Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.062902

ARTICLE

BIG-ABAC: Leveraging Big Data for Adaptive, Scalable, and Context-Aware Access Control

Sondes Baccouri[1,2,*,#] **and Takoua Abdellatif** [3,#]

[1]SERCOM Lab, Tunisia Polytechnic School, Bizerte, 7000, Tunisia
[2]Science and Technology for Defense Lab (STD), Military Research Center, Taeib Mhiri, Aouina, Tunis, 2045, Tunisia
[3]SERCOM Lab, Tunisia Polytechnic School, Sousse, 4089, Tunisia

*Corresponding Author: Sondes Baccouri. Email: baccouri.sondes@ept.u-carthage.tn

[#]These authors contributed equally to this work

Received: 30 December 2024; Accepted: 28 February 2025; Published: 11 April 2025

ABSTRACT: Managing sensitive data in dynamic and high-stakes environments, such as healthcare, requires access control frameworks that offer real-time adaptability, scalability, and regulatory compliance. **BIG-ABAC** introduces a transformative approach to Attribute-Based Access Control (ABAC) by integrating real-time policy evaluation and contextual adaptation. Unlike traditional ABAC systems that rely on static policies, BIG-ABAC dynamically updates policies in response to evolving rules and real-time contextual attributes, ensuring precise and efficient access control. Leveraging decision trees evaluated in real-time, BIG-ABAC overcomes the limitations of conventional access control models, enabling seamless adaptation to complex, high-demand scenarios. The framework adheres to the **NIST ABAC standard** while incorporating modern distributed streaming technologies to enhance **scalability and traceability**. Its flexible policy enforcement mechanisms facilitate the implementation of regulatory requirements such as HIPAA and GDPR, allowing organizations to align access control policies with compliance needs dynamically. Performance evaluations demonstrate that BIG-ABAC processes **95%** of access requests within **50 ms** and updates policies dynamically with a latency of **30 ms**, significantly outperforming traditional ABAC models. These results establish BIG-ABAC as a benchmark for **adaptive, scalable, and context-aware** access control, making it an ideal solution for dynamic, high-risk domains such as healthcare, smart cities, and Industrial IoT (IIoT).

KEYWORDS: ABAC; big data; context-aware; decision trees; adaptive policy; scalability

1 Introduction

In today's data-driven world, ensuring secure and efficient access to sensitive information is paramount, particularly in dynamic and high-demand environments such as healthcare, emergency response, industrial operations, and critical infrastructure. These sectors require adaptive and real-time access control mechanisms to balance privacy, security, and operational efficiency. For example, in healthcare, regulatory compliance frameworks such as HIPAA must be reconciled with the need for immediate access to critical medical information during emergencies. Thus, access control frameworks must not only enforce stringent security measures but also dynamically adapt to evolving contextual and operational requirements to maintain compliance and usability.

Traditional Attribute-Based Access Control (ABAC) solutions, while conceptually robust, often struggle with scalability, real-time adaptability, and compliance challenges, particularly in IoT environments, where

devices and data streams evolve continuously. As highlighted in [1], effective access control in IoT requires mechanisms that handle heterogeneous devices, decentralized decision-making, and dynamic policy updates. Existing ABAC models typically rely on static policy structures, such as predefined rules or offline-evaluated decision trees, which limit their applicability in dynamic environments where attributes, contextual data, and policies frequently change. This lack of real-time adaptability results in performance bottlenecks, limited scalability, and reduced responsiveness, especially in high-demand operational scenarios.

Although some implementations attempt online policy evaluation, many fail to fully integrate essential components outlined in the NIST ABAC standard [2], such as the Policy Information Point (PIP), Context Handler, Policy Decision Point (PDP), Policy Enforcement Point (PEP), and Policy Administration Point (PAP). The partial implementation of these components often leads to fragmented architectures that are incapable of efficiently processing or enforcing policies in real-time, thus affecting scalability and traceability [3–6]. While several studies [7–11] have attempted to incorporate elements like Context Handlers and PDPs, they often overlook critical aspects such as dynamic PAP or PEP, which are essential for real-time policy updates and enforcement. These limitations underscore the need for a comprehensive solution that ensures full NIST ABAC compliance while maintaining real-time adaptability, dynamic policy updates, scalability, and robust traceability.

To address these challenges, we introduce **BIG-ABAC**, a novel and flexible framework designed to overcome the inherent limitations of traditional access control systems. Unlike conventional models that rely on static policies or periodic updates, BIG-ABAC continuously adapts to policy modifications (e.g., regulatory changes) and contextual variations (e.g., patient location, emergency status). This ensures that access control decisions remain precise, responsive, and compliant, even as conditions evolve dynamically.

BIG-ABAC integrates all core components of the NIST ABAC standard into a cohesive and adaptive framework, enabling seamless real-time policy adaptation. These components work in synergy to ensure fine-grained, context-aware access control, supported by robust logging and compliance mechanisms (see Fig. 1).

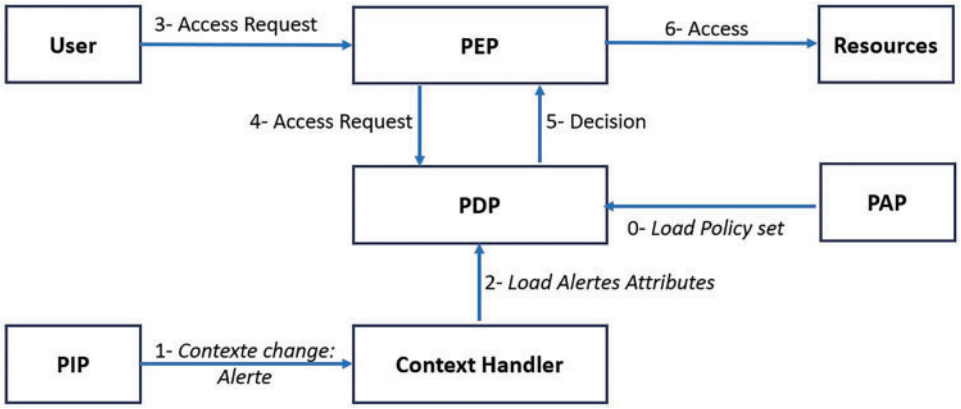

Figure 1: BIG-ABAC system workflow incorporating NIST ABAC components

By leveraging modern data processing and policy evaluation techniques, BIG-ABAC delivers unmatched scalability, responsiveness, and regulatory compliance. The framework features a dynamic decision-making engine capable of continuously recalculating access policies in real-time based on contextual updates. This ensures that access control decisions remain synchronized with operational demands, eliminating latency and bottlenecks associated with manual policy updates.

BIG-ABAC also strengthens traceability and regulatory compliance through comprehensive audit logging embedded within the PDP and a centralized data lake. These mechanisms capture essential metadata, such as timestamps, user actions, policy updates, and access decisions, enabling real-time anomaly detection, forensic analysis, and operational optimization. Unlike conventional systems, which often lack comprehensive logging, BIG-ABAC ensures consistent traceability across distributed environments, mitigating compliance vulnerabilities and inefficiencies.

Experimental evaluations validate the framework's efficacy, demonstrating a **40% reduction in latency** compared to conventional access control systems and confirming that **95% of concurrent access requests are processed within 50 ms**. These findings highlight BIG-ABAC's scalability, responsiveness, and ability to handle complex, dynamic workloads without performance degradation.

Although this study focuses on healthcare emergency scenarios, BIG-ABAC's capabilities extend across various high-demand sectors, including finance, smart cities, and critical infrastructure. Its ability to process real-time contextual data and execute dynamic policy updates makes it a transformative access control solution for environments requiring rapid, secure, and reliable data access.

The remainder of this paper is structured as follows: Section 2 provides an overview of existing access control mechanisms, identifying their limitations in dynamic and high-demand environments. Section 3 introduces the BIG-ABAC framework, detailing its architecture and core functionalities. Section 4 discusses the application of BIG-ABAC in emergency healthcare, focusing on its implementation and evaluation. Finally, Section 5 concludes with a summary of contributions and potential directions for future research.

2 Related Work

The integration of ABAC with scalable data processing and real-time decision-making is essential for ensuring secure, context-aware, and fine-grained access control, particularly in dynamic environments such as healthcare. Systems like electronic health records (EHR), IoT-enabled healthcare platforms, and emergency response services demand mechanisms that not only enforce privacy compliance but also support rapid, adaptive access control decisions [12]. Despite advancements, existing frameworks exhibit limitations in dynamic policy enforcement, real-time adaptability, and compliance with standardized access control models. Many approaches fail to efficiently handle high-concurrency scenarios while lacking robust traceability mechanisms for access accountability.

This section critically examines ABAC frameworks applied in healthcare and dynamic environments, assessing their flexibility, scalability, compliance, and traceability. The discussion highlights existing limitations and analyzes how BIG-ABAC overcomes these challenges through real-time policy updates, adaptive decision-making, and enhanced traceability mechanisms.

Chenthara et al. [3] proposed a Multi-Layer Access Control (MLAC) model that integrates Role-Based Access Control (RBAC), ABAC, and Discretionary Access Control (DAC) for managing EHR access. The introduction of pseudoroles allowed more flexible policy enforcement by merging static roles with dynamic attributes. However, the framework lacks real-time contextual updates and dynamic rule adjustments, limiting its ability to adapt to evolving access scenarios. BIG-ABAC enhances this by supporting continuous policy updates and automated adaptation of access rules based on live contextual changes, ensuring that policy enforcement remains dynamic and responsive.

Longstaff and Noble [4] introduced an ABAC framework that integrates access control policies within structured query mechanisms for databases. While this ensures fine-grained enforcement at the data level, the approach does not support dynamic policy changes without modifying the query structure. BIG-ABAC

overcomes this limitation by decoupling policy evaluation from storage management, enabling real-time updates to access control rules without impacting data access mechanisms.

Yang et al. [5] introduced Hierarchical Binary Matrix Coding Attribute-Based Access Control (HBMC-ABAC), which employs a binary coding method to improve policy-matching efficiency in large-scale environments. While this enhances scalability, the lack of support for dynamic policy updates reduces adaptability in evolving access scenarios. BIG-ABAC extends this approach by integrating mechanisms that dynamically adjust policies in response to changing operational conditions, enabling both scalability and real-time adaptability.

Pal et al. [6] developed a hybrid policy-based and ABAC framework for IoT healthcare. While their model enables fine-grained access decisions, its reliance on static policies reduces its ability to handle rapidly changing access control conditions. BIG-ABAC addresses these limitations by incorporating a flexible and scalable architecture that supports continuous policy modifications while maintaining system efficiency under high concurrency.

Psarra et al. [7] developed a context-aware ABAC framework for emergency healthcare scenarios, where policies dynamically adapt to patient conditions. While the model supports attribute-driven access control, it does not include mechanisms for dynamically updating policy rules as contextual conditions change in real-time. BIG-ABAC addresses this by integrating continuous rule recalculations based on evolving contextual data, ensuring immediate alignment between policy enforcement and real-world situations.

Butpheng et al. [8] introduced an ABAC model designed to enhance security and privacy in cloud-based IoT healthcare environments. Although their approach integrates contextual attributes for access decisions, it suffers from high latency and lacks a mechanism for real-time policy updates. BIG-ABAC improves upon this by utilizing a distributed architecture that maintains active synchronization between policies and access requests, ensuring minimal latency and high adaptability under high-demand conditions.

Salehi Shahraki et al. [9] presented Decentralized Multi-Authority Attribute-Based Access Control (DMA-ABAC), a blockchain-based decentralized access control model. While the framework ensures strong traceability by maintaining immutable access logs, the reliance on blockchain technology introduces delays in policy enforcement due to network consensus overhead. BIG-ABAC ensures a balance between traceability and performance by using structured and decentralized logging mechanisms that enable efficient auditing without introducing significant processing delays.

Walid et al. [10] proposed a privacy-preserving ABAC framework utilizing knowledge graphs and anonymization techniques to manage access to sensitive data. While effective in protecting privacy, the framework lacks mechanisms to accommodate real-time policy changes. BIG-ABAC enhances this approach by incorporating dynamic access control adaptations based on evolving user consent and regulatory requirements, ensuring compliance while preserving flexibility.

De Oliveira et al. [11] introduced Acute Care Attribute-Based Access Control (AC-ABAC), a dynamic access control model tailored for acute healthcare environments. Their model supports real-time contextual updates but lacks mechanisms for dynamic rule updates and comprehensive policy administration. BIG-ABAC extends this work by enabling continuous rule adjustments, ensuring that all access control components operate in synchronization and maintain full compliance with standardized access control frameworks.

Tall and Zou [13] explored the application of ABAC in Big Data environments using batch processing techniques. While their model provides extensive logging and policy verification, it does not support real-time adaptability, as access control decisions are applied in periodic batches. BIG-ABAC eliminates this

bottleneck by enabling immediate policy adjustments and decision-making based on live contextual inputs, ensuring continuous enforcement without processing delays.

The reviewed frameworks demonstrate significant advancements in access control technologies but highlight critical gaps in real-time policy enforcement, scalability, and compliance with standardized models. BIG-ABAC addresses these limitations by integrating an event-driven policy management system, enabling dynamic policy updates, and ensuring adaptive, traceable, and scalable access control.

Table 1 compares various ABAC frameworks based on four critical factors: flexibility, scalability, compliance with NIST ABAC guidelines, and traceability. While existing solutions provide incremental improvements in individual aspects, BIG-ABAC uniquely integrates all dimensions, addressing limitations such as static rule management, latency issues, and incomplete compliance mechanisms.

Table 1: Comparison of ABAC solutions in terms of flexibility, scalability, compliance, and traceability

Reference	Flexibility in contextual attribute and rules management	Scalability in data storage and processing	Compliance with NIST ABAC guidelines	Traceability of access decisions
[3]	Supports hybrid role-attribute mechanisms, lacks real-time updates	Limited scalability for dynamic access scenarios	Partial compliance, lacks components for policy administration	Logging for retrospective access control verification
[4]	Static query-based enforcement, no real-time adaptability	Scalability limited by query structure dependencies	No compliance with standardized frameworks	Query-level access monitoring
[5]	Binary code-based policy matching for efficiency, but lacks dynamic rule updates	High scalability for large-scale data environments	No support for dynamic updates in policy rules	Basic logging without adaptive traceability
[6]	Static policies limit adaptability	No support for large-scale dynamic contexts	No mention of integration with standardized models	Centralized access decision logging
[7]	Context-aware policies, lacks dynamic rule updates	Limited scalability due to preconfigured rule handling	Supports policy decision points but lacks dynamic updates	Partial traceability mechanisms
[8]	Integrates IoT context but lacks live updates	Latency issues under concurrent access requests	Partial compliance with access control models	Event logging without real-time validation
[9]	Decentralized access verification without real-time adjustments	Blockchain consensus latency impacts scalability	Partial, includes some policy evaluation mechanisms	Strong traceability but with processing overhead
[10]	Privacy-preserving access control with knowledge graphs but lacks real-time updates	Limited scalability due to static anonymization techniques	Partial compliance with ABAC models	Basic traceability without real-time policy adaptation
[11]	Dynamic contextual updates, lacks flexible rule modifications	Limited adaptability for evolving access conditions	Supports policy decision points but lacks comprehensive policy updates	Static auditing without dynamic adaptation
[13]	Batch-processing ABAC limits real-time adaptability	High scalability but lacks continuous rule enforcement	Partial compliance with ABAC principles	Periodic logging without instant traceability
BIG-ABAC	Fully dynamic rule updates with real-time adaptability	Distributed model ensures high concurrency processing	Comprehensive integration of access control models	Adaptive logging and real-time compliance tracking

Table 2 summarizes the methods and technologies employed by related ABAC frameworks, highlighting the innovative techniques and limitations. BIG-ABAC stands out by combining real-time rule updates with scalability and comprehensive compliance, providing a unified approach to modern access control challenges.

Table 2: Summary of methods used in related works

Reference	Main approach	Methods/technologies used	Limitations
[3]	Multi-Layer Access Control (MLAC) combining RBAC, ABAC, and DAC	Pseudoroles for role-attribute integration; EHR-specific policies	No real-time contextual updates; limited scalability
[4]	ABAC for Big Data through query modification	Embeds permissions in Structured Query Language (SQL)/NoSQL queries	Static queries; lacks dynamic rule updates
[5]	Scalable ABAC using binary codes for Big Data	Binary code-based policy matching for efficiency	Static codes; no support for dynamic rules
[6]	Policy-based ABAC for IoT healthcare	Combines policy-based and ABAC approaches for fine-grained access control	Relies on static policies; scalability limitations
[7]	Context-aware ABAC for emergency healthcare scenarios	Dynamic attribute handling using specialized Context Handlers	No support for dynamic rule updates; limited scalability
[8]	Security and privacy in IoT-cloud healthcare	Context-aware ABAC; IoT data integration with cloud systems	Latency issues; lacks real-time rule management
[9]	Decentralized ABAC using blockchain for healthcare	Blockchain for logging and decentralized access control	High latency in cross-authority environments
[10]	Privacy-preserving ABAC using knowledge graphs and anonymization	Knowledge graphs for data modeling; static anonymization for privacy	No dynamic rule updates; limited flexibility
[11]	Dynamic AC-ABAC model for acute healthcare	Contextual updates using dynamic Context Handlers and PDP integration	No dynamic rule updates; partial support for PAP/PEP
[13]	Batch-processing ABAC for Big Data with sensitivities	Batch processing for sensitive data policies; detailed logging	No real-time adaptability; lacks dynamic rule handling
BIG-ABAC	Real-time ABAC with dynamic rule updates and scalable architecture	Distributed streaming; adaptive PDP; policy enforcement logging; real-time contextual evaluation; dynamic attribute-based policy updates	Addresses limitations of dynamic updates, scalability, and traceability

3 BIG-ABAC System Description

To address the critical challenges of flexibility, scalability, and traceability in dynamic, high-demand environments, we propose BIG-ABAC, a novel framework explicitly designed to meet the access control demands of emergency response domains. BIG-ABAC integrates advanced data processing technologies and aligns with the NIST ABAC standard, providing a robust solution for real-time, contextually aware, and dynamic access control.

A key contribution of the BIG-ABAC architecture lies in its ability to seamlessly manage real-time policy updates and enforce context-sensitive decisions in rapidly changing scenarios, such as those encountered during healthcare emergencies or disaster response operations. By leveraging a distributed streaming platform and integrating all core NIST ABAC components, BIG-ABAC ensures efficient, secure, and responsive access control, even under extreme operational pressures.

At the core of the architecture:

- A distributed data streaming platform facilitates real-time data ingestion and processing.
- The Data Lake stores contextual changes, policy updates, and access logs for compliance and continuous model improvement.

The proposed architecture, as illustrated in Fig. 2, relies on a tightly coordinated workflow among its components to ensure seamless adaptability. Each component plays a distinct role in ensuring that access control decisions remain dynamic, precise, and contextually relevant.

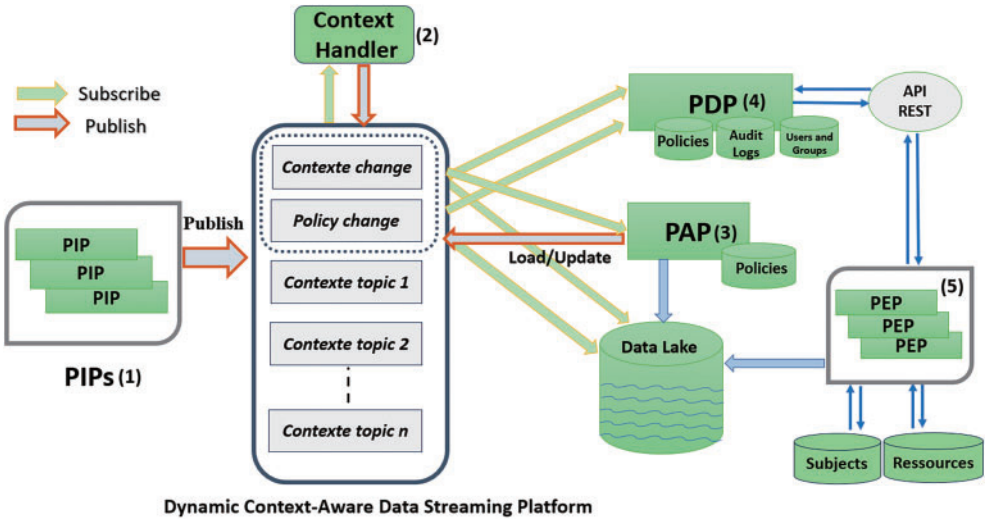

Figure 2: BIG-ABAC architecture

The following describes how each component contributes to this dynamic and coordinated workflow:

1. Policy Information Points (PIPs): PIPs serve as the primary data collection mechanisms, gathering real-time contextual information from various sources, including Application Programming Interfaces (APIs), environmental sensors, and external systems. This data encompasses critical attributes such as user roles, locations, timestamps, and emergency statuses. Once collected, the data is categorized and published to specific context topics within the distributed streaming platform. These topics are organized based on data type, relevance, or other logical groupings, ensuring that the system maintains a continuous and up-to-date understanding of the operational environment.
2. Context Handler: The Context Handler processes data from multiple context topics within the distributed streaming platform, applying detection algorithms to identify critical operational changes. These include:
   - *Threshold-Based Detection:* Identifies significant events, such as ICU admissions or fire alarm activations, based on predefined thresholds.
   - *Clustering Models:* Detects anomalies in access patterns, such as spikes in requests during emergencies or irregular employee access behavior, by grouping data points and flagging outliers.

Upon detecting a significant change, the Context Handler publishes updates to the *Context Change* topic, prompting the Policy Administration Point (PAP) to dynamically adjust access control policies in response to evolving conditions. Simultaneously, the Policy Decision Point (PDP) consumes these updates to refine real-time access decisions based on emergency contexts. All contextual updates are stored in the data lake, ensuring traceability and compliance with regulatory requirements.

3. Policy Administration Point (PAP): The PAP dynamically manages access control policies. Upon consuming real-time contextual updates from the context change topic, it notifies administrators to review and update policies, ensuring they align with new operational conditions. In emergency scenarios, the PAP can initiate temporary policy modifications, which are then published to the policy change topic. These updates allow access policies to dynamically adapt to real-time changes, ensuring the system remains responsive and compliant.
4. Policy Decision Point (PDP): The PDP serves as the central decision-making entity in BIG-ABAC, evaluating access requests based on:
   - *Policy Rules:* Implements ABAC rules to ensure compliance with security policies. It continuously processes live data streams from the *Policy Change* topic to incorporate newly generated or updated rules from the PAP.
   - *Contextual Attributes:* Integrates real-time contextual data from the *Context Change* topic, including risk levels, user locations, and access history.
   - *Real-Time Decision Computation:* Dynamically evaluates policy trees to determine access permissions based on the most up-to-date contextual and policy data.

   The PDP maintains continuous adaptability to evolving operational conditions while ensuring strict regulatory compliance. All access control decisions are systematically logged to support auditability and forensic analysis.
5. Policy Enforcement Point (PEP): The PEP executes the access control decisions made by the PDP, either granting or denying access to requested resources. It operates as the interface between the access control system and protected assets, ensuring enforcement aligns with dynamically evaluated policies. To maintain traceability and compliance, all access decisions and enforcement actions are systematically logged in the Data Lake. By enforcing PDP decisions in real-time, the PEP guarantees secure, policy-compliant access while preserving a comprehensive audit trail for regulatory oversight.

Example Scenario: Emergency Access Control

To illustrate the real-time adaptability of BIG-ABAC in handling multi-factor decision-making, consider the following emergency response scenario:

Scenario: Disaster Response in a Restricted Zone

Context: A firefighter responding to a large-scale fire at an industrial facility needs temporary access to restricted blueprints of the building for safe navigation and hazard identification. Under normal conditions, only authorized engineers have clearance to access these documents due to confidentiality constraints.

Access Control Process in BIG-ABAC:

- Step 1: Contextual Trigger - The PIP detects an emergency and registers the firefighter's entry into a restricted zone. - The Context Handler publishes this status change to the context change topic.
- Step 2: Policy Update - The PAP consumes the emergency event from the context change topic. - The PAP notifies the administrator of the emergency status and suggests a temporary policy update. - The updated policy, granting temporary access to first responders, is published to the policy change topic.
- Step 3: Multi-Factor Policy Evaluation (PDP) - The PDP evaluates the firefighter's request, factoring in:
  - Role: Firefighter (normally restricted).

  - Context: Emergency activation (grants temporary access).
  - Geolocation: Inside the disaster zone.
  - Risk Level: High (requires immediate intervention).
  - Based on these multi-factor attributes, the PDP temporarily grants access.

- Step 4: Access Enforcement (PEP) - The PEP enforces the decision, allowing access to the blueprints. - The decision is logged in the Data Lake for traceability and post-incident review.
- Step 5: Automatic Policy Revocation - Once the firefighter exits the disaster zone, the policy is revoked. - Any unauthorized future access attempts trigger an alert.

Outcome: BIG-ABAC ensures that access permissions adapt dynamically based on real-time situational data. By integrating policy updates with administrative oversight, it prevents unauthorized access while ensuring that emergency responders receive necessary information efficiently. Through multi-factor evaluation, BIG-ABAC dynamically adjusts access control permissions, ensuring security, scalability, and responsiveness across diverse operational domains.

4 Implementation and Evaluation

This section provides an overview of BIG-ABAC in the context of emergency healthcare, as well as the implementation and evaluation of the flexible framework. Emergency healthcare was chosen as a critical use case to demonstrate the effectiveness and security of the framework, given its stringent requirements for real-time processing, scalability, and security. The implemented architecture represents a practical adaptation of the general BIG-ABAC framework, specifically tailored to address the unique needs and operational constraints of healthcare scenarios while maintaining the flexibility and scalability needed for broader applications in high-demand domains such as finance and smart cities.

The prototype was developed on a 14-inch MacBook Pro (Apple M3 Pro chip, 18 GB memory, 1 TB storage), providing the necessary computational power to manage large-scale data streams, dynamically update decision trees, and enforce access control policies effectively. The evaluation focused on performance metrics such as **flexibility**, **scalability**, and **traceability**, which are essential for dynamic environments requiring timely and secure access to sensitive information.

4.1 Framework Overview

This section presents the BIG-ABAC framework, outlining its architecture and key components designed to address the challenges of dynamic, high-demand environments. The framework ensures real-time adaptability, scalability, and traceability, making it suitable for emergency healthcare and other critical domains that require secure, context-aware access control.

BIG-ABAC operates as a dynamic and event-driven access control system, integrating real-time context monitoring to ensure policies remain aligned with evolving conditions. The framework processes continuous streams of contextual updates, enabling policy enforcement that remains responsive to operational changes.

To achieve real-time adaptability, BIG-ABAC is structured around several key components that enable policy management, decision-making, enforcement, and auditability.

Apache Kafka [14] serves as the backbone of the BIG-ABAC framework, providing high-throughput and low-latency streaming for real-time policy and context updates. It enables efficient real-time processing of contextual changes, ensuring that access control decisions are based on the most up-to-date information. The system is designed for scalability, allowing policy evaluations to be distributed across multiple nodes, which enhances parallel processing and prevents bottlenecks. Additionally, Kafka facilitates seamless message

distribution, ensuring that all access control components receive synchronized updates without delays or inconsistencies.

BIG-ABAC utilizes specialized Kafka topics to ensure that access control remains dynamically adaptive. These topics process and distribute real-time data for fine-grained decision-making:

- Ambulance Status: Tracks real-time ambulance locations and availability.
- Patient Location: Provides dynamic patient location updates for access decisions.
- Patient Consent: Streams consent updates to ensure compliance with privacy policies.
- Environmental Conditions: Monitors hazards such as weather and air quality that may impact emergency responses.
- Policy Change: Distributes updated access control policies dynamically.
- Context Change: Publishes contextual updates such as patient conditions or emergency status to refine decisions.

To achieve this level of adaptability, BIG-ABAC is structured around key components that enable policy management, decision-making, enforcement, and auditability. Each component plays a critical role in ensuring seamless operation across high-demand environments. The subsequent sections delve into the roles of these components and the topics that facilitate real-time policy updates and contextual data processing. Fig. 3 illustrates the architecture.

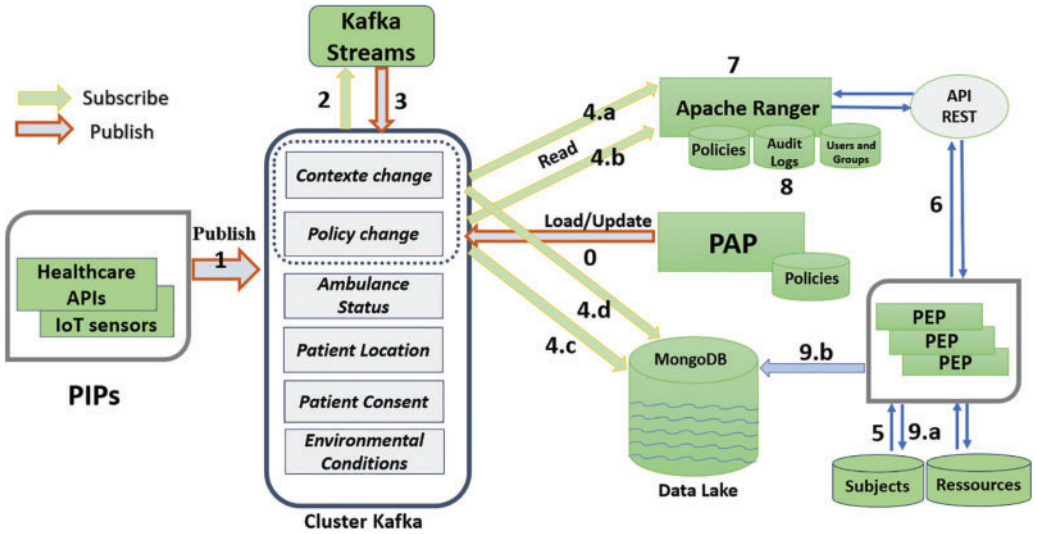

Figure 3: BIG-ABAC architecture in emergency healthcare

Core Components for Emergency Healthcare Access Control:

- **PIP–Healthcare APIs and IoT Sensors:** PIPs gather real-time contextual attributes (e.g., vitals, locations) and publish them to topics for immediate processing. This ensures decisions are based on accurate, up-to-date information, enhancing **precision** and responsiveness.
- **Dynamic Streaming Platform–Apache Kafka:** Apache Kafka ensures **real-time data flow**, processing contextual updates and policy changes with horizontal scalability. It supports dynamic policy updates without disruptions, critical for high-demand scenarios like emergency healthcare.
- **Context Handler–Kafka Streams:** Kafka Streams processes and analyzes data from multiple topics, detecting significant contextual changes. The results are published to the "*Context Change*" topic,

enabling real-time adaptation to evolving scenarios. It communicates with the Kafka cluster using Kafka's native binary protocol over TCP, ensuring efficient, high-throughput data streaming in a distributed environment.

- **PAP–Kafka Client and API:** The PAP is responsible for managing and updating access control policies dynamically based on administrative decisions and contextual updates. It interacts with Kafka as both a producer and consumer to ensure real-time policy synchronization. The workflow follows these steps:
  - An administrator modifies policy rules.
  - The PAP publishes the updated policy to the "Policy Change" topic in Kafka.
  - Kafka distributes the updated policies to all subscribed components.

Policies are structured in JSON format and published asynchronously, ensuring minimal latency and high availability.

- **PDP–Apache Ranger [15]:** Apache Ranger serves as the PDP, dynamically evaluating and enforcing access control policies by integrating real-time updates from Kafka. It ensures adaptive and scalable decision-making while maintaining compliance with regulatory requirements. The PDP operates as follows:
  - **Real-Time Policy Updates:** Apache Ranger acts as a Kafka consumer, continuously retrieving and applying policy modifications published to the "*Policy Change*" topic. This guarantees that access control policies remain updated without requiring manual intervention.
  - **Context-Aware Decision-Making:** The PDP subscribes to the "*Context Change*" topic, dynamically adjusting access control decisions based on real-time changes in user roles, patient conditions, and environmental factors.
  - **Policy Synchronization via a Representational State Transfer (REST) API:** The `applyPolicyChange` function utilizes HTTP PUT requests to update Apache Ranger policies programmatically, ensuring synchronization between policy definitions and access control enforcement in real-time.
  - **Access Request Processing:** The PDP receives access requests from the PEP via REST API. Each request includes user attributes, requested resources, and contextual parameters. Apache Ranger evaluates the request against the latest policies and contextual updates before issuing an access decision.
  - **Seamless Policy Enforcement:** Once a decision is made, the PDP returns a response to the PEP, which enforces the decision (grant or deny). All actions are logged in MongoDB for audit, traceability, and compliance analysis.
  - **High Availability and Scalability:** By leveraging Kafka's distributed streaming architecture, multiple PDP instances operate in parallel, ensuring fault tolerance and scalability under high-concurrency scenarios.
- **PEP–Healthcare APIs:** The PEP acts as the enforcement layer, ensuring that access control decisions made by the PDP are executed in real-time:
  - Receives access requests from healthcare applications.
  - Forwards requests to the PDP via REST API for evaluation.
  - Enforces the PDP's decision (grant or deny access) to ensure compliance with active policies.
  - Logs all access attempts, decisions, and enforcement actions into MongoDB for audit and traceability.
- **Data Lake–MongoDB [16]:** MongoDB serves as a centralized data lake, storing data from the "*Context Change*" and "*Policy Change*" topics to ensure traceability and regulatory compliance. It supports

auditing, advanced analytics, and policy optimization. Additionally, it logs all access attempts, policy decisions, and enforcement actions, providing a comprehensive audit trail for security and compliance monitoring.

The proposed framework transforms Apache Ranger from a static PEP into a dynamic PDP by integrating it with Kafka for real-time policy evaluation and seamless contextual updates. This enhancement enables instant decision recalculations, ensuring adaptive and context-sensitive access control in fast-evolving environments.

To minimize latency, lightweight APIs serve as PEPs, enforcing PDP decisions with near-instant response times. Kafka topics orchestrate policy synchronization and contextual updates. The Policy Change topic ensures timely rule updates, while the Context Change topic dynamically integrates real-time data, such as patient conditions and resource availability. By decoupling policy updates from contextual changes, this design prevents disruptions, enhances clarity, and optimizes system performance.

Apache Kafka guarantees scalability with high-throughput, low-latency event streaming, prioritizing critical operations like emergency alerts. Meanwhile, MongoDB acts as a centralized audit and analytics hub, consolidating logs, contextual updates, and compliance records. The result is a system that maintains robust traceability while ensuring flexible and privacy-preserving access control.

Together, these components create a highly responsive, real-time framework that dynamically adjusts to patient conditions, environmental risks, and evolving access policies without performance degradation. Fig. 3 visually illustrates this seamless interaction, highlighting BIG-ABAC's adaptability and precision in critical healthcare scenarios.

The BIG-ABAC framework enforces dynamic access control rules, ensuring policies remain aligned with real-time changes. These rules include:

- R1 - Emergency Detection: Activates flexible access pathways upon detecting a medical emergency.
- R2 - Medical Specialty: Grants access based on a healthcare professional's specialization.
- R3 - Consent: Enforces patient or representative consent before access is granted.
- R4 - Patient Assignment: Restricts access to only the EHR of the patient under active care.
- R5 - Time of Access: Ensures requests are made within authorized time frames.
- R6 - Location: Limits access to predefined geographic zones (e.g., hospital, ambulance).
- R7 - Treatment Requirement: Grants access only when data is essential for immediate care.
- R8 - Data Update Authorization: Allows patient data updates only by authorized personnel.
- R9 - Start Emergency Session: Enables authorized professionals to initiate emergency response.
- R10 - End Emergency Session: Restricts emergency session closure to assigned personnel.

When an acute healthcare emergency is triggered, requiring immediate access to sensitive resources, the BIG-ABAC framework activates the following steps to ensure secure, dynamic, and context-aware access control, as illustrated in Fig. 3. The process unfolds as follows:

- **Policy Initialization and Update (Step 0):** The PAP is responsible for defining and updating access control policies based on management authority and operational needs, such as during a healthcare crisis. Initially, PAP initializes baseline policies, which are loaded into Apache Ranger via the "Policy Change" topic. When updates to policy rules are required (e.g., relaxing access restrictions during an emergency), PAP dynamically modifies the policies and notifies the PDP through the "Policy Change" topic for immediate application.
- **Data Collection (Step 1):** IoT sensors and healthcare APIs collect real-time contextual data, such as patient vitals, ambulance status, and environmental conditions. This data is published to dedicated Kafka topics (e.g., "Ambulance Status," "Patient Location," "Environmental Conditions") for processing.

- **Context Handling (Step 2–3):** Kafka Streams aggregates and processes the data from context topics, identifying critical operational changes or generating alerts. The processed data is then published to the "Context Change" topic to update downstream components.
- **Context Update and Policy Synchronization (Step 4):**
  - **Step 4.a: Context Updates via Kafka**
    - Apache Ranger consumes updates from the "*Context Change*" topic to dynamically adjust its decision-making logic.
    - This enables real-time contextual adaptations, such as integrating patient status updates and environmental conditions into access control evaluations without manual intervention.
  - **Step 4.b: Policy Updates via Kafka**
    - The PAP publishes updated policies to the "*Policy Change*" topic in Kafka.
    - The Kafka Consumer (running within the PDP) subscribes to this topic and retrieves the new policy updates.
  - **Step 4.c: Policy Synchronization in Apache Ranger**
    - The `applyPolicyChange` function in the PDP translates Kafka policy updates into a format compatible with Apache Ranger.
    - The updated policies are applied to Apache Ranger via HTTP PUT requests using the Ranger REST API.
    - This ensures immediate enforcement of new policies without requiring system downtime.
  - **Step 4.d: Context and Policy Storage in MongoDB**
    - MongoDB, as part of the data lake, subscribes to the "*Policy Change*" topic to store all policy updates in real-time for auditability and compliance tracking.
    - This ensures retrospective analysis of policy changes and facilitates regulatory compliance verification.
    - MongoDB also consumes updates from the "*Context Change*" topic, storing all contextual changes for traceability and historical data analysis.
    - By efficiently indexing these updates, MongoDB supports rapid query execution for policy refinement and anomaly detection in emergencies.
- **Access Request Forwarding (Step 5–6):** A healthcare professional submits an access request (e.g., to retrieve an electronic health record). The PEP receives this request and forwards it to the PDP for evaluation.
- **Access Decision (Step 7):** The PDP evaluates the access request in real-time, using the latest policies from the "Policy Change" topic and contextual data from the "Context Change" topic. Based on this evaluation, the PDP decides to either grant or deny access.
- **Audit Logging (Step 8):** Upon issuing a decision, the PDP logs the decision, including the timestamp, user action, contextual attributes, and policy details, in the Data Lake. This creates a complete journal of operational events for forensic analysis, compliance, and monitoring.
- **Enforcing the Decision (Step 9):**
  - (Step 9.a): The PEP enforces the PDP's decision, granting or denying access to the requested resource.
  - (Step 9.b): MongoDB records the access event, including the granted/denied status, for further traceability and compliance monitoring.

This workflow clarifies the distinct responsibilities of each component, from data ingestion and context handling to policy synchronization, decision-making, and enforcement. By efficiently handling dynamic contextual changes, BIG-ABAC ensures scalability, real-time adaptability, robust traceability, and compliance

with regulations like HIPAA and GDPR. The modular design also allows seamless integration into various emergency healthcare scenarios, meeting the critical demands of high-stakes domains.

BIG-ABAC incorporates several optimizations to enhance performance under high-demand conditions. The system employs load balancing across multiple PDP instances, distributing incoming requests efficiently to prevent bottlenecks. Frequently accessed policy rules and contextual attributes are cached within PDP memory, reducing redundant queries and minimizing response latency. Multi-threaded execution allows the PDP to evaluate multiple concurrent access requests, ensuring that system responsiveness is maintained even under peak demand. These optimizations collectively enhance BIG-ABAC's ability to handle large-scale, real-time access control scenarios while maintaining low latency and high availability.

To ensure robustness and resilience, BIG-ABAC integrates structured error-handling mechanisms. Kafka's Dead-Letter Queue (DLQ) captures and isolates faulty messages, preventing unprocessable records from disrupting system operations. If an error occurs during policy retrieval, the PDP enforces the last successfully applied policy to maintain operational continuity. The PEP implements a retry mechanism for failed access requests, ensuring that temporary disruptions do not result in service denial. Additionally, MongoDB logs all failed access attempts and system anomalies, enabling real-time monitoring and post-incident analysis. These mechanisms collectively ensure fault tolerance and reliability, even in high-demand emergency scenarios.

BIG-ABAC incorporates automated version control and structured software update mechanisms to maintain system integrity. All policy versions are stored in MongoDB, allowing administrators to track historical changes and roll back to previous configurations if needed. The system employs containerized deployment strategies using Docker and Kubernetes, enabling seamless upgrades of the PAP, PDP, and PEP components without service disruption. Before applying updates to the production environment, policy modifications undergo validation in a controlled testing environment to minimize potential risks. Kafka ensures reliable propagation of policy updates by enforcing message ordering and validation, guaranteeing that only correctly formatted policies are applied. These version control strategies ensure that BIG-ABAC remains stable, adaptable, and continuously aligned with evolving security and regulatory requirements.

Although initially developed for emergency healthcare, the modular design of BIG-ABAC allows seamless integration into other high-stakes environments requiring secure, context-aware access control. The ability to process real-time contextual data and enforce dynamic policy updates makes it applicable to IoT ecosystems, financial services, smart cities, and industrial automation.

For further details and implementation examples, refer to the GitHub Repository [17].

4.2 BIG-ABAC Evaluation

The performance of BIG-ABAC was evaluated comprehensively across three key metrics: **flexibility**, **scalability**, and **traceability**. A set of 15 dynamic test cases inspired by real-world healthcare scenarios from AC-ABAC [11] were developed. These tests simulated dynamic contextual attributes, policy updates, and high-concurrency access demands. The goal was to benchmark BIG-ABAC against AC-ABAC, demonstrating its superior performance in adaptive, scalable, and compliant access control. Detailed results and simulation scenarios are publicly accessible in the repository [17].

Flexibility

To evaluate the efficiency of dual adaptability, a simulated emergency healthcare scenario was implemented using the **AnyLogic platform's agent-based and event-driven simulation environment**. The simulation tested the framework's ability to dynamically adjust both policy rules and contextual inputs. The process is detailed in Algorithm 1, implemented in AnyLogic. The workflow involved:

- The PAP publishing an emergency policy to the *Policy Change* topic.
- The PDP dynamically recalculates the decision tree in real-time, integrating updates from both *Policy Change* and *Context Change* topics.
- Concurrent access requests from healthcare professionals (e.g., specialists, general personnel, and unauthorized users) processed to test responsiveness and decision accuracy.

Algorithm 1: Dual adaptability simulation for dynamic policy updates (AnyLogic)

Require: Emergency Trigger, Arrival Rate λ, Service Rate μ, Simulation Time T
Ensure: Policy Update Latency, Context Update Latency, Access Decision Latency
Initialize agents for PAP, PDP, Kafka, and MongoDB
Define Emergency Trigger at $t_{\text{emergency}}$
while $t < T$ **do**
    Generate access requests with inter-arrival times $\Delta t \sim \text{Exp}(1/\lambda)$
    if $t = t_{\text{emergency}}$ **then**
        PAP publishes an emergency policy to *Policy Change* topic
        PDP recalculates decision tree using updates from *Policy Change* and *Context Change* topics
    end if
    Process requests through PDP and log decisions in MongoDB
end while
Output metrics: Policy Update Latency, Context Update Latency, Access Decision Latency

Fig. 4 illustrates the efficiency of BIG-ABAC's dual adaptability. Before the update, the decision tree strictly enforced Rule R2, limiting access to specialists. After the update, the recalculated decision tree dynamically broadened access to include general healthcare personnel, ensuring immediate availability of critical patient data during emergencies. This validates BIG-ABAC's capability to adapt policies and contextual attributes in real-time while maintaining compliance.

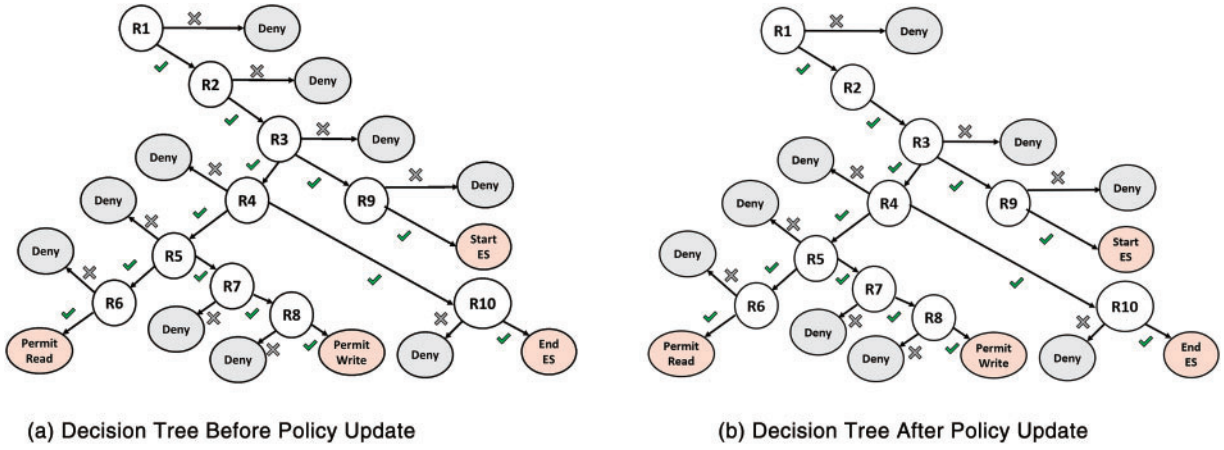

Figure 4: Comparison of decision trees before and after policy update

Performance metrics are summarized in Table 3.

These results underscore BIG-ABAC's ability to deliver dynamic, low-latency access control decisions by seamlessly adapting to both policy changes and contextual updates. This dual adaptability positions

the framework as a robust solution for scenarios requiring immediate and precise access control in high-stakes domains.

Table 3: Dual adaptability test results

Metric	Result	Interpretation
Policy update latency	30 ms	Seamless integration of new policies with negligible delay.
Context update latency	25 ms	Ensures immediate responsiveness to contextual changes.
Access decision latency	45 ms	Maintains rapid decision-making under updated policies and contexts.
Access requests handled	95% within 50 ms	Efficiently processes concurrent requests under dual updates.
Comparison with legacy systems	40% reduction in latency	Demonstrates superior adaptability over static frameworks.

The BIG-ABAC framework demonstrates exceptional flexibility by enabling real-time adaptation of policies and contextual attributes, achieving dual adaptability. This dual adaptability ensures seamless updates to both dynamic policy rules and evolving contextual inputs. Unlike AC-ABAC [11], which relies on static policies requiring manual updates, BIG-ABAC leverages real-time data streams from IoT sensors and APIs to dynamically adjust access rules without human intervention. This capability is critical in scenarios like emergency healthcare, where rapid policy adjustments and context-sensitive decisions are necessary to prevent bottlenecks and ensure operational efficiency.

BIG-ABAC adheres to the principles outlined in **NIST SP 800-162**, ensuring compliance with standards for scalable and dynamic access control. The framework achieves flexibility through several mechanisms:

- **Dynamic Attribute Processing:** Apache Ranger, functioning as the Policy Decision Point (PDP), processes contextual attributes such as user roles, patient conditions, and environmental factors in real-time, ensuring policies remain contextually relevant.
- **Efficient Policy Updates:** Kafka's *Policy Change* topic synchronizes updates from PAP to the PDP, ensuring timely and consistent policy enforcement.
- **Real-Time Contextual Updates:** Contextual changes published to the *Context Change* topic allow the PDP to dynamically recalibrate decision trees, enabling granular and adaptive access control decisions.

These features collectively enhance operational efficiency, ensuring precise and context-sensitive decisions in dynamic, high-demand environments.

Scalability

The scalability of BIG-ABAC was evaluated using **AnyLogic's discrete-event simulation**, which modeled the system under increasing access request loads. The goal was to assess latency, throughput, and adaptability compared to AC-ABAC. The simulation systematically increased the request arrival rate to observe the system's ability to maintain performance in high-demand environments.

Request bursts were simulated by gradually increasing arrival rates from 100 to 1000 requests per second, while monitoring latency and throughput. Policies were dynamically updated to assess adaptability under high loads. The scalability simulation process is detailed in Algorithm 2, which describes the structured approach used in AnyLogic to generate requests, detect high loads, and trigger dynamic policy updates.

Algorithm 2: Scalability simulation in AnyLogic

Require: Initial Rate λ_0, Max Rate λ_{max}, Simulation Time T
Ensure: Metrics: Latency, Throughput
1: Initialize AnyLogic environment
2: **while** $t < T$ **do**
3: Generate requests with increasing λ
4: **if** High Load Detected **then**
5: Trigger dynamic policy updates
6: **end if**
7: Log Latency and Throughput
8: **end while**
9: Output Metrics

The evaluation focused on the following metrics:

- **Latency:** BIG-ABAC maintained an average latency of 40 ms under 1000 concurrent requests, significantly lower than AC-ABAC's 70 ms. As shown in Fig. 5, AC-ABAC exhibited exponential latency growth, whereas BIG-ABAC sustained consistent performance due to its distributed streaming architecture and real-time policy evaluation.
- **Throughput:** BIG-ABAC processed 95% of requests within 50 ms, outperforming AC-ABAC, which achieved only 78%. This high throughput underscores BIG-ABAC's efficiency in handling concurrent access requests.
- **Adaptability:** BIG-ABAC dynamically adjusted policies in real time, ensuring uninterrupted decision-making during peak demand. In contrast, AC-ABAC required manual policy recalibrations under high loads, causing performance degradation.

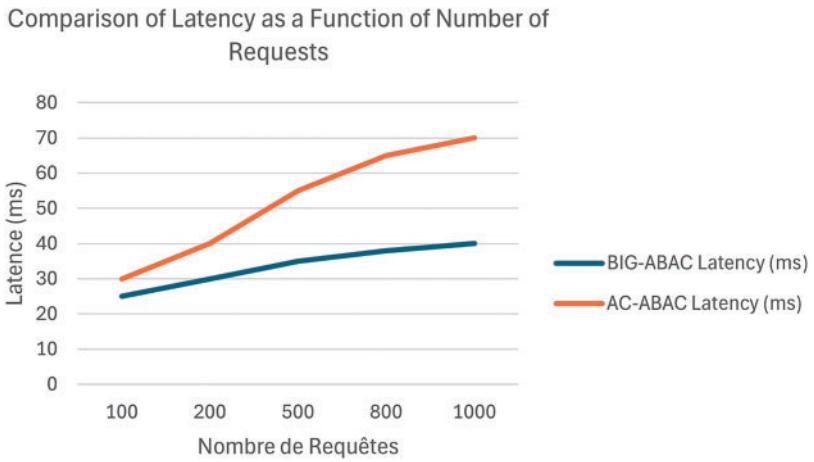

Figure 5: Comparison of latency as a function of number of requests

These results highlight the role of BIG-ABAC's distributed streaming architecture in achieving scalability. Kafka efficiently distributes workloads across brokers, while Apache Ranger updates policies without system downtime. MongoDB and PostgreSQL ensure seamless logging and policy storage, maintaining

traceability and performance under high loads. The framework's scalability was further enhanced by the following optimizations:

- **Load Balancing:** Kafka partitions access request topics, distributing workloads evenly across PDP instances.
- **Distributed Caching:** Frequently accessed policy decisions are cached within PDP nodes, reducing PostgreSQL query dependence and improving response times.
- **Parallel Processing:** Apache Ranger PDP instances handle multiple access requests simultaneously, significantly reducing queuing times.

Simulation results showed that these optimizations allowed BIG-ABAC to maintain an average latency of 40 ms under 1000 concurrent requests while ensuring high throughput. The reduction in latency directly translates to improved system responsiveness under high concurrency. Given that BIG-ABAC reduced access decision latency by 40% compared to AC-ABAC, this optimization ensures that access control evaluations remain efficient even when scaling to millions of concurrent users. The system can process a significantly higher number of access requests within the same time frame, mitigating potential bottlenecks that could arise from high-frequency access demands.

Furthermore, as the number of concurrent users increases, the combination of Kafka's horizontal scaling, Ranger's parallelized PDP evaluations, and distributed caching mechanisms ensures that decision latencies remain low, preventing exponential growth in processing time. This makes BIG-ABAC particularly suited for large-scale, real-time access control scenarios, such as nationwide healthcare infrastructures or industrial IoT environments where policy evaluations must be conducted at scale without performance degradation.

By maintaining low latency, high throughput, and real-time adaptability, BIG-ABAC provides a scalable and reliable access control solution for critical applications, such as healthcare, smart cities, and critical infrastructure. The simulation demonstrated BIG-ABAC's ability to consistently perform under increasing loads, making it superior to AC-ABAC for resource-intensive, dynamic environments.

Traceability

To assess the traceability capabilities of BIG-ABAC, we evaluated its logging, monitoring, and compliance mechanisms under high-demand scenarios. The evaluation focused on the framework's ability to capture and synchronize audit logs in real-time, ensuring consistency and accountability across its distributed architecture.

The traceability assessment was conducted using a simulated emergency healthcare scenario, leveraging the **AnyLogic platform** to model dynamic access control processes. The evaluation included:

- **Access Request Logging:** All access requests from healthcare professionals, including timestamps, user identifiers, and requested resources, were logged in a PostgreSQL-backed audit table within Apache Ranger.
- **Policy Update Monitoring:** Updates to policy rules, triggered by contextual changes (e.g., patient emergencies), were tracked in the *Policy Change* topic and logged in a centralized MongoDB-based data lake.
- **Decision and Enforcement Logging:** The PEP captured and logged key events, such as decisions received from the PDP and enforcement actions (grant/deny). These logs were enriched with metadata for compliance and forensic analysis.

Table 4 illustrates an example of the Apache Ranger Audit Log, highlighting the detailed event records captured during access control processes. These logs, stored in a PostgreSQL audit table, provide a structured and searchable format, enabling efficient troubleshooting and compliance monitoring.

Table 4: Ranger audit log example

Timestamp	Resource	Component	Action	Request data	Client type	Client IP	Result	Agent host	Log file path	Severity level
2024-09-01 12:34:56 PM	/healthcare/ehr/patient12345	EHR-Handler	READ	SELECT * FROM patient12345	Node.js API	192.168.1.10	PERMIT	agent1	/var/log/ranger/ehr/audit-123.log	Info
2024-09-01 12:38:10 PM	/healthcare/ehr/patient12345	EHR-Handler	READ	SELECT * FROM patient12345	Node.js API	192.168.1.20	DENY	agent2	/var/log/ranger/ehr/audit-124.log	Warn
2024-09-01 12:45:20 PM	/healthcare/ehr/patient67890	EHR-Handler	UPDATE	UPDATE patient67890 SET data...	Python Client	192.168.1.30	PERMIT	agent3	/var/log/ranger/ehr/audit-125.log	Info
2024-09-01 12:50:15 PM	/healthcare/ehr/patient23456	EHR-Handler	UPDATE	UPDATE patient23456 SET data...	Node.js API	192.168.1.40	DENY	agent4	/var/log/ranger/ehr/audit-126.log	Error
2024-09-01 01:00:40 PM	/healthcare/ehr/session123	EHR-Handler	END SESSION	END session123	Python Client	192.168.1.50	DENY	agent5	/var/log/ranger/ehr/audit-127.log	Warn

To quantify the effectiveness of traceability, the following metrics were monitored:

- **Log Consistency:** Measured as the percentage of synchronized logs across MongoDB and PostgreSQL repositories under high-concurrency scenarios.
- **Latency of Log Updates:** Time taken to capture and propagate an event log from the PEP to the centralized data lake.
- **Audit Completeness:** Ratio of logged events to total system events (requests, decisions, and enforcement actions).
- **Anomaly Detection Time:** Time required to flag and log unauthorized access attempts or misconfigured policies.

The performance metrics for traceability are summarized in Table 5.

Table 5: Traceability evaluation results

Metric	Result	Interpretation
Log consistency	100%	Ensures all events are captured and synchronized across MongoDB and PostgreSQL.
Log update latency	30 ms	Demonstrates real-time responsiveness in logging critical events.
Audit completeness	98%	Captures nearly all system events, ensuring robust traceability.
Anomaly detection time	40 ms	Enables rapid detection and logging of unauthorized access attempts.

To highlight the advancements of **BIG-ABAC**, its performance was compared against AC-ABAC in terms of traceability. The results indicate:

- **Enhanced Log Synchronization:** BIG-ABAC ensured real-time synchronization between MongoDB and PostgreSQL, whereas AC-ABAC exhibited delays exceeding 200 ms.
- **Improved Anomaly Detection:** BIG-ABAC flagged unauthorized access attempts in 40 ms, compared to AC-ABAC's average detection time of 80 ms.
- **Comprehensive Auditing:** BIG-ABAC achieved a 98% audit completeness rate, while AC-ABAC managed only 85%, primarily due to its reliance on static policies and limited logging mechanisms.

The traceability evaluation validates BIG-ABAC's ability to deliver robust and real-time logging mechanisms that ensure compliance, accountability, and anomaly detection. The framework's seamless integration of MongoDB and PostgreSQL ensures synchronized, detailed logs that facilitate compliance with regulatory requirements such as GDPR and HIPAA, providing auditability and traceability for policy enforcement.

These capabilities position BIG-ABAC as a superior choice for high-stakes domains requiring stringent traceability. Future work will focus on enhancing the scalability of the logging infrastructure to accommodate exponentially growing data volumes in large-scale deployments.

The evaluations of **BIG-ABAC** across **flexibility**, **scalability**, and **traceability** underscore its significant advancements in addressing the challenges of dynamic, high-demand environments. Leveraging the **AnyLogic simulation tool**, the framework was rigorously analyzed under real-world-inspired scenarios, providing comprehensive insights into its operational capabilities.

In terms of **flexibility**, BIG-ABAC demonstrated seamless real-time policy adaptation with a policy update latency of **30 ms** and an access decision latency of **45 ms**. Simulated emergency scenarios validated its dual adaptability, enabling dynamic adjustments to both contextual attributes and policy rules without manual intervention.

For **scalability**, BIG-ABAC sustained consistent performance under increasing workloads, processing **95% of requests within 50 ms** and maintaining an average latency of **40 ms**. Comparative analyses against **AC-ABAC** highlighted BIG-ABAC's ability to handle high-concurrency environments efficiently, avoiding performance degradation even under peak demands.

Regarding **traceability**, the MongoDB-based data lake and PostgreSQL-backed audit logs provided a robust and centralized logging infrastructure. The framework achieved real-time compliance monitoring and forensic capabilities, with synchronized audit logs capturing **98% of system events**. Additionally, anomaly detection and regulatory adherence were validated through simulations, reinforcing its readiness for high-stakes domains.

By leveraging *Apache Kafka*, *Apache Ranger*, and *MongoDB*, BIG-ABAC overcomes the limitations of static frameworks such as *AC-ABAC*, enabling real-time policy adaptation and contextual decision-making. Its scalable and context-aware architecture makes it well-suited for critical applications in *healthcare*, *smart cities*, and *Industrial IoT*, ensuring efficient and dynamic access control in high-demand environments.

BIG-ABAC's scalability and adaptability extend its applicability beyond healthcare, particularly in Industrial IoT (IIoT) and smart cities. For instance, reference [18] proposes an IIoT access control framework that dynamically enforces policies to mitigate unauthorized behaviors. Future research can optimize BIG-ABAC for large-scale, real-time security applications in similar domains. Another promising direction is the integration of Machine Learning (ML) for predictive access control. By analyzing user behavior, environmental conditions, and historical data, ML models can anticipate access needs and detect anomalies, enhancing security while minimizing policy misconfigurations. Further work should focus on improving scalability under high-concurrency workloads through distributed caching, load balancing, and parallel processing. Automating compliance enforcement would streamline policy updates and reduce administrative overhead, while enhanced audit and anomaly detection mechanisms would strengthen traceability and security. Evaluating BIG-ABAC's efficacy in IIoT, smart cities, and critical infrastructure is essential. Incorporating ML-driven policy adaptation and automated compliance mechanisms can advance BIG-ABAC into a next-generation access control framework capable of addressing evolving security challenges.

5 Conclusion

BIG-ABAC introduces a dynamic and adaptable approach to ABAC by enabling real-time updates to policy rules and contextual attributes. This capability addresses the limitations of traditional ABAC systems, which rely on static policies and static configurations, making them less effective in environments with continuously evolving access control requirements. Static policies define fixed access rules that do not account for real-time contextual changes, while static configurations limit the ability to dynamically adapt to new operational constraints and security conditions. Unlike conventional frameworks, BIG-ABAC leverages a distributed architecture that supports high-concurrency workloads while maintaining low-latency performance and robust traceability. By integrating continuous policy and context updates, the framework ensures uninterrupted, fine-grained access control in mission-critical environments such as healthcare, smart cities, and industrial automation. Performance evaluations demonstrated BIG-ABAC's efficiency, processing 95% of access requests within 50 ms and dynamically updating policies with a latency of 30 ms, significantly outperforming conventional ABAC models. Furthermore, its centralized logging and auditing mechanisms enhance regulatory compliance, forensic analysis, and operational transparency.

Future research will focus on extending BIG-ABAC's capabilities through predictive analytics and AI-driven anomaly detection to enable proactive security measures and improve decision-making under uncertainty. These advancements will solidify BIG-ABAC's role as a scalable and adaptable solution for next-generation access control systems across diverse domains.

Acknowledgement: Not applicable.

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

Author Contributions: Sondes Baccouri conceived and designed the study, implemented the framework and drafted the manuscript. Takoua Abdellatif contributed to the framework's evaluation and revised the manuscript. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The BIG-ABAC GitHub Repository [17] provides detailed resources, including:

- **Test Scenarios:** A document describing the framework's evaluation scenarios.
- **Dynamic Rules Documentation:** Details of rules for emergency healthcare scenarios.
- **Installation Guide:** Instructions for setting up and deploying the framework.
- **Technologies Used in BIG-ABAC:** A comprehensive document outlining the underlying technologies, implementation components, and system architecture.

These resources offer insights into BIG-ABAC's implementation, adaptability, and evaluation methodology.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Qiu J, Tian Z, Du C, Su S, Zuo Q, Fang B. A survey on access control in the age of internet of things. IEEE Internet of Things J. 2020;7(7):5934–53. doi:10.1109/JIOT.2020.2969326.
2. National Institute of Standards and Technology (NIST). Guide to Attribute Based Access Control (ABAC). NIST special publication 800-162. 2014 [cited 2025 Feb 27]. Available from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf.
3. Chenthara S, Ahmed K, Whittaker F, Mohamed A, Aamir F, Karim MA, et al. Privacy-preserving data sharing using multi-layer access control model in electronic health records. ICST Trans Scalable Inf Syst. 2018;6(22):159356. doi:10.4108/eai.13-7-2018.159356.
4. Longstaff JJ, Noble J. Attribute based access control for big data by query modification. In: 2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService); 2016; Oxford, UK. p. 58–65. doi:10.1109/BigDataService.2016.35.
5. Yang K, Zhao L, Wang T, Huang M, Liu X, Chen J, et al. Research on ABAC access control based on big data platform. J Cyber Secur. 2021;3(4):187–99. doi:10.32604/jcs.2021.026735.
6. Pal S, Kumar R, Gupta N, Singh P, Patel P, Sharma M, et al. Policy-based access control for internet of things healthcare. J Netw Comput Appl. 2019;139(3):57–74. doi:10.1016/j.jnca.2019.04.013.
7. Psarra E, Christodoulakis M, Koutroumpouchos N, Ioannidis S, Petroulakis N, Kapetanakis I, et al. Accessing electronic health records in critical incidents using context-aware attribute-based access control. Intell Decis Technol. 2021;15(4):667–79. doi:10.3233/IDT-210214.
8. Butpheng C, Ouda A, Venayagamoorthy K, Balakrishnan G, Zungeru AM, Gupta P, et al. Security and privacy in internet of things-cloud e-health systems: a comprehensive review. Symmetry. 2020;12(7):1191. doi:10.3390/sym12071191.

9. Salehi Shahraki A, Taghipour H, Mohammadi M, Khatibi B, Norouzian M, Sadeghi H, et al. A dynamic access control policy model for sharing of healthcare data in multiple domains. In: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE); 2019; Rotorua, New Zealand. p. 618–25. doi:10.1109/TrustCom/BigDataSE.2019.00088.
10. Walid R, Joshi KP, Elluri L. Secure and privacy-compliant data sharing: an essential framework for healthcare organizations. In: Giri D, Vaidya J, Ponnusamy S, Lin Z, Joshi KP, Yegnanarayanan V, editors. Proceedings of the International Conference on Medical Computing. Singapore: Springer Nature Singapore; 2024. p. 15–26. doi:10.1007/978-981-97-2066-8_2.
11. de Oliveira MT, Costa J, Fernandez J, Almeida R, Moreira C, Silva P, et al. AC-ABAC: attribute-based access control for electronic medical records during acute care. Expert Syst Appl. 2023;213(4):119271. doi:10.1016/j.eswa.2022.119271.
12. Cobrado UN, Sharief S, Regahal NG, Zepka E, Mamauag MB, Velasco LC, et al. Access control solutions in electronic health record systems: a systematic review. SSRN 2023 [Internet]. [cited 2025 Feb 27]. Available from: https://ssrn.com/abstract=4819213.
13. Tall AM, Zou CC. A framework for attribute-based access control in processing big data with multiple sensitivities. Appl Sci. 2023;13(2):1183. doi:10.3390/app13021183.
14. Kreps J, Narkhede N, Rao J. Kafka: a distributed messaging system for log processing. Linkedin engineering blog [Internet]. 2011. [cited 2025 Feb 27]. Available from: https://kafka.apache.org/.
15. Apache Software Foundation. Apache ranger: data security for hadoop ecosystems. 2017. [Internet]. [cited 2025 Feb 27]. Available from: https://ranger.apache.org/.
16. MongoDB Inc. MongoDB documentation. 2024. [Internet]. [cited 2025 Feb 27]. Available from: https://www.mongodb.com/docs/.
17. Baccouri SBIG-ABAC. Attribute-based access control framework for dynamic environments. 2024 [Internet]. [cited 2025 Feb 27]. Available from: https://github.com/BaccouriSondes/BIG-ABAC.
18. Wang J, Zhu M, Li M, Sun Y. An access control method against unauthorized and noncompliant behaviors of real-time data in industrial internet of things. IEEE Internet Things J. 2024;11(1):708–27. doi:10.1109/JIOT.2023.3285992.