench: A XACML policy benchmark. *Soft Computing, 24*(21), 16081–16096.
31. Rezvani, M., Rajaratnam, D., Ignjatovic, A., Pagnucco, M., & Jha, S. (2019). Analyzing XACML policies using answer set programming. *International Journal of Information Security, 18*(4), 465–479.
32. de Carvalho, R. M., Del Prete, C., Martin, Y. S., Araujo Rivero, R. M., Önen, M., Schiavo, F. P., Rumín, Á. C., Mouratidis, H., Yelmo, J. C., & Koukovini, M. N. (2020). Protecting citizens' personal data and privacy: joint effort from GDPR EU cluster research projects. *SN Computer Science, 1*(4), 1–16.
33. Bertin, E., Hussein, D., Sengul, C., & Frey, V. (2019). Access control in the Internet of Things: A survey of existing approaches and open research questions. *Annals of Telecommunications, 74*(7), 375–388.
34. Bruno, E., Gallier, R., & Gabillon, A. (2019). Enforcing access controls in IoT networks. In *International conference on future data and security engineering* (pp. 429–445). Springer, Cham.
35. Yaseen, Q., Jararweh, Y., Panda, B., & Althebyan, Q. (2017). An insider threat aware access control for cloud relational databases. *Cluster Computing, 20*(3), 2669–2685.
36. Yaseen, Q. & Panda, B. (2010). Predicting and preventing insider threat in relational database systems. In *IFIP international workshop on information security theory and practices* (pp. 368–383). Springer, Berlin, Heidelberg.
37. Yaseen, Q., & Panda, B. (2012). Insider threat mitigation: Preventing unauthorized knowledge acquisition. *International Journal of Information Security, 11*(4), 269–280.
38. Lazouski, A., Martinelli, F., Mori, P., & Saracino, A. (2017). Stateful data usage control for android mobile devices. *International Journal of Information Security, 16*(4), 345–369.
39. Da Silva, C. E., Diniz, T., Cacho, N., & de Lemos, R. (2018). Self-adaptive authorization in OpenStack cloud platform. *Journal of Internet Services and Applications, 9*(1), 1–17.
40. Elgedawy, I., Khurshid, S., Masood, R., & Shibli, M. A. (2018). CRESCENT+: A self-protecting framework for reliable composite web service delivery. *Iran Journal of Computer Science, 1*(2), 65–87.
41. Nyrkov, A., Romanova, Y., Ianiushkin, K., & Li, I. (2018). Data processing model in hierarchical multi-agent system based on decentralized attribute-based encryption. In *Energy management of municipal transportation facilities and transport* (pp. 429–438). Springer, Cham.
42. Son, H. X., Nguyen, M. H., & Vo, H. K. (2019). Toward an privacy protection based on access control model in hybrid cloud for healthcare systems. In *International joint conference: 12th international conference on computational intelligence in security for information systems (CISIS 2019) and 10th international conference on European transnational education (ICEUTE 2019)* (pp. 77–86). Springer, Cham.
43. Cao, Y., Huang, Z., Yu, Y., Ke, C., & Wang, Z. (2020). A topology and risk-aware access control framework for cyber-physical space. *Frontiers of Computer Science, 14*(4), 1–16.
44. Ryan, D., De Leon, M. P., Grant, N., Butler, B., Vogel, S., Mirz, M., & Lyons, P. (2019). Deriving policies from connection codes to ensure ongoing voltage stability. *Energy Informatics, 2*(1), 1–14.
45. Elmisery, A. M., Rho, S., & Aborizka, M. (2019). A new computing environment for collective privacy protection from constrained healthcare devices to IoT cloud services. *Cluster Computing, 22*(1), 1611–1638.
46. Oglaza, A., Laborde, R., Zaraté, P., Benzekri, A., & Barrère, F. (2017). A new approach for managing Android permissions: Learning users' preferences. *EURASIP Journal on Information Security, 2017*(1), 1–16.
47. Krempel, E., Birnstill, P., & Beyerer, J. (2017). A Privacy-Aware Fall Detection System for Hospitals and Nursing Facilities. *European Journal for Security Research, 2*(2), 83–95.
48. Amin, S. O., Siddiqui, M. S., & Hong, C. S. (2008). A novel IPv6 traceback architecture using COPS protocol. *Annals of Telecommunications-Annales des Télécommunications, 63*(3), 207–221.
49. Liu, Y., Zhang, J., & Zhan, J. (2021). Privacy protection for fog computing and the internet of things data based on blockchain. *Cluster Computing, 24*(2), 1331–1345.

50. Vassis, D., Belsis, P., Skourlas, C., & Pantziou, G. (2010). Providing advanced remote medical treatment services through pervasive environments. *Personal and Ubiquitous Computing, 14*(6), 563–573.
51. Basile, C., Cappadonia, A., & Lioy, A. (2011). Network-level access control policy analysis and transformation. *IEEE/ACM Transactions On Networking, 20*(4), 985–998.
52. Gogoulos, F. I., Antonakopoulou, A., Lioudakis, G. V., Mousas, A. S., Kaklamani, D. I., & Venieris, I. S. (2014). On the design of a privacy aware authorization engine for collaborative environments. *Electronic Markets, 24*(2), 101–112.
53. Cuevas, A., Febrero, M., & Fraiman, R. (2004). An anova test for functional data. *Computational Statistics & Data Analysis, 47*(1), 111–122.
54. Kim, T. K. (2017). Understanding one-way ANOVA using conceptual figures. *Korean Journal of Anesthesiology, 70*(1), 22–26.
55. Górecki, T., & Smaga, Ł. (2015). A comparison of tests for the one-way ANOVA problem for functional data. *Computational Statistics, 30*(4), 987–1010.
56. Deng, F., Lu, J., Wang, S. Y., Pan, J., & Zhang, L. Y. (2019). A distributed PDP model based on spectral clustering for improving evaluation performance. *World Wide Web, 22*(4), 1555–1576.
57. Salesforce.com. (n.d.). 12 Benefits of Cloud Computing. https://Www.Salesforce.Com/Products/Platform/Best-Practices/Benefits-of-Cloud-Computing/. Retrieved June 5, 2021, from https://www.salesforce.com/products/platform/best-practices/benefits-of-cloud-computing/
58. Indu, I., Anand, P. R., & Bhaskar, V. (2018). Identity and access management in cloud environment: Mechanisms and challenges. *Engineering Science and Technology, an International Journal, 21*(4), 574–588.

Gaurav Deep received his B.Tech.Degree in Computer Science and Engineering from JMIT, Radour, Yamuna Nagar, India in 2003, M.Tech. Degree in Computer Science and Engineering from Punjab Technical University, Jalandhar, India, in 2009, and pursuing a PhD in Computer Science and Engineering from the Jaypee University of Information Technology (JUIT), Waknaghat, India. His research area of interest includes Data security from Insider Threats in Distributed Systems. His work includes Authentication Mechanisms, including Blockchain. He has already implemented various techniques of Data Security hiding using Steganography in MP3, Barcodes, Video Files, X Ray, MRI and Ultrasound.

Jagpreet Sidhu received his B.Tech degree in Computer Science and Engineering from Punjab Technical University, Jalandhar, Punjab, India, in 2004 and M.E. degree in Information Technology from Panjab University, Chandigarh, India, in 2010. He completed a PhD in Computer Science and Engineering from Panjab University, Chandigarh, India in 2016, working on a trust problem in cloud computing. He has over 10 years of experience in Academia, working as an Assistant Professor (SG) with the Department of CSE, JUIT. He has published many papers in international journals and conferences. His research interests include cloud computing and trust issues in distributed systems.

Rajni Mohana received the B.Tech. Degree in computer science and engineering from BPUT, Rourkela, India, in 2003, the M.Tech. Degree in computer science and engineering from Punjab Technical University, Jalandhar, India, in 2009, and a PhD degree in Computer Science and Engineering from the Jaypee University of Information Technology (JUIT), Waknaghat, India, in 2013. She has over 19 years of experience in Academia. She is currently an Associate Professor with the Department of CSE, JUIT. She also works on interoperability problems in service-oriented architecture computing. She has published her research in various impact factor journals, such as Senors (MDPI) and Proceedings of the National Academy of Sciences: Physical Sciences. She has published around 20 articles in various ESCI and Scopus journals. She has guided four master's students and two PhD students. She is currently guiding two more students. Her research interest includes sentiment analysis cloud computing. She served as the TPC Chair for the 2015 International Conference on Image Information Processing, JUIT, in December 2015 and the TPC Co-Chair for the 2017 International Conference on Image Information Processing, in December 2017. She serves as a reviewer for various renowned international journals and international conferences.