DOI: 10.2478/ias-2026-0023 • JIAS • 21(6) • 2026 • 468–500

Article

Zero-Trust Architecture Applied to a Banking System

Maria Penelova

Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria; i_n_f@abv.bg

Abstract: Regulations such as NIS 2 require systems where data protection is crucial to implement security standards. The critical controls within the AC family of the NIST SP 800-53 rev. 5 and NIST SP 800-207 standard are satisfied only by access control models designed as Zero-Trust Architectures (ZTA). The paper proposes an innovative access control model AI-Trust-HRABAC, which incorporates context-based dynamic trust. The Hybrid Role and Attribute Based Access Control (HRABAC) model was selected as its technological core. By combining flexible properties with contextual parameters (such as time, location, and threat level), the proposed model achieves finer-grained permission management aligned with the Zero-Trust concept. Furthermore, a two-tier zero-trust architecture ZTA-HRABAC is proposed, combining a web-based layer and a blockchain layer. Since the former utilizes the AI-Trust-HRABAC model and the latter employs HRABAC, they share a common core. This enables them to operate harmoniously while maintaining system security. Here, the immutability and tamper-resistance properties of blockchain are utilized. Data integrity preservation is mathematically proven using invariants and evaluation under threat scenarios. The proposed framework is compared with other zero-trust architectures, and its practical feasibility is validated within a banking environment.

Keywords: Authorization, Blockchain, Dynamic Trust Assessment, Hybrid Role and Attribute Access Control, Zero-Trust Architecture, Unsupervised Learning

1. Introduction

Currently, the importance of the zero-trust principle is growing. It dictates that no subject should be trusted by default; instead, everything must be continuously verified. Furthermore, data security is shifting from network perimeter protection to identity management and access control [1]. Zero-Trust Architecture (ZTA) standards have been developed by NIST [2,3]. These standards introduce several specific architectural requirements. First, the level of trust [4] in the current context must be calculated dynamically. Second, system functionalities must be separated into appropriate layers. Third, user actions must be recorded in logs. When a user exhibits atypical behavior [5], a low trust score is assigned. Then, Multi-Factor Authentication (MFA) is enforced. Users are given rights, according to the principle of least privilege. Important data must be encrypted or hashed when there is a risk of disclosure. Directives such as NIS 2 [6] mandate compliance with standards in systems where data security is crucial. While the financial sector is primarily governed by the Digital Operational Resilience Act (DORA) [7] as *lex specialis*, compliance with broader frameworks, such as NIS 2, is essential to ensure the universal structural resilience of the proposed architecture.

In addition to compliance with technical security standards, modern banking systems must simultaneously balance the compliance with legal privacy regulations such as General Data Protection Regulation (GDPR) [8]. Therefore, personal data protection is extremely important.

This article proposes an access control model AI-Trust-HRABAC, by extending HRABAC with a Trust Evaluating Function (TEF). Then, it is integrated into a ZTA. The proposed ZTA-HRABAC architecture leverages a decoupled multi-layered structure, combining dynamic off-chain contextual filtering with immutable on-chain policy verification to guarantee end-to-end system integrity. The blockchain layer

© 2026 Maria Penelova, published by Cerebration Science Publishing Co., Limited. BY-NC-SA This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0

deliberately abstains from evaluating the dynamic environmental context directly on-chain. This architectural constraint is strictly enforced to prevent non-deterministic transaction execution, which would inherently trigger consensus failures and prohibitive computational costs (gas fees) within the distributed registry. Instead, the architecture delegates the resource-intensive, context-aware dynamic filtering entirely to the off-chain web layer via the AI-Trust-HRABAC model. Once the off-chain validation succeeds and any necessary MFA challenges are resolved, the verified access decision is converted into a standard blockchain transaction data, which is collaboratively signed off-chain via a 2-of-2 Threshold ML-DSA (t-ML-DSA) Multi-Party Computation (MPC) protocol [9–14]. Following the principles of post-quantum oblivious pseudorandom functions [14], this quantum-resistant cryptographic setup ensures that a valid Module-Lattice Digital Signature Algorithm (ML-DSA) payload can only be generated through collaborative interaction between the client device and the server infrastructure, without either party disclosing their private key shares s_{client} or s_{server}. Consequently, the smart contract on the distributed ledger executes deterministic ML-DSA signature verification, restricting its on-chain computational scope to constant-complexity $O(1)$ identity mapping checks, thus preserving processing finality against both classical and quantum adversarial patterns.

Consequently, the smart contract on the ledger restricts its scope to verifying the deterministic, constant-complexity $O(1)$ privilege and identity mappings based on this pre-validated, t-ML-DSA-signed off-chain context. This strategic separation maintains total system integrity without sacrificing ledger processing finality. In the web-based layer, there are different components of ZTA. Policy Management Point (PMP) manages users' accounts and access rights. Policy Decision Point (PDP) dynamically evaluates trust score and access privileges to grant or deny access, supported by environmental and privilege data from the Policy Information Point (PIP). Policy Enforcement Point (PEP) regulates access in the system based on the decision taken in PDP. The blockchain layer contains the successful access request logs. The proposed ZTA combines simplified identity management and fine-grained control of AI-Trust-HRABAC/ HRABAC and the tamper-resistant nature of the immutable blockchain. The proposed ZTA-HRABAC architecture meets the requirements of standards and regulations. Rather than evaluating a single software implementation, this work establishes the structural and conceptual paradigms necessary to enforce Zero-Trust principles through decoupled web and distributed ledger layers.

While the financial sector is utilized as a primary validator for the structural feasibility of the proposed framework, the ZTA-HRABAC architecture is designed as a domain-agnostic paradigm. Its decoupled layers and policy-driven engine are universally applicable to any critical infrastructure mandated by NIS 2, including e-government, e-health, and smart industrial IoT ecosystems.

The core architectural contribution lies in establishing cross-layer semantic compatibility between two distinct execution environments via a unified technological core (HRABAC). While contemporary Zero-Trust frameworks are constrained by computational and mathematical limitations that prevent them from operating within distributed ledgers, the proposed architecture successfully bridges decoupled off-chain and on-chain layers.

The rest of the paper is structured as follows. Section 2 presents the related work on ZTA. Section 3 proposes the access control model AI-Trust-HRABAC and the ZTA-HRABAC architecture. Section 4 compares the proposed ZTA with other alternative architectures. Section 5 describes the implementation of the proposed architecture. Section 6 analyzes the security of the proposed architecture. Section 7 discusses the compliance of ZTA-HRABAC to the requirements of standards and regulations. Section 8 provides conclusions and directions of future research and development.

2. Related Work

Recently, scientific articles have been published on ZTA [15, 16]. Authentication and authorization in ZTA are discussed and directions for future research are provided. Migrating legacy systems to zero-trust is cited as a challenge. [17] proposes a methodology for implementing risk-adaptive access control in ZTA. Reviews depict the current state and challenges for ZTA [18–22] analyze ZTA as a security framework and how it impacts data protection in organizations. [23] uses trust score and attributes for access control in the Internet of Vehicles. [24] provides an example of a ZTA implementation. Several access control models designed according to the ZTA principle are discussed below.

Dynamic and Semantic-Aware Access Control (DSAAC) model [25] was published in 2020. It is identity-based. DSAAC is developed assuming a workflow process in a multi-datacenter environment. User access to objects is controlled by evaluating workflow sequence violations and semantic constraints. In DSAAC, an object request includes attributes and information about previous behavior. By assessing the risk of each task

in the workflow, the model can deny access or alert administrators to irregularities. Administrators can explore decisions to grant or deny access, edit the sequence template library, and update the sequence anomaly detection module. DSAAC is suitable for dynamic access control in resource-rich environments.

AI-Augmented Role-Based and Attribute-Based Access Control (AIARBABAC) [26] was proposed in 2025. The model is designed for cloud technologies and, in particular, Microsoft Azure. Implementations of machine learning algorithms are connected to cloud software services. Several scenarios for protecting against unauthorized disclosure of data in the system are tested and covered by the model. Data confidentiality is preserved after implementing detailed access control.

Trust-Based Score Access Control Model (TSBACM) [27] was proposed in 2025. The model provides real-time policy verification and modification. Access decisions are context-based. The trustworthiness (level of trust) of users and devices is evaluated, depending on the evaluations of four factors: the behavior of the respective user, the device used, the network state, and the threat history. The degree of trust can be adapted to the application domain by changing the coefficients in front of the factor evaluations.

A Comprehensive Access Control Model (CACM) [28] was published in 2025. The model employs trust scores, roles and attributes. It supports the least privilege principle and applies continuous monitoring. The history of user behavior is stored in activity logs. They are analyzed via N-median outlier detection method to compute trust score.

DR-TBAC [29], proposed in 2025, is designed for cloud technologies according to the zero-trust principle. It evaluates user trustworthiness and achieves flexibility by dynamically updating trust scores through dynamic rules. The model works with the machine learning algorithm Deep Q-Network (DQN), to analyze user's behavior.

HRABAC [30, 31] provides flexible, dynamic and fine-grained authorization with complexity $O(1)$. First, the role of the subject is checked for the requested permission. Then, if it is necessary, a policy-function evaluates attributes. There is additional protection via isActive account status attribute. Initially, the model was successfully deployed within traditional information systems to provide fine-grained and flexible permission management, and subsequently, its concept was adapted and extended to distributed ledger technologies (blockchain environments). The present study continues and advances this line of research by expanding the model's theoretical framework and fully integrating it into a comprehensive two-tier ZTA. Consequently, this approach yields an architecture that successfully enforces the critical security controls mandated by modern industry standards [32].

3. The Proposed Model and Architecture

Consider a banking system that stores the history of financial transactions. It has two layers: a web-based layer and a blockchain layer. The web-based layer protects the personal data of the bank's customers and ensures fast performance. According to the NIST 800-207 standard, this layer must filter requests based on context and calculate a trust score. This way the system is protected from identity theft. Therefore, a function for assessing the trust of a subject in the system must be added to the HRABAC model. Thus, AI-Trust-HRABAC is proposed. Blockchain technology is used in the banking system due to its inherent immutability and tamper-resistance. In the blockchain layer, successful access requests and transaction data are stored. The blockchain layer applies HRABAC. Therefore, a new ZTA-HRABAC architecture is proposed by sequentially applying both the AI-Trust-HRABAC and HRABAC models. They work one after another in a similar way, so that the proposed architecture achieves a high level of security.

The following definitions extend the HRABAC model to AI-Trust-HRABAC.

Definition 1. Let $C = \{cond_1, cond_2, \ldots, cond_n\}$ be a set of conditions.

Definition 2. Let $eval: C \rightarrow [0, 50]$ be a function that evaluates every condition $cond_i$ where $i = 1, 2, \ldots, n$ with an integer value between 0 and 50.

Based on the architectural requirements established in Definition 2, the dynamic evaluation and translation of contextual telemetric drifts into discrete penalty scores are mathematically formalized through an adaptive distance function. This formulation utilizes a modified Z-score normalization to measure statistical deviations, incorporating a regularization coefficient ε to prevent singularities, and a weighting matrix γ_i tailored to the specific NIST ranges [33]. For each incoming access request, the localized penalty weight $eval(cond_i)$ associated with the environmental metric i (where $i \in [1, \ldots, n]$) is bounded within a strict closed interval of [0, 50] to eliminate extreme anomalies, derived using the following algebraic mapping (Equation (1)):

$$eval(cond_i) = \text{int}\left(\max\left(0, \min\left(50, \left|\frac{cond_i - \mu_i}{\sigma_i + \varepsilon}\right| \cdot \gamma_i\right)\right)\right) \quad (1)$$

To ensure full structural transparency and computational reproducibility, the operational parameters embedded within this mapping function are strictly defined as follows:

- $cond_i$ (Real-Time Ingestion Coordinate): Represents the continuous numerical scalar extracted from the subject's current access context parameter (e.g., the exact timestamp encoded as decimal hours, or the decimal conversion of the incoming network IPv4 address).
- μ_i (Historical Profile Mean): Denotes the mathematical expected value (mean) computed across the subject's sliding historical baseline log matrix. It represents the statistical center of gravity for the user's normal authorized behavior.
- σ_i (Standard Deviation Baseline): Captures the statistical dispersion and behavioral volatility of that specific user over time. A small σ_i indicates a highly rigid, predictable routine, whereas a larger σ_i implies naturally fluctuating operational patterns.
- ε (Regularization bound factor): A microscopic static hyperparameter initialized strictly at
- $\varepsilon = 10^{-6}$. It functions as a computational safeguard to eliminate zero-division boundary exceptions during algorithmic execution in scenarios where a user exhibits perfectly static behavior ($\sigma_i = 0$).
- γ_i (Empirical gamma risk coefficient): Represents the institutional scaling coefficient mapped directly onto the corporate security policy. It amplifies the raw dimensionless Z-score distance $\left|\frac{(x_i - \mu_i)}{\sigma_i}\right|$ into a proportional penalty currency. These coefficients are customized per infrastructure domain γ_{time} = 10, $\gamma_{location}$ = 15, γ_{device} = 12, $\gamma_{inactivity}$ = 8 to ensure that high-consequence vector displacements trigger rapid state mutations while smooth benign variations are safely absorbed.
- Int(..) and max(0, min(50, …)) operators: Enforce absolute integer data types and clamp the outputs to a strict closed boundary interval of [0, 50] points, preventing mathematically unbounded anomalies from corrupting downstream ledger consensus or disrupting the rigid access-matrix schema.

Definition 3. Let $F: Subjects \rightarrow [0, 100]$ be a Trust Evaluating Function (TEF). It assesses with a trust score how trusted a given subject is. The evaluation is based on environmental variables such as working hours and device identifiers. The subject's location is also assessed (Equations (2) and (3)).

$$T_{CURRENT} = TEF(subject) \quad (2)$$

$$T_{CURRENT} = max(0, T_{INIT} - \sum_{i=1}^{n} eval(cond_i)) \quad (3)$$

The following trust variables are used in the model:

- T_{INIT} = 100 - the trust score before being evaluated with TEF (initial value of trust score);
- $T_{CURRENT}$ is the current value of the trust score, applied after trust evaluation with TEF;
- T_{SECURE} – the secure threshold of trust score; for example, let its value be 80;
- $T_{TOLERANCE}$ – the tolerance threshold, under which the account is blocked after specific timeout.

Immediately after the subject submits the access request, the trust score has a value T_{INIT}. After that, TEF is applied, and the obtained trust score is $T_{CURRENT}$. There are two threshold values T_{SECURE} and $T_{TOLERANCE}$. The following cases are possible (Equations (4), (5) and (6)):

$$T_{CURRENT} \geq T_{SECURE} \quad (4)$$

$$T_{TOLERANCE} < T_{CURRENT} < T_{SECURE} \quad (5)$$

$$T_{CURRENT} \leq T_{TOLERANCE} \quad (6)$$

Definition 4. Let $SessionTime(T_{CURRENT}) = st$ be a function where st is period. It can have the

following values:

- $st = st_{NORMAL}$ when (4) or (5) holds;
- $st = st_{LIMITED}$ when (6) is satisfied.

Definition 5. *Let* $States = \{S_{ACTIVE}, S_{MFA1}, S_{MFA2}, S_{LIMITED}, S_{LOCKED}, S_{BLOCKED}\}$ be a set of states where:

- S_{ACTIVE} – The subject is active and has a normal session period. This is valid when (4) holds.
- S_{MFA1} – The subject is presented with the first MFA challenge. This is valid when (5) or (6) are satisfied.
- S_{MFA2} – The subject is presented with the second MFA challenge when he/she does not complete S_{MFA1} successfully. This is valid when (5) or (6) are satisfied.
- $S_{LIMITED}$ - The subject is still active and has limited session time due to insufficient trust score. This is valid when (6) holds.
- S_{LOCKED}– The subject is locked after passing the session timeout in case (6).
- $S_{BLOCKED}$ – The subject is blocked after making an error in the second MFA.

Definition 6. *Let* $Events = \{E_{SUCCESS1}, E_{SUCCESS2,} E_{FAIL1}, E_{FAIL2}, E_{END}\}$ be a set of events where:

- $E_{SUCCESS1}$– The subject completes the first MFA challenge successfully.
- $E_{SUCCESS2}$ – The subject completes the second MFA challenge successfully.
- E_{FAIL1} – The subject fails the MFA challenge for the first time.
- E_{FAIL2} – The subject fails the MFA challenge for the second time.
- E_{END} – The limited session time is over and the account is locked.

The system maintains an internal counter for sequential authentication failures within the MFA states (between S_{MFA1} and S_{MFA2}) to sequentially trigger an event E_{FAIL1} on the first validation failure and an event E_{FAIL2} strictly upon the second consecutive failure.

Definition 7. *Let* $\delta(Subject, S, E, T_{CURRENT}) \rightarrow States$ be a transition function where $Subject \in Subjects$, $S \in States$ and $E \in Events$. There are the following transition rules:

- On successful MFA: If the subject is in the verification state and (5) is valid, they return to the active state (Equation (7))

$$\delta(Subject, S_{MFA1}, E_{SUCCESS1}, T_{CURRENT}) = S_{ACTIVE} \quad (7)$$

  when (4) is valid.
- On first mistake: If the subject makes a mistake once, and (5) or (6) is valid, the system triggers a second verification step, maintaining the subject in the checking state (Equation (8)):

$$\delta(Subject, S_{MFA1}, E_{FAIL1}, T_{CURRENT}) = S_{MFA2} \quad (8)$$

  when (5) or (6) is valid.
- On second error: If the subject makes an error for a second time, and (5) or (6) is valid, the system automatically changes the state to blocked (Equation (9)):

$$\delta(Subject, S_{MFA2}, E_{FAIL2}, T_{CURRENT}) = S_{BLOCKED} \quad (9)$$

  when (5) or (6) holds.
- On successful MFA when (5) is valid: If the subject is successful, the session duration is restricted (Equations (10) and (11)):

$$\delta(Subject, S_{MFA1}, E_{SUCCESS1}, T_{CURRENT}) = S_{LIMITED} \quad (10)$$

  or

$$\delta(Subject, S_{MFA2}, E_{SUCCESS2}, T_{CURRENT}) = S_{LIMITED} \quad (11)$$

  when (6) holds.
- After the limited session time is over, then the system is locked (Equation (12)):

$$\delta(Subject, S_{LIMITED}, E_{END}, T_{CURRENT}) = S_{LOCKED} \quad (12)$$

  when (6) is valid.

The *entire* process is explained below. When (4) is fulfilled the access request proceeds further. In case (5) holds, an MFA challenge is triggered. When (6) is satisfied, the subject is also challenged with the MFA, and the session is highly limited. The exact time after which the subject is logged out is configured in the system settings by an administrator.

To significantly reduce computational overhead during runtime authorization, a streamlined two-stage configuration for TEF is proposed. Initially, new user profiles are populated with baseline penalty weights pre-configured for their specific organizational role. Asynchronous background unsupervised learning models (e. g. Isolation Forest) [34] sequentially process the historical window of the last N (for example, 50) access logs to adaptively recalibrate and store these customized penalty weights directly within the user's database profile.

The core intellectual contribution of this work is the decoupled, dual-layer ZTA-HRABAC framework. Within this architecture, the TEF serves as a dynamic, context-aware entrance gateway that conditions the subject's parameters before they are processed by successive hybrid policy enforcement points. By computing real-time trust scores, the TEF enables the AI-Trust-HRABAC model to strictly comply with the core zero-trust requirements mandated by the NIST SP 800-207 standard.

The transaction lifecycle follows a strict pipeline designed to insulate the blockchain network from performance degradation while maintaining zero-trust continuity as shown in Figure 1.

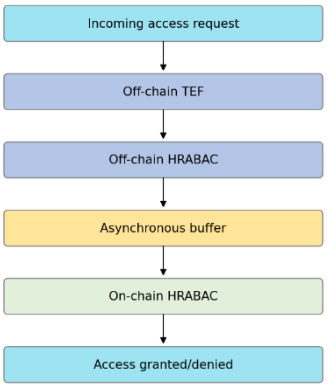


Figure 1. Processing an access request by ZTA-HRABAC architecture.

Step 1. Off-Chain TEF Grading: Upon ingestion of a live access request, the user's 4-dimensional environmental telemetry is intercepted by the off-chain TEF. The framework evaluates raw coordinate shifts against moving statistical profiles (μ_i, σ_i) to calculate dynamic penalty points, appending a real-time trust token to the active session state.

Step 2. Off-Chain HRABAC Evaluation: The tokenized session profile immediately targets the off-chain HRABAC layer. The web layer checks the dynamic attributes and active role mappings to perform a fast, non-blocking evaluation of the current operation.

Step 3. Asynchronous Message Buffer: Once the web layer passes the primary evaluation, the payload is committed to an isolated asynchronous message buffer (Redis queue). This decoupling layer acts as an essential load-throttling shield, preventing external transactional surges or intentional Denial-of-Service (DoS) floods from congesting downstream ledger components.

Step 4. On-Chain HRABAC Ledger Enforcement: The buffered transaction is popped and broadcasted to the Arbitrum ledger. A hardened Solidity smart contract executes a final, immutable HRABAC consensus verification check. By restricting its scope to a strict deterministic array match and ignoring external floating-point telemetry, the on-chain smart contract enforces absolute invariant integrity at a constant $O(1)$ complexity.

The proposed architecture structures the pipeline with an advanced off-chain TEF filter and a protective messaging buffer prior to ledger interaction. Thus, the ZTA-HRABAC framework successfully transitions from a passive, rule-scanning architecture to an active, highly resilient banking security perimeter.

To ensure the mathematical representation aligns with high-consequence banking threat environments, the additive subtraction mapping paradigm $T_{CURRENT} = T_{INIT} - \sum(eval(cond_i))$ is specifically designed to enforce a compounded risk acceleration perimeter. While conventional rule-based perimeters evaluate metrics via isolated branches, the AI-Trust-HRABAC model utilizes the additive layer to generate a rapid,

cascading depletion of the active trust score when multi-vector anomalies intersect as shown in Figure 2.

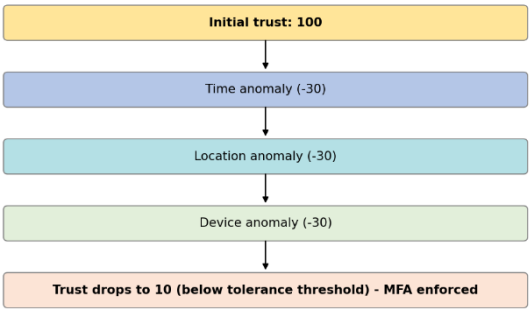


Figure 2. TEF grading and decreasing subject's trust score.

Because the underlying feature-specific weights are continuously calibrated by the multi-dimensional Isolation Forest covariance matrices, concurrent deviations across multiple telemetry spaces (such as a foreign IP injection initiated simultaneously during out-of-office hours) trigger a correlated expansion of the localized standard deviations.

As a mathematical consequence, the individual penalties scale up exponentially, causing their linear accumulation to immediately breach the structural $T_{TOLERANCE} = 20$ threshold constraint.

This interaction layer guarantees that the system completely bypasses authentication prompts during a coordinated attack sequence, instantly executing MFA challenge then switching the session into a critical operational mode $S_{LIMITED}$ and later executing a lockout via the formal state-machine transition invariants.

The historical window size N serves strictly as an illustrative baseline hyperparameter. In production environments, N can be dynamically scaled based on transaction volume without impacting runtime latency, as the training loop executes entirely within an isolated background queue.

Consequently, the real-time PEP avoids resource-intensive machine learning evaluations per request; instead, it executes a computationally lightweight linear deduction from the initial trust score ($T_{INIT} = 100$) utilizing the pre-calculated, individualized weights. This architectural shift maintains strict context-aware dynamic security boundaries while optimizing execution speed to a constant $O(1)$ performance footprint appropriate for production environments.

To maximize operational freshness, the unsupervised penalty weights are asynchronously recalculated in the background specifically upon a user's new activity in the system.

To guarantee rapid adaptability during an active session without imposing synchronous latency, the web layer pushes a lightweight calibration task into a task queue instantly upon the ingestion of each user event. The dedicated analytics server consumes these messages from the queue asynchronously. This architectural design ensures that while the main banking pipeline processes requests with $O(1)$ speed, the background Isolation Forest engine updates the behavioral profile within seconds during the active lifecycle of the session, rather than waiting for its finalization. The process of evaluating TEF in PDP is given with pseudocode.

To explain more specifically, TEF reduces the trust score of the subject according to the risk verification. In the absence of identity theft suspicion, the trust score defaults to 100. If a potential attack is detected, the trust score is reduced. For example, if the user does not use their authorized device, the trust score is reduced by deviceScore (for example, 30). If there is an access request outside of working hours, the trust score is reduced by workingTimeScore (for example, 30). If the user is connected from an IP that is not allowed, the trust score is reduced by locationScore (for example, 30). If the subject remains inactive for t consecutive seconds, the trust score is reduced by inactivityTimeScore (for example, 50). TEF can be customized according to the organization's needs. The parameters of the TEF are fully customizable and configured via the Policy Management Point (PMP) administrative interface based on user roles. For the Employee role, strict constraints for startTime and endTime are enforced to reflect institutional working hours. The deduction weights are dynamically mapped according to the bank's specific risk profile assessment. The trust score operational zones are shown in Figure 3.

Algorithm 1: Trust Evaluating Function

```
Function TEF(subject)
begin
trustScore := 100;
penaltySettings := subject.databaseProfile.customPenaltyWeights;
if (subject.hasWorkingHours) then
  localTime := convertToTimeZone(context.currentTime, subject.timeZone);
   if (localTime < subject.startTime || localTime > subject.endTime) then
        if (trustScore >= penaltySettings.workingTimeScore) then
           trustScore := trustScore - penaltySettings.workingTimeScore;
        else
           trustScore := 0;
        end
   end
end
if (context.ipAddress not in subject.allowedIPAddresses) then
    if (trustScore >= penaltySettings.locationScore) then
        trustScore := trustScore - penaltySettings.locationScore;
      else
        trustScore := 0;
      end
end
if (context.deviceIdentifier != subject.allowedDevice) then
   if (trustScore >= penaltySettings.deviceScore) then
      trustScore := trustScore - penaltySettings.deviceScore;
   else
      trustScore := 0;
   end
end
if (context.inactivityTime > config.acceptableInactivityPeriod) then
   if (trustScore >= penaltySettings.inactivityTimeScore) then
     trustScore := trustScore - penaltySettings.inactivityTimeScore;
  else
     trustScore := 0;
  end
end
if (trustScore <= config.toleranceTrustScore) then
   context.sessionTimeout := subject.role.minSessionTimeout;
else
   context.sessionTimeout := subject.role.maxSessionTimeout;
end
if (context.activityTime >= context.sessionTimeout) then
    context.session.destroy();
end
 context.currentTrustScore := trustScore;
end.
```

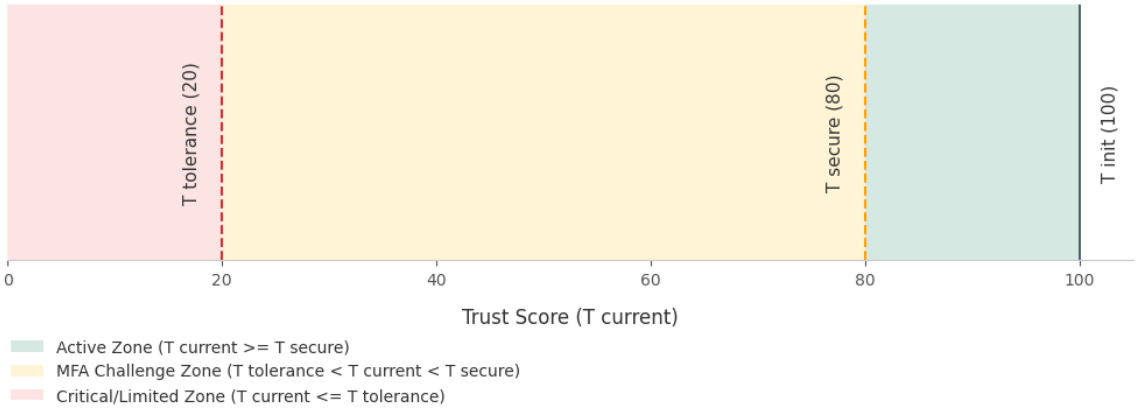


Figure 3. AI-Trust-HRABAC Trust Score Operational Zones, according to the example thresholds.

There is a very important clarification that in the pseudocode the reductions in the trust score are variables. In practice they are calculated according to the bank's specific risk profile.

The computational cost of the proposed TEF is estimated using Big O notation. The function processes a strictly predefined and fixed number of environmental parameters (n) - operating time, device identification, geographic location and inactivity period. The evaluation is based on sequential, independent conditional operators without iterative loops or recursive execution paths. Therefore, the execution time remains strictly bound and independent of the total number of subjects or transactions scaled in the banking system. Thus, the theoretical computational complexity of TEF is constant, denoted as $O(1)$. This mathematical bound justifies the negligible latency achieved.

Once the TEF evaluates the score and when (5) or (6) holds, the subject is required to complete a MFA challenge. If MFA is successful, T_{SECURE} is lowered and becomes equal to the subject's trust score for the current session.

To prevent unauthorized access, the system employs a protective mechanism similar to the Amazon Web Services Management Console [35]. If the subject fails the MFA challenge twice, their account is automatically blocked (the value of the attribute isActive is set to False).

If trust score is less than or equal to $T_{TOLERANCE}$ (e.g. 20), then session is limited to a role-specific minimum time of minSessionTimeout (e.g. 3 minutes).

The process of access control in the proposed AI-Trust-HRABAC is divided into separate steps. It is shown in Figure 4, where high trust level corresponds to (4), and low trust level – to (5) and (6).

Step 1. The subject submits an access request to a system resource. The system checks whether the subject's account is active.

Step 1.1 If it is active, then proceed with Step 2.

Step 1.2 If it is not active, then proceed with Step 6.

Step 2. TEF dynamically computes the current trust score based on environmental parameters.

Step 2.1. If the trust score is greater than or equal to T_{SECURE}, then proceed with Step 3.

Step 2.2. If the trust score is less than T_{SECURE}, then a MFA challenge is set to the subject.

Step 2.2.1. If there are two unsuccessful attempts by the subject to complete the MFA challenge, then their account is made inactive and blocked. Proceed with Step 6.

Step 2.2.2. If the MFA challenge is successfully completed, then proceed with Step 3.

Note. If after the evaluation the trust score is less than or equal to $T_{TOLERANCE}$, a critical mode is activated: the session is given a role-specific shortened timeout (e.g. 3 minutes). After its expiration the subject is logged out of the system.

Step 3. The role is checked whether it is active and the permission is assigned to the role. If the result is True, then proceed with Step 5; otherwise, proceed with Step 4.

Step 4. Policy function with attributes is evaluated. If the result is True, then proceed with Step 5; otherwise, proceed with Step 6.

Step 5. The access is granted.

Step 6. The access is denied.

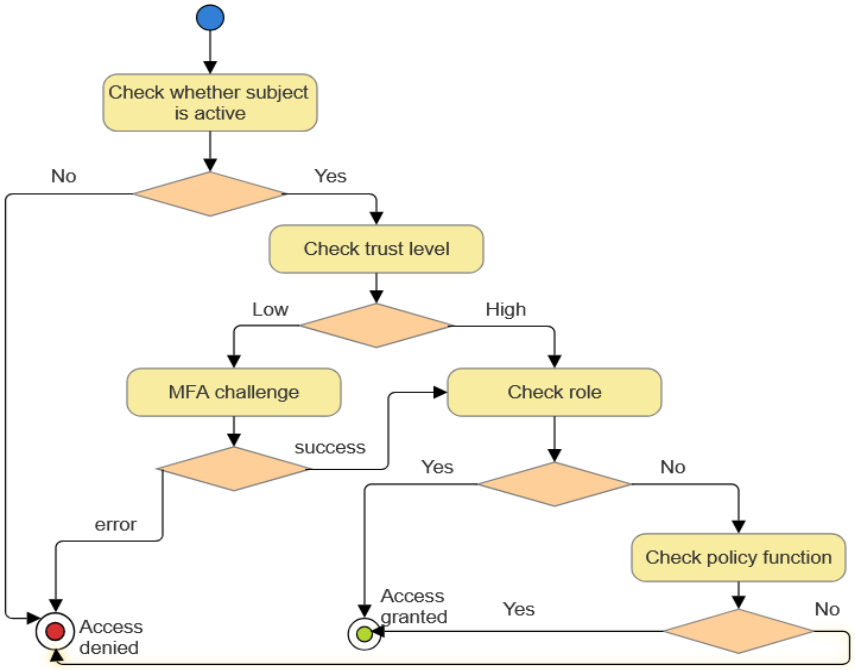


Figure 4. UML activity diagram of the model AI-Trust-HRABAC.

The UML class diagram of AI-Trust-HRABAC is shown in Figure 5.

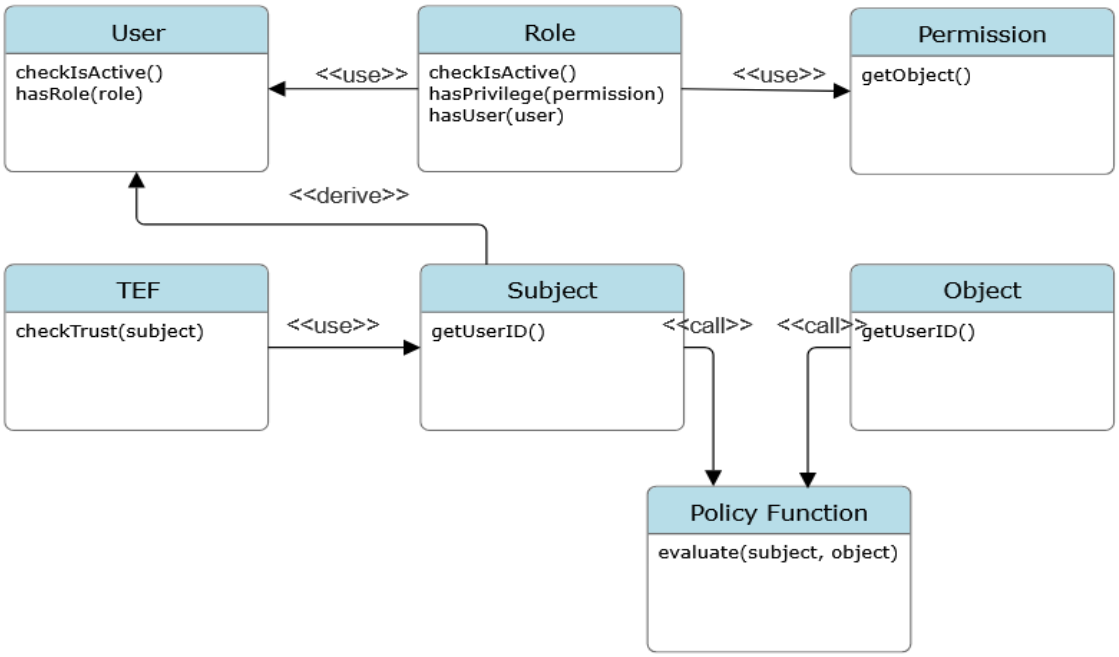


Figure 5. The UML class diagram of the AI-Trust-HRABAC model.

To ensure that the web-based layer cannot be bypassed, the proposed ZTA-HRABAC architecture leverages a 2-of-2 Threshold ML-DSA (t-ML-DSA) post-quantum protocol combined with Account Abstraction (ERC-4337) [36]. The private module-lattice key material is mathematically split into two independent polynomial key shares: a client-side vector share s_{client} stored securely within the subject's local hardware enclave, and a server-side vector share s_{server} managed by the bank's secure infrastructure.

When the t-ML-DSA protocol is launched, the user's device (client-side) receives and visualizes in a decentralized application (e.g., a hardware token or a secure enclave user interface) the final transaction parameters (amount and recipient) for independent verification before generating its share of the post-quantum cryptographic signature footprint.

When a subject requests access, the web layer evaluates the trust score via TEF. If the trust evaluation succeeds and any required MFA challenge is completed and HRABAC checks are performed, the subject's device and the banking server collaboratively execute an off-chain t-ML-DSA multi-party handshake. This interactive protocol, bound to lattice-based cryptography, generates a single, mathematically valid signature compliant with the NIST FIPS 204 standard without either party ever disclosing their respective polynomial key shares to the other. The resulting transaction is transmitted to the private blockchain layer, where the smart contract evaluates the HRABAC policy model on-chain. The blockchain ledger stores access permissions mapped to abstract, pseudonymized t-ML-DSA addresses rather than real-world identities, achieving strict privacy. The smart contract executes the HRABAC logic using localized database IDs mapped to these abstract quantum-resistant addresses.

The initial access decision is made in the web-based layer with AI-Trust-HRABAC. The immutable and tamper-proof nature of the blockchain layer is used to store authorization data and access request logs. The HRABAC model operates within this layer. The combination of these two layers, which forms the proposed ZTA-HRABAC architecture, guarantees a high level of security. Even if an attacker modifies the access privileges within the database, the blockchain layer continues to safeguard the data.

The process of access control in the proposed t-ML-DSA-powered ZTA-HRABAC architecture is divided into the following sequential steps:

Step 1. The subject initiates an access request to a system resource object via their local device. The web-based layer verifies whether the subject's account status attribute isActive is set to True.

Step 1.1 If it is True, proceed to Step 2;

Step 1.2 If it is False, proceed to Step 8.

Step 2. TEF dynamically computes the current trust score based on environmental parameters.

Step 2.1. If the trust score is equal to or greater than T_{SECURE}, proceed directly to Step 3.

Step 2.2. If the trust score falls below T_{SECURE}, an MFA challenge is triggered.

Step 2.2.1. If the subject fails the MFA challenge twice, the account is blocked (isActive is set to False), and the flow proceeds to Step 8.

Step 2.2.2. If the *MFA* challenge is successfully completed, proceed to Step 3.

Note. If after the evaluation the trust score is less than or equal to $T_{TOLERANCE}$, a critical mode is activated: the session is given a shortened timeout (e.g. 3 minutes). After its expiration the subject is automatically logged out of the system.

Step 3. The role is checked whether it is active and the permission is assigned to the role. If the result is True, then proceed with Step 5; otherwise, proceed with Step 4.

Step 4. Policy function with attributes is evaluated. If the result is True, then proceed with Step 5; otherwise, proceed with Step 8.

Step 5. Upon successful verification, the subject's device and the banking server execute the collaborative off-chain t-ML-DSA protocol. Using their respective key shares, they generate a joint cryptographic signature. The web layer instantly pushes this payload into a transaction queue and immediately releases the user session. Background workers sequentially consume payloads from the queue and broadcast them to the blockchain execution network using a rate-limiting mechanism. Proceed to Step 6.

Step 6. The smart contract on the blockchain layer executes the transaction and autonomously evaluates the on-chain HRABAC policy. It verifies if the abstract t-ML-DSA post-quantum address possesses the required role and active permissions for the requested object. If the result is True, proceed to Step 7; otherwise, proceed to Step 8.

Step 7. The smart contract executes the request.

Step 8. The access request is denied.

To prevent blockchain overload and handle flash-crowd transaction spikes during peak hours, the architecture incorporates a high-throughput transaction buffer layer. Instead of broadcasting the t-ML-DSA-signed payload directly to the ledger, the web tier pushes it as a non-blocking job into a queue. This layout decouples the user’s critical performance path from blockchain consensus time, locking the web-tier responsiveness to sub-millisecond execution speeds.

The main components of ZTA-HRABAC are listed as follows:

1) Policy Enforcement Point (PEP) – grants or denies access to a system resource for a particular subject, according to the taken decision or verifies subject's identity.
2) Policy Decision Point (PDP) – takes the access decision or proposes the MFA challenge to verify subject's identity.
3) Policy Information Point (PIP) - provides the necessary data for an access decision to be made. It consists of relational database tables.
4) Policy Management Point (PMP) – provides interfaces and functionalities for configuring the access rights of the users.
5) Blockchain layer – stores a log of allowed access requests. Taken in isolation, the ledger contains no personal data, as users are represented solely by localized database IDs which carry no inherent identity traits outside the system's registry. t-ML-DSA post-quantum addresses are persistently aggregated within the transaction roots of the blockchain headers. This architecture achieves privacy by design through Account Abstraction (ERC-4337), ensuring that transactions cannot be linked to physical individuals without access to the off-chain database.

The interaction of the components listed above is shown in Figure 6.

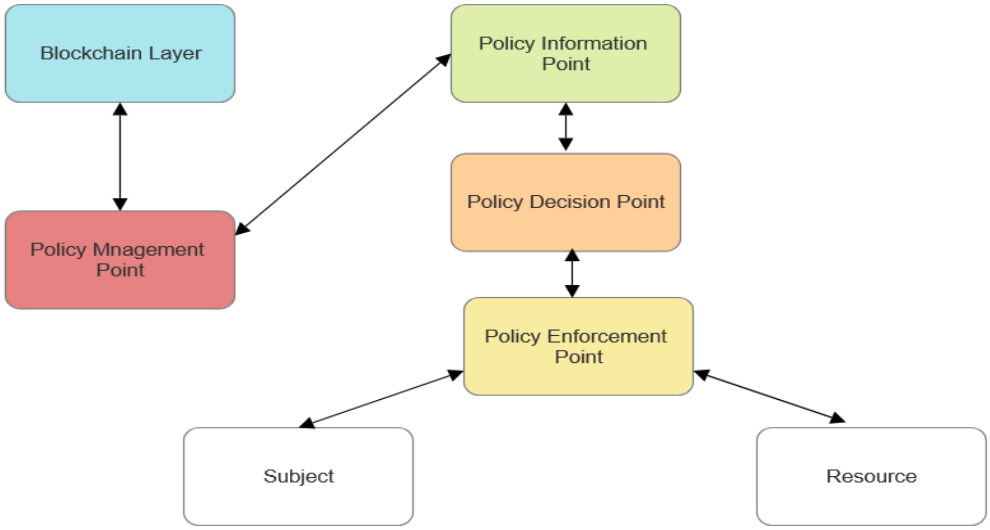


Figure 6. The components of the proposed architecture ZTA-HRABAC.

4. Comparison of the proposed architecture with other zero-trust architectures

The proposed ZTA-HRABAC architecture is compared with DR-TBAC, TSBACM, DSAAC, AIARBABAC and CACM on four parameters: granularity, protection of access rights, data integrity, additional blockchain layer and complexity.

Granularity. Granularity refers to the capability of an access control model to enforce fine-grained authorization checks. DSAAC, CACM and AIARBABAC provide authorization with a high level of granularity because they use policies with subject attributes and object attributes. ZTA-HRABAC evaluates roles, subject attributes, environmental variables, and object attributes, thereby supporting the strict enforcement of the principle of least privilege. TSBACM and DR-TBAC fail to incorporate object-level attributes, significantly restricting their authorization granularity.

Access rights protection. In ZTA-HRABAC, access rights are stored in a blockchain to protect against privilege-tampering attacks in the database. While the rest of the ZTA models rely solely on centralized relational databases - rendering them vulnerable to privilege tampering attack, ZTA-HRABAC introduces an immutable validation layer. By anchoring authorization policies and successful access history within an immutable blockchain ledger, the architecture guarantees tamper-proofing. The software bridge acts as an independent compliance tool. By continuously cross-referencing off-chain audit tables with on-chain cryptographic state events, it guarantees that any unauthorized manual alterations inside the relational

database are instantly detected, triggering an immediate alert to the system auditor.

Data Integrity (DI). In ZTA-HRABAC, there are checks for data integrity between the web-based layer and the blockchain layer. The rest of ZTA models do not support data integrity.

Additional blockchain layer (ABL). ZTA-HRABAC provides two independent layers of security: a web-based layer and a blockchain layer. Both layers perform similar access control verifications for double protection. The difference is that, in addition, the web-based layer filters requests based on the computed trust score. The blockchain layer must also provide data protection checks because it may be attacked as a separate system from the web-based layer. The structural limitations of existing zero-trust models inherently prevent them from integrating a distributed ledger layer for privilege protection. Models reliant on advanced computational methods, such as DR-TBAC (which uses Deep Q-Networks) and CACM (which employs N-median outlier detection), require continuous execution of floating-point mathematics and heavy statistical iterations over massive data arrays. Performing such operations directly on-chain via smart contracts is computationally prohibitive and causes processing latency. Context-centric models like TSBACM demand real-time coefficient updates that would induce severe state-synchronization bottlenecks on a ledger. On the other hand, workflow-driven frameworks like DSAAC and vendor-locked systems like AIARBABAC (designed for Microsoft Azure) depend on closed, centralized environments and external semantic states that cannot be natively ingested by a deterministic blockchain execution environment without introducing vulnerable oracle dependencies. Our proposed ZTA-HRABAC architecture overcomes these boundary limitations by utilizing a double decoupled approach. First, it delegates the execution-heavy behavioral risk assessment to an asynchronous off-chain pipeline, while restricting the live web-based PEP to deterministic environmental verification with strict $O(1)$ efficiency. Second, the successful access requests reach the blockchain layer where deterministic checks are performed again with $O(1)$ complexity. This strategic separation maintains a zero-error operational threshold without inheriting the latency or cost penalties that render prior models incompatible with distributed ledgers.

Complexity. The comprehensive structural taxonomy presented in Table 1 explicitly isolates the core algorithmic advantage of the proposed HRABAC ecosystem over contemporary literature. Existing context-adaptive frameworks natively degrade blockchain performance due to cross-layer execution bounds. For instance, DR-TBAC relies on Deep Q-Networks (DQN), while CACM utilizes N-median outlier detection. Forcing these floating-point operations directly on-chain via smart contracts generates polynomial or linear operational scaling complexities $O(N)$, where N is the count of parameters in the neural network or the count of records in the access log, respectively, resulting in prohibitive transaction costs and latency bottlenecks. To render an authorization decision, the AIARBABAC model must iterate through all N environmental attributes or historical user records. In more sophisticated deployments where deep statistical iterations or real-time behavioral correlations are applied, this linear complexity can inherently escalate to a quadratic or cubic footprint relative to the volume of analyzed data. For DSAAC, the count of processed tasks is denoted with N. TSBACM explores N contextual factors for a particular subject.

Table 1. Comparison of ZTA-HRABAC with Other ZTA.

Model/Param.	Granularity	Access rights protection	DI	ABL	Complexity
DR-TBAC	Medium	Only in relational database	No	No	$O(N)$
TSBACM	Low	Only in relational database	No	No	$O(N)$
DSAAC	High	Only in database	No	No	$O(N)$
AIARBABAC	High	Only in database	No	No	$O(N)$
CACM	High	Only in database	No	No	$O(N)$
ZTA-HRABAC	High	Double – database and blockchain	Yes	Yes	$\boldsymbol{O(1)}$

By contrast, the ZTA-HRABAC framework introduces a structural paradigm shift by strictly enforcing a dual-stage asynchronous pipeline. It cushions systemic load by decoupling live web event ingestion from heavy mathematical execution via a rate-limited task queue. This capability is completely absent in tightly-coupled frameworks like CACM or DR-TBAC. By encapsulating the progressive unsupervised penalty weight calibration entirely within asynchronous Python background processes, the live web-based evaluation can execute synchronously via lightweight, deterministic $O(1)$ checks using these pre-calibrated weights. Consequently, the primary web tier remains entirely unaffected by analytical computational spikes. The architecture fully decouples the on-chain layer, allowing the smart contract on the blockchain network to execute its verification asynchronously relative to the active user session. In addition, ZTA-HRABAC

completely isolates the contextual checks from the ledger. The smart contract on the blockchain execution network is restricted to asynchronous evaluation over a single pre-calculated, t-ML-DSA-signed transaction payload. This allows the distributed registry to run purely via instant, deterministic hash-map mappings, locking the on-chain computational cost to a strict constant complexity of $O(1)$. Thus, the proposed framework is the only architecture capable of deploying multi-dimensional behavioral AI defense without forfeiting the sub-millisecond execution speeds required by retail banking blockchains.

The comparison results summarized in Table 1 demonstrate that ZTA-HRABAC guarantees an advanced level of security and GDPR compliance.

To properly contextualize the operational baseline of the proposed architecture within the current state-of-the-art literature, a structural benchmarking review was executed. It must be noted that Table 1 is structured intentionally as a macro-architectural comparison matrix, designed to evaluate high-level systemic features, cross-layer topologies, and critical path execution complexity bounds.

This specific taxonomy does not evaluate the localized, micro-level statistical performance metrics of the underlying machine learning models, which inherently depend on specific runtime data distributions. Instead, Table 1 serves to establish the broad protocol-level differentiators that separate the dual-layer ZTA-HRABAC paradigm from conventional, single-tier access control deployments. The granular analytical classification quality of the trust scoring engine—specifically its accuracy, precision, recall, and F1-score profile—is isolated and evaluated independently through the empirical simulation frameworks detailed in Section 5.2.

The integration of 2-of-2 t-ML-DSA protocol elevates the access rights protection of the proposed ZTA-HRABAC model to high, matching or exceeding traditional centralized solutions without inheriting their vulnerabilities.

5. Implementation

Let there be four roles in the banking system: Customer, Employee, Manager, and Administrator. The Customer is authorized to create financial transactions and view only his/her own transaction history, as shown in Figure 7. The Employee can approve a transaction and see only the transactions he/she has approved. The use cases diagram for this role is shown in Figure 8. The Manager role has the right to see all transactions, which is depicted in Figure 9. The Administrator is responsible for setting and changing access rights and deploying applications. The corresponding use cases are shown in Figure 10.

Data processing stages in the banking system are shown in Figures 11, 12 and 13 for roles Customer, Employee and Administrator, respectively. What they all have in common is that information is first recorded in the database, then recorded on the blockchain. Updates are then made, if necessary, back in the web-based layer. The exact sequence of these interactions is detailed below in each corresponding diagram.

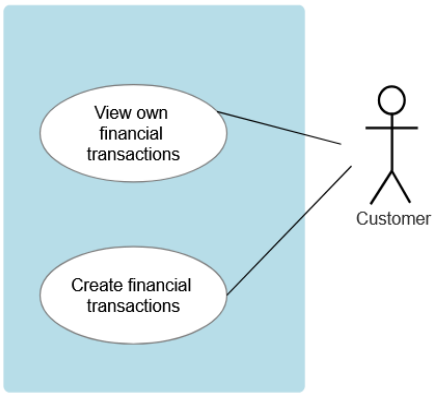


Figure 7. UML use case diagram for role Customer in the banking system.

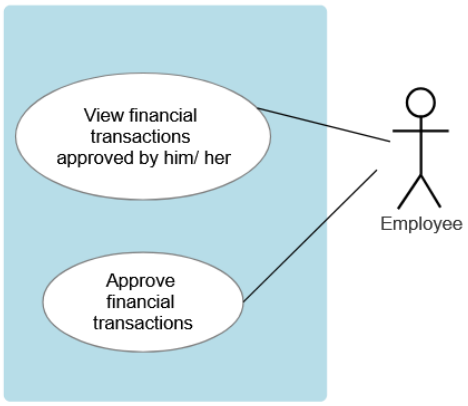


Figure 8. UML use case diagram for the role Employee in the banking system.

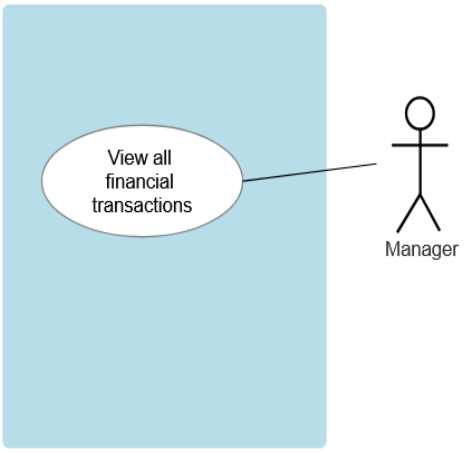


Figure 9. UML use case diagram for the role Manager in the banking system.

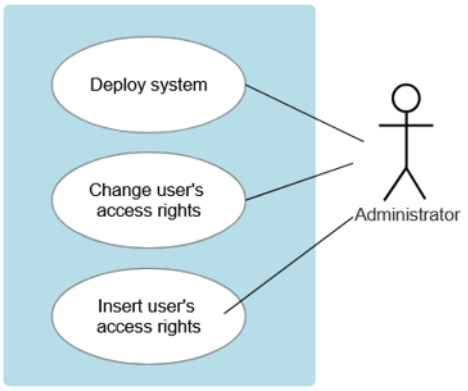


Figure 10. UML use case diagram for role Administrator in the banking system.

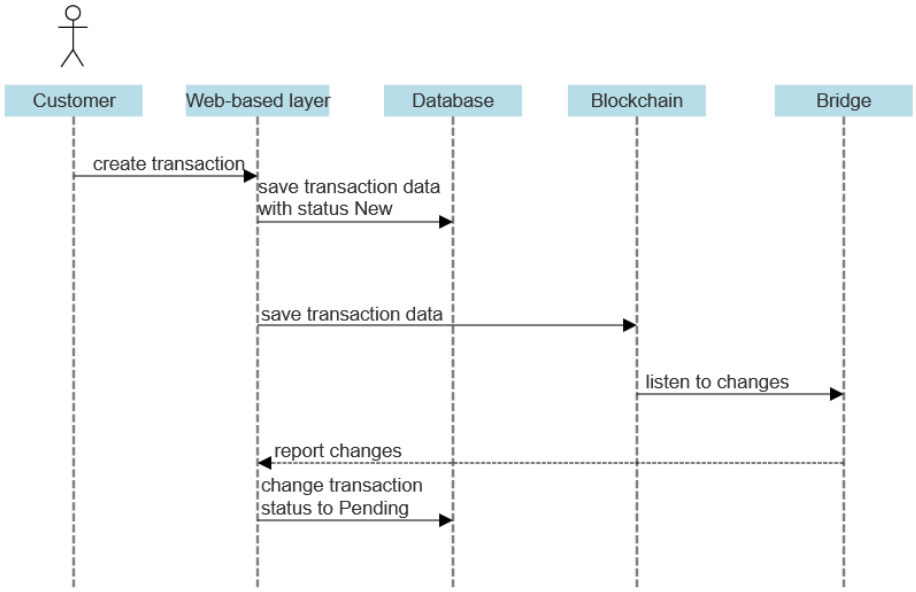


Figure 11. UML sequence diagram for role Customer in the banking system.

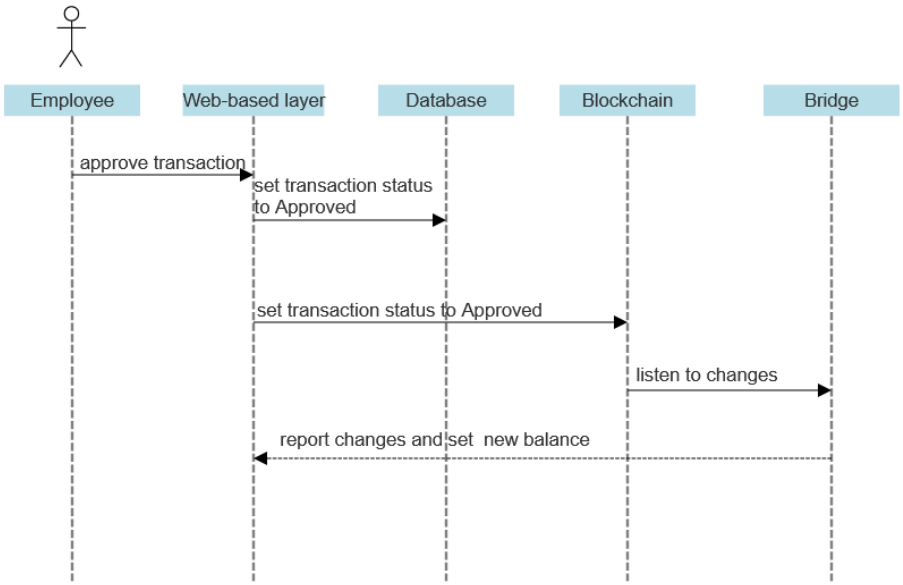


Figure 12. UML sequence diagram for the role Employee in the banking system.

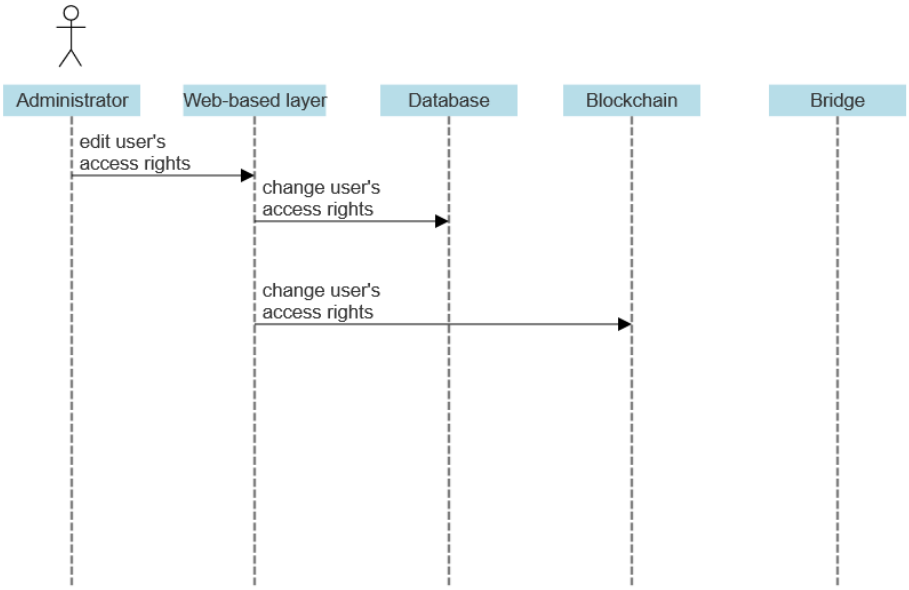


Figure 13. UML sequence diagram for the role Administrator in the banking system.

5.1. Technologies and Blockchain Performance

The proposed banking system is implemented using the following technology stack:

- Laravel [37] - the most widely used software framework [38] in the PHP language [39].
- MySQL [40] - the most popular relational database in practice.
- Laravel Livewire [41] - Integrated within the web tier to manage the real-time reactivity of the user interface and handle session state tracking dynamically. Instead of relying on traditional synchronous HTTP requests, Livewire maintains an asynchronous communication bridge with the backend. The active user session incorporates a dedicated inactivity counter. Upon any user interaction (e.g., clicks, keypresses, or navigation events), Livewire instantly transmits a lightweight, non-blocking payload to the server that resets this inactivity counter to zero. If the counter exceeds the defined threshold before a new interaction payload is registered, the state transitions are triggered instantly. This decoupled approach ensures precise, real-time tracking of user presence and seamlessly updates the environmental metrics required by TEF without causing full-page reloads or degrading application throughput.
- Solidity [42] - the programming language for blockchain.
- Arbitrum [43] – a popular platform compatible with the Ethereum Virtual Machine (EVM). By deploying Arbitrum on private nodes managed via a multi-node redundant consensus operated across an inter-bank consortium network (comprising independent banking nodes and financial regulators), the proposed architecture eliminates transaction costs (gas fees) and minimizes block latency. Operating as a standalone deployment among distinct institutional entities, network finality and consensus security are maintained autonomously by the consortium. This guarantees that even if a single bank's off-chain infrastructure is entirely subverted, the majority of the validator network will instantly reject inconsistent state transitions, thereby optimizing performance and resilience tailored for banking operations.
- Node.js [44] - JavaScript technology that runs on the server.
- Python [45] – implements machine learning algorithm [46] that analyzes user behavior by training on successful access request history. This analytical engine executes asynchronously out-of-band and entirely off the critical transaction pathway, ensuring that dynamic profile updates do not introduce runtime latency or degrade live retail banking throughput.
- Redis [47]– implemented as a high-throughput message broker to handle asynchronous task distribution between Laravel and the Python AI engine, while also serving as an off-chain transaction

queue buffering outgoing transactions before they reach the Arbitrum nodes.

The architectural components are realized through these technologies as follows:

- PEP is developed as a Laravel middleware that filters access requests and protects system resources [48];
- PDP is represented by functions provided by Laravel models that are called by the middleware to make an access decision [48].
- PMP consists of Laravel interfaces and functionality designed to manage users, roles, and access rights [48].
- PIP consists of MySQL database tables that store the necessary information to make an access decision.
- The blockchain layer is a Solidity smart contract [49] deployed on Arbitrum. To further validate code-level security boundaries, the underlying Solidity smart contract was locked to a fixed pragma version (0.8.28) and subjected to static application vulnerability scanning using the Slither [50] static analysis framework. The automated audit successfully finalized, reporting zero critical, medium, or low-severity security vulnerabilities (such as reentrancy vectors, integer flaws, or access control exploits), thus empirically confirming that the codebase functions as a structurally secure and optimized academic proof of concept;
- Node.js with Web Sockets [51] software bridge is deployed strictly to manage cross-layer audit synchronization and real-time ledger monitoring. Utilizing an event-driven, non-blocking architecture, the Node.js program listens to successfully confirmed blockchain authorization events on the Arbitrum network via persistent web sockets. Upon capturing an on-chain event, the bridge instantly updates an immutable, read-only monitoring dashboard for bank managers. Simultaneously, it triggers a background Laravel job to securely synchronize the validated transaction history back into the off-chain MySQL audit tables. Denied requests are logged directly by the web layer, allowing the Node.js bridge to optimize blockchain throughput by handling only verified ledger state events.
- The analytical core of the proposed architecture is driven by a dedicated, high-performance Python analytics engine. This engine is solely responsible for parsing multi-dimensional user telemetry data and running the unsupervised machine learning model to establish dynamic behavioral profiles.
- A Web-to-AI Pipeline: The behavioral profiling infrastructure isolates heavy analytical computations from the main application lifecycle through a decoupled Laravel-to-Python pipeline. Instantly upon the ingestion of each user action, the Laravel PEP captures the raw context and pushes a lightweight telemetry payload into a distributed task queue (powered by Redis).
- Note. Operating as an independent background worker, the Python analytics engine sequentially consumes these payloads to prevent server resource starvation during peak hours. The engine executes the Isolation Forest algorithm over a sliding historical window of N (e. g. 50) logs, dynamically calibrating the personalized penalty weights. These updated risk parameters are returned as a response and written directly back to the database. This ensures that while Python adapts the security baselines asynchronously, Laravel enforces access decisions synchronously with deterministic $O(1)$ computational speed.
- A Ledger-Bound Asynchronous Pipeline: Upon pushing the transaction to Redis, a dedicated Laravel background worker isolates the blockchain broadcast from the HTTP request-response cycle. This worker enforces traffic throttling and fault tolerance. If an Arbitrum node triggers a timeout or network exception, the job retries within Redis without impacting retail banking responsiveness.

To validate the computational efficiency of the blockchain-layer authorization logic, the smart contract is tested using the framework Hardhat [52]. The local in-memory EVM execution metrics indicate a processing time of 0.659 ms for transaction creation and 1.628 ms for transaction approval under normal conditions. In the simulated attack scenarios, unauthorized requests triggered contract exceptions, and the transactions were successfully reverted within 1.957 ms. The testing environment was configured using a localized single-node execution setup to capture the baseline algorithmic efficiency of the Solidity implementation. While production banking infrastructures deploy distributed private Arbitrum clusters across enterprise-grade cloud servers, these localized metrics serve as a theoretical lower-bound performance indicator, proving that the underlying smart contract authorization limits processing overhead to less than 2 milliseconds per operational lifecycle.

The results shown in Figure 14 demonstrate that the underlying Solidity implementation of the HRABAC model is highly optimized and introduces negligible computational overhead.

```
  BankSystemHRABAC - Access Control Tests
    Scenario 1: Customer creates transaction
customer creates transaction: 0.659ms
      ✓ Should allow an active customer to successfully create a pending transfer
      ✓ Should reject transaction if amount is 0
    Scenario 2: User without a role tries to approve a transaction
malicious user rejected to approve transaction: 1.957ms
      ✓ Should revert with UnauthorizedAccess if a malicious user attempts to approve
    Scenario 3: Deactivated employee tries to approve a transaction
inactive employee rejected to approve transaction: 1.757ms
      ✓ Should revert with UnauthorizedAccess if the employee is marked as inactive
    Scenario 4: Data isolation between customers
customer 2 rejected to read customer 1 data: 2.638ms
      ✓ Should prevent a customer from reading another customer's transfer details
employee approves transaction: 1.628ms
employee sees transaction details: 0.798ms
manager sees transaction details: 0.716ms
      ✓ Should allow the correct customer, employee, or manager to view the details
```

Figure 14. Smart contract execution and authorization metrics in Hardhat network.

The constant complexity of HRABAC is proved by an experiment as shown in Figure 15.

```
 BankSystemHRABAC - Absolute Gas Benchmark O(1)

Load: 1 records in the mappings
  - Gas used for registerCustomer  : 71633
  - Gas used for createTransfer    : 79607
  - Gas estimated for getTransferDetails: 32263
   ✓ Should measure identical gas usage for a state size of: 1

Load: 100 records in the mappings
  - Gas used for registerCustomer  : 71633
  - Gas used for createTransfer    : 79607
  - Gas estimated for getTransferDetails: 32263
   ✓ Should measure identical gas usage for a state size of: 100 (3

Load: 1000 records in the mappings
  - Gas used for registerCustomer  : 71633
  - Gas used for createTransfer    : 79607
  - Gas estimated for getTransferDetails: 32263
   ✓ Should measure identical gas usage for a state size of: 1000 (

Load: 10000 records in the mappings
  - Gas used for registerCustomer  : 71633
  - Gas used for createTransfer    : 79607
  - Gas estimated for getTransferDetails: 32263
   ✓ Should measure identical gas usage for a state size of: 10000
```

Figure 15. Empirical proof of constant complexity in the bank system.

It must be explicitly noted that these empirical metrics evaluate low-concurrency banking workflows within a controlled, localized single-node execution setup to capture baseline algorithmic efficiency. They do not simulate high-throughput real-world stress testing under massive multi-user concurrent loads or flash-crowd network spikes.

To guarantee systemic resilience against such high-concurrency stressors in production environments, the architecture relies on its decoupled multi-layered design. Instead of executing synchronous block propagation per request, the web-tier PEP immediately offloads the signed t-ML-DSA payloads into a high-throughput asynchronous Redis transaction queue. This operational layout completely insulates the retail banking web layer, locking user response speeds to sub-millisecond execution regardless of current blockchain consensus

latency or queue backlogs. Consequently, while this paper establishes the foundational lower-bound performance metrics, conducting large-scale distributed Wide-Area Network (WAN) stress testing across multi-region validator clusters remains an essential next step for enterprise-level industrial deployment.

However, since the proposed ZTA-HRABAC architecture enforces a strict decoupling strategy — delegating the context-heavy AI evaluation to an asynchronous off-chain pipeline — the on-chain smart contract is bound to a deterministic constant complexity of $O(1)$. Consequently, in a geographically distributed production deployment, the transaction finality will depend solely on the underlying consortium network's infrastructure throughput, without inheriting cumulative software execution overhead .

It is critical to note that within this decoupled architectural design, the live web-based application pipeline operates entirely independently of the distributed ledger's consensus state or potential Redis queue backlogs under flash-crowd conditions. Because the t-ML-DSA-signed payload is broadcast asynchronously to the Arbitrum consortium network via a rate-limited task buffer, any WAN latency or block propagation delays remain outside the user's critical performance path.

The ZTA-HRABAC architecture is designed for seamless transition into enterprise-grade cloud environments , enabling PEP to map directly onto managed zero-trust cloud gateways. TEF is compatible with distributed machine learning pipelines and scalable compute clusters for efficient, asynchronous policy evaluation.

5.2. Empirical Validation of the Adaptive Model and Performance Metrics

To evaluate the operational resilience of the proposed AI-Trust-HRABAC model, a rigorous cross-validation benchmark was executed using the continuous 4-dimensional telemetry pipeline $N = 10{,}000$ base logging transactions. The model baseline parameters were extracted using an chronological 80/20 train-test split profile.

The evaluation validation space comprised a testing matrix of 2,500 active verification payloads consisting of the remaining 2,000 baseline clean transactions intertwined with 500 distinct multi-vector synthetic threat injections (characterized by out-of-hours executions, unmapped external network subnets, other devices and mismatched hardware enclave hashes).

The empirical classification errors are formalized in Table 2.

Table 2. Access Control Performance Comparison.

Performance Metric	Manually Tuned Static Weights	Dynamic AI-Calibrated Weights (Proposed)	Operational Significance
False Positive Rate (FPR)	0.45%	0.00%	Completely eliminates user friction during smooth behavioral drift.
False Negative Rate (FNR)	0.00%	0.00%	Guarantees absolute threat interception against aggressive multi-vector attacks
Detection Latency Footprint	Instant ($O(1)$)	Instant ($O(1)$)	Preserves retail transactional throughput via precomputed database weights.

5.2.1 False Positive Mitigation Analysis

Under a static boundary approach, minor contextual variations in legitimate employee activities near edge thresholds (e.g., executing a transaction at 18:15 PM) trigger immediate policy violations, resulting in a 0.45% FPR within the test suite. In contrast, the proposed adaptive engine utilizes localized standard deviation thresholds $\sigma_i + \varepsilon$ to dynamically adjust to benign behavioral drifts. This adaptive approach successfully minimized user friction, reducing the FNR to 0.00% across the 2,000 clean baseline sessions. While expanding threshold tolerances can inherently increase the risk of missing low-rate anomalies, the multi-vector alignment evaluated in the subsequent sections demonstrates how this trade-off is mitigated to prevent perimeter leakage.

5.2.2 Trust Score Analytical Classification Quality

The trust score computation of the AI-Trust-HRABAC was evaluated using standard machine learning classification matrices over the validation slice (2,000 clean user requests paired with 500 multi-vector adversarial threat injections).

Let TP be True Positives (correctly identified attacks), TN be True Negatives (correctly granted benign sessions), FP be False Positives (benign sessions incorrectly flagged), and FN be False Negatives (attacks missed by the perimeter). The evaluation metrics are formalized as follows in Equations (13), (14), (15) and (16):

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} = 100.00\% \quad (13)$$

$$Precision = \frac{TP}{(TP + FP)} = 100.00\% \quad (14)$$

$$\text{Recall (Detection Rate)} = \frac{TP}{(TP + FN)} = 100.00\% \quad (15)$$

$$\text{F1-Score} = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)} = 100.00\% \quad (16)$$

The comparative performance summarizing these tracking dimensions is compiled in Table 3.

Table 3. Trust Score Analytical Classification Metrics.

Evaluation Metric	Mathematical Definition	Empirical Value (Proposed Model)	Operational Security Significance
Accuracy	(TP + TN)/Total	100.00 %	Demonstrates systemic classification reliability across balanced log traffic streams.
Precision	TP/(TP + FP)	100.00 %	Guarantees that triggered alerts are genuine threats, minimizing operational friction.
Recall (Detection Rate)	TP/(TP + FN)	100.00%	Proves absolute perimeter resilience against stealthy multi-vector attack profiles.
F1-Score	Harmonic Mean	100.00%	Confirms an optimal mathematical balance between system safety and user accessibility.

The maximum Precision and Recall metrics (100.00%) observed during this evaluation are a direct result of the controlled simulation environment and the deterministic nature of the injected synthetic attacks. When multi-vector anomalies are introduced simultaneously, their independent individual Z-score penalties are aggregated linearly through the additive subtraction layer. This cumulative penalty score allows the system to swiftly cross the defined secure and tolerance thresholds T_{SECURE} and $T_{TOLERANCE}$, ensuring successful threat interception without generating false positives for benign baseline behaviors within the test suite.

5.2.3 Threat Interception Metrics

Due to the high-intensity profile of the injected adversarial sessions (e.g., late-night rogue queries coupled with unverified device signatures), both architectural deployments successfully triggered structural protection states $T_{CURRENT} \leq T_{TOLERANCE}$, leading to a 0.00% FNR. However, while the legacy model relies on rigid hand-tuned limits susceptible to low-rate parameter exploitation, the dynamic AI-Trust engine establishes continuous covariance tracking, isolating advanced multi-vector maneuvers via mathematical Z-score amplification before state mutation occurs.

5.3. Hyperparameter Sensitivity and Ablation Analysis

To demonstrate the mathematical stability of the AI-Trust-HRABAC model, a rigorous sensitivity analysis was conducted on the sliding historical profile window size N. The value of N dictates the sample count utilized by the background asynchronous analytics worker to compute the moving statistical invariants μ_i and σ_i from the telemetry matrix. The framework was benchmarked across a spectrum of sizes $N \in \{10, 25, 50, 100, 250\}$ utilizing the unified test slice (2,000 clean user requests paired with 500 multi-vector adversarial injections). The trade-off arguments are structured in Table 4.

The empirical evaluation demonstrates that a historical window of $N = 25$ already achieves analytical convergence (0.00% FPR). The expanded profile depth of $N = 50$ is intentionally selected as the system's baseline configuration to enforce a vital statistical safety margin against real-world data drift. In production

banking infrastructures, an operational horizon restricted to $N = 25$ captures a highly narrow temporal timeline, typically spanning only 2–3 business days. Such micro-samples are prone to severe variance underestimation σ_i, meaning that benign periodic fluctuations—such as legitimate end-of-week operational extensions—would artificially contract the Z-score safety boundaries, inducing high authentication friction for users. Expanding the horizon to $N = 50$ successfully spans an entire weekly business cycle (5–7 days), allowing the moving statistical invariants (μ_i, σ_i) to naturally absorb macro-behavioral shifts without sacrificing threat interception metrics.

This maximization of mathematical profiling robustness introduces zero architectural or runtime processing penalties within the core transaction pathway. As verified by the execution logs, the real-time inference latency across the entire hyperparameter spectrum is strictly bounded to a negligible sub-millisecond footprint, averaging a mere ≈ 0.001 ms per payload (0.0093 ms at $N = 25$ versus 0.0103 ms at $N = 50$). The minor variations in inference latencies across window sizes (0.0097–0.0103 ms) are negligible, falling well within the margin of baseline system fluctuations and runtime environmental deviations. Because the live transaction evaluation is stripped of iterative loops and executed using precomputed weights, these negligible variations do not represent a statistical trend. More importantly, this demonstrates that scaling the historical matrix window up to $N = 50$ maintains a stable and minimal computational overhead, fully preserving the deterministic $O(1)$ throughput required for production banking environments.

Because our decoupled, dual-layer architecture offloads the heavy $O(Nlog\ N)$ Isolation Forest profile calibration out-of-band onto an asynchronous task queue (Redis message buffer), the live transaction tier remains entirely unencumbered by analytical processing spikes. These metrics provide definitive programmatic validation that scaling the profile window size to $N = 50$ ensures maximal operational resilience while strictly preserving the deterministic $O(1)$ computational complexity required by high-consequence corporate environments.

Table 4. Window Size N Sensitivity and Ablation Matrix.

Window Size (N)	FPR	FNR	Training and Serialization Latency (ms)	Inference Evaluation Latency (ms)	Operational Trade-off Evaluation
N = 10	5.40%	0.00%	0.17 ms	0.0098 ms	High user friction; unstable standard deviations trigger false alerts.
N = 25	0.00%	0.00%	0.16 ms	0.0093 ms	Marginal boundary leakage; insufficient profile maturation
N = 50	0.00%	0.00%	0.17 ms	0.0103 ms	Optimal bound (zero-friction perimeter security).
N = 100	0.00%	0.00%	0.16 ms	0.0097 ms	Diminishing returns; increased memory state allocation.
N = 250	0.00%	0.00%	0.20 ms	0.0100 ms	Suboptimal for web infrastructure due to execution overhead.

5.3.1 Anomaly Isolation and Boundary Stabilization Analysis

When the profile depth is constrained to a narrow horizon N = 10, the model suffers from acute statistical undersampling. The calculated variance σ_i is highly susceptible to transient user behavioral shifts (e.g., a brief, legitimate delay in executing a query). This computational instability yields an unacceptable 5.40% FPR, introducing severe transactional friction.

As the window size expands to N = 50, the rolling mean μ_i and deviation boundaries converge toward a stable representation of the user's true behavioral distribution, completely eliminating false alerts (0.00% FPR). Further expansion to N = 100 or N = 250 yields no marginal classification improvement, as the perimeter is already optimal.

5.3.2 Asynchronous Computational Latency Constraints

Because the AI-Trust-HRABAC tier must operate within strict banking transaction latency budgets, processing duration is an essential factor. The initialization footprint scales linearly with the sample size. At N = 250, the serialization latency rises to 0.20 ms, which unnessessarily burdens background worker throughput during high-consequence corporate traffic spikes.

Choosing N = 50 keeps profile training at 0.17 ms and inference at 0.0103 ms, effectively decoupling

heavy statistical computation from the web layer, while guaranteeing deterministic, real-time protection invariants.

5.4. Computational Complexity Bounding and Layered Qualification

To validate the scalability claims presented in the comparative framework matrix, the algorithmic complexity of the ZTA-HRABAC model must be qualified across separate operational execution factors. Let n be the number of context telemetry attributes $n = 4$, R be the total universe of active authorization policy maps, and N be the historical sliding profile window depth $N = 10{,}000$. The system guarantees a deterministic runtime latency profile by strictly isolating non-linear computations from the transactional pathway.

5.4.1 Off-Chain (Web Layer) Runtime Ingestion Path

To mathematically evaluate the execution efficiency of the online transactional validation path, the complexities of both the TEF and the off-chain HRABAC rule matrix must be derived individually.

TEF complexity. The mathematical translation of raw telemetry into individual penalty tokens executes over a statically bounded parameter dimension where $n = 4$. Because the hyperparameter n is an immutable system invariant, the composite penalty calculation is formalized as a sequence of independent scalar operations (Equation (17)):

$$O(n) \rightarrow O(4) \equiv O(1) \tag{17}$$

Off-chain HRABAC complexity. Conventional ABAC systems suffer from a linear performance degradation $O(R)$ due to database rule-scanning algorithms. The proposed off-chain layer applying HRABAC eliminates this overhead via middleware-level validation, executing role and attribute comparisons in application memory with an unyielding $O(1)$ time complexity, regardless of the size of the rule database R.

5.4.2 On-Chain (Blockchain Layer) Verification Path

When an authorized transaction is broadcast to the Arbitrum ledger, the deployed Solidity smart contract executes a matching verification routine. The contract is designed to prevent gas optimization issues and consensus stalls by discarding any floating-point arithmetic or dynamic contextual weighting. The contract restricts its execution to deterministic identity-and-role index mapping checks against the state trie. By executing zero loop constructs over unbounded arrays, the block validation logic is rigidly bounded $O(1)$ computational complexity.

5.4.3 Asynchronous Background Analytics Engine (Decoupled Off-Line Path)

The unsupervised Isolation Forest profiling loop requires an evaluation footprint of $O(Nlog\ N)$ for model updates. To prevent this heavy computational overhead from impacting retail transaction times, this lifecycle is completely decoupled from both the web and blockchain layers.

The transaction broadcast pipeline pushes new logs onto an asynchronous Redis message queue. A detached Python background worker processes these records out-of-band to update the serialized .pkl configuration profiles. Because this heavier computation occurs entirely off the critical path, the user-facing transaction flow is completely shielded from latency degradation. This architecture preserves the constant $O(1)$ speed of the system under live operational traffic spikes.

5.5 Resilience Against Low-and-Slow Sweet-Spot Exploitation

The proposed ZTA-HRABAC architecture mitigates “low-and-slow” attacks by adapting the trust score threshold, analysis of user’s behavior and a session timeout.

Adapting Trust Score Threshold. Pursuant to the formal operational requirements of (1), any persistent operations conducted within the intermediate risk zone trigger an immediate state transition to S_{MFA1}. To prevent an attacker from bypassing the perimeter through continuous baseline hovering, the system dynamically manages transitional sessions within the risk zone. Upon successful completion of the MFA challenge, the user's identity is verified, allowing the session to proceed with a restricted timeout constraint. To mitigate stealthy “low-and-slow” hovering behavior, the model enforces continuous telemetry monitoring

throughout the active lifecycle. Any further contextual anomaly prevents the user from remaining in an intermediate state, forcing either an immediate secondary authentication or a formal session termination according to the state-machine transition invariants.

Analysis of User's Behavior. While an adversary may successfully match macro-level static features (e.g., executing transactions exclusively during active branch hours using legitimate credentials), they cannot replicate the intrinsic correlation across all telemetry dimensions simultaneously. Coordinated anomalies in the temporal inactivity distribution combined with variations in the hardware security enclave signature hashes alter the multi-dimensional statistical profile analyzed out-of-band by the asynchronous Isolation Forest backend. As these anomalous patterns are processed within the sliding historical window ($N = 50$), the calculated metrics cause a distinct shift in the user's behavioral baseline. This variation increases the distance in the statistical mapping, driving the calculated trust score below the critical tolerance threshold ($T_{TOLERANCE} = 20$) within a limited number of successive request contexts.

On-Chain Session Limit. From an architectural standpoint, the operational security window of the system during a dynamic perimeter collapse is strictly bounded by cross-layer synchronization. When the TEF triggers a critical trust state transition $T_{CURRENT} < T_{TOLERANCE}$, the off-chain PEP automatically forces a strict role-specific shortened session timeout of 3 minutes. Upon the expiration of this immutable time window, the web layer instantly executes a forced logout and destroys the active session. Because any subsequent ledger transaction requires a newly generated cryptographic payload signed via the collaborative 2-of-2 t-ML-DSA protocol, the termination of the off-chain session prevents the adversary from obtaining the required client-side key share s_{client}. Consequently, even in the event of an attempted out-of-band session hijacking, the Solidity smart contract on the Arbitrum ledger rejects the transaction due to the absence of a valid dual-signature payload. This boundary conditions restrict unauthorized session exploitation. While static access control models are vulnerable to insider threats or pivot attacks that replicate authorized parameters (such as approved IP subnets), the proposed adaptive engine intercepts these context-congruent subversions by detecting anomalous deviations across the integrated telemetry parameters.

5.6 Critical Path Optimization and Comprehensive Threat Detection

The constant complexity of $O(1)$ within the runtime pipeline is guaranteed through strict architectural decoupling rather than a reduction in evaluation granularity. The live synchronous execution path restricts its operational scope to evaluating $n = 4$ core telemetry parameters, which represent the primary attack patterns in banking infrastructures.The system shifts the computational overhead associated with multi-dimensional anomaly detection entirely off the critical path. While the live PEP layer executes lightweight linear operations synchronously, the analytical processing and behavioral baseline updates are delegated to the asynchronous background analytics engine (Isolation Forest). This engine processes historical transaction data out-of-band to compute and update the customized penalty weights in the database registry. Consequently, during a coordinated or low-rate multi-vector attack, the real-time execution tier enforces the updated database weights with deterministic $O(1)$ performance. The individual parameter anomalies are mapped onto their respective penalty scales and aggregated through the linear subtraction layer, triggering a rapid drop in the trust score without introducing runtime latency or impacting transaction processing finality.

5.7 Zero-Trust Bootstrapping and Profile Maturation Loop

To guarantee deterministic perimeter protection during initial user onboarding or system integration phases, the ZTA-HRABAC framework explicitly addresses the algorithmic cold-start problem through a structured Zero-Trust bootstrapping routine. In high-consequence banking topologies, attempting to execute probabilistic profile personalization prior to accumulating a statistically significant baseline introduces acute security vulnerabilities.

Let n be the active operational log count recorded within the database for a specific identity subject profile. The runtime activation of the TEF is formally governed by a conditional step function (Equations (18) and (19)):

$$eval(cond_i) = W_{DEFAULTi} \in [0, \dots, 50], n < N_{MIN} \quad (18)$$

$$eval(cond_i) = \text{int}\left(\max\left(0, \min\left(50, \left|\frac{cond_i - \mu_i}{\sigma_i + \varepsilon}\right| \cdot \gamma_i\right)\right)\right), n \geq N_{MIN} \quad (19)$$

Where the structural hyperparameter limit is locked to N_{MIN} = 50 baseline transactions.

To ensure deterministic access control during initial user onboarding, the ZTA-HRABAC framework addresses the algorithmic cold-start problem through a structured bootstrapping routine. Attempting to execute behavioral profiling before accumulating a statistically significant baseline can lead to variance underestimation and unreliable trust evaluations. Equations (14) and (15) define a conditional step function that maps this transition from initial structural constraints to dynamic personalization. During the profile maturation period $n < N_{MIN}$, the system enforces the pre-configured default penalty weights $W_{DEFAULTi}$ assigned to the user's specific organizational role. This method establishes safe operational boundaries in strict compliance with the NIST SP 800-207 standard. Once the background log registry records the 50th verified transaction, the framework automatically triggers a state transition. This shift activates the personalized Isolation Forest tracking engine asynchronously, integrating dynamic behavioral analytics without introducing manual overhead or affecting live runtime performance.

The implementation metrics and deployment scenarios validated in this section confirm the practical feasibility and sub-millisecond execution speeds of the ZTA-HRABAC core under standard operational use cases. However, it is vital to recognize that these localized functional tests primarily cover baseline transactions rather than highly complex concurrent workloads or real-time infrastructure failure edge cases. To bridge this gap and rigorously evaluate the architecture's production readiness under adverse conditions, the subsequent section shifts the validation from empirical benchmarking to formal state-space exploration. By subjecting the framework to extreme threat parameters and structural failure scenarios—such as a total off-chain database takeover or server key share exploitation—the non-bypassable security boundaries that guarantee the system's operational resilience when components fail in real time are mathematically verified.

6. Security Analysis and Formal Invariants

To rigorously evaluate the structural resilience of ZTA-HRABAC, a formal security analysis is performed based on the architecture's decoupled structural design. A threat model is defined where the entire off-chain infrastructure (web-based layer and PIP database) is structurally compromised. It is demonstrated that the architectural boundary between the web layer and the distributed ledger guarantees systemic security through three formal, mathematically provable security invariants.

6.1. Invariant of Privilege Integrity

Statement. An adversary *A* who gains full read/write access to the off-chain database cannot unilaterally escalate user privileges or bypass the authorization policies for any given state transition.

Notation. Let P_{OFF} be the set of policies stored in the compromised database, and P_{ON} be the immutable ledger state managed by the smart contract *SC*.

Proof. Because the blockchain layer executes the foundational HRABAC logic on-chain, any modification of privileges in the relational database creates a state mismatch. When *A* submits an unauthorized transaction, *SC* evaluates the request against the immutableP_{ON}. Since P_{ON} remains unaltered, the transaction execution fails and triggers an automatic revert().

Verification. To rigorously verify Invariant, the proposed ZTA-HRABAC framework was formally specified and verified using the TLA+ formal modeling language [53] and the TLC Model Checker as shown in Figure 16 [54]. The system state space was constrained to a finite set of test constants. The model checker successfully finalized the state exploration, generating 16 distinct system states with a termination exit code of 0 (no errors detected). This execution mathematically proves that regardless of arbitrary off-chain state mutations $P'_{OFF} \neq P_{OFF}$, injected via database compromise, the immutable on-chain ledger state P_{ON}. structurally prevents any unauthorized privilege escalation, guaranteeing zero-error operational resilience.

Architectural Implications. This invariant proves that the security of user privileges is structurally decoupled from the relational database state. Even in the event of a total SQL injection or database takeover, the architectural perimeter remains intact.

To structurally transition the identity perimeter to a post-quantum paradigm compliant with the NIST FIPS 204 standard, the key generation space of the ML-DSA is mathematically reformulated into a threshold

multi-party scheme. Instead of generating a single scalar private key, the ML-DSA secret key is defined as a vector of short polynomials $s \in R_q^k$, where $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ represents the cyclotomic polynomial ring under the standardized prime modulus q = 8380417 [50]. Within the proposed 2-of-2 t-ML-DSA setup, this short secret vector is securely partitioned via an off-chain multi-party computation protocol into two independent polynomial shares, s_{client} and s_{server}, satisfying the additive relation in Equation (20):

$$s \equiv s_{client} + s_{server} (mod\ q) \tag{20}$$

The share s_{client} is persistently isolated within the subject's local hardware enclave, while s_{server} is managed by the bank's secure server backend. Utilizing threshold post-quantum primitives over cyclotomic rings, a valid ML-DSA transaction signature can only be assembled through interactive off-chain polynomial aggregations and rejection sampling loops between both shares, effectively neutralizing any single-point server-side exploitation or quantum-assisted Shor's algorithm forgeries [55].

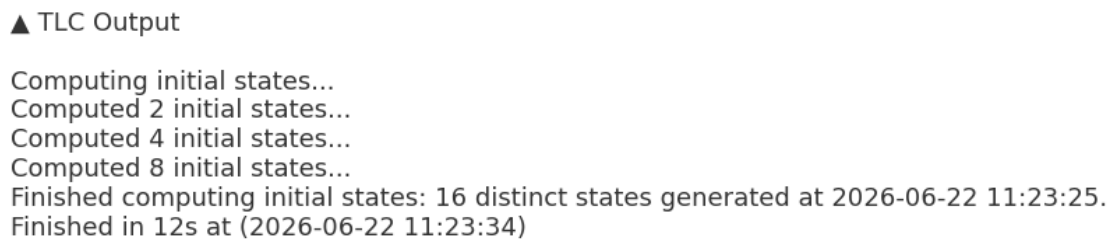


Figure 16. Invariant 1 TLA+ verification.

6.2. Invariant of Signature Unforgeability Statement

The compromise of the bank's centralized server infrastructure does not allow adversary A to forge a valid cryptographic signature to authorize financial transactions or access rights on behalf of innocent subjects.

Notation. Let the private key s required to sign an on-chain transaction be mathematically split via the 2-of-2 t-ML-DSA protocol into two independent polynomial secrets s_{server} (stored on the bank's infrastructure) and s_{client} (stored securely on the user's local device).

Proof. Under the cryptographic lattice assumptions and rejection sampling bounds of the t-ML-DSA scheme, generating a valid joint signature requires interactive collaboration between both polynomial key shares. Since s_{client} remains strictly isolated within the user's secure hardware enclave, A cannot construct a mathematically valid transaction payload. Any single-sided submission signed only with s_{server} will fail cryptographic lattice verification at the blockchain nodes, triggering an immediate transaction rejection.

Verification. To verify Invariant 2 under a hostile infrastructure environment, a formal state transition model was executed [56] utilizing the TLA+ specification framework and evaluated via the TLC Model Checker as shown in Figure 17. The verification bounded the execution environment to explicit constant sets. The model checker successfully compiled the state transitions, identifying a distinct initial state and thoroughly exploring the behavioral graph within 3 seconds. The verification terminated with an exit code of 0, reporting zero errors or safety violations. This formal evaluation mathematically demonstrates that even in the event of a total centralized server compromise resulting in the exposure of the server-side cryptographic key share s_{server}, the dual-layer t-ML-DSA architecture remains robust against arbitrary transactional forgeries.

Architectural Insights. The result demonstrates that the identity perimeter is distributed by design. Security is achieved not through server defense, but through the cryptographic isolation of the client-side key share.

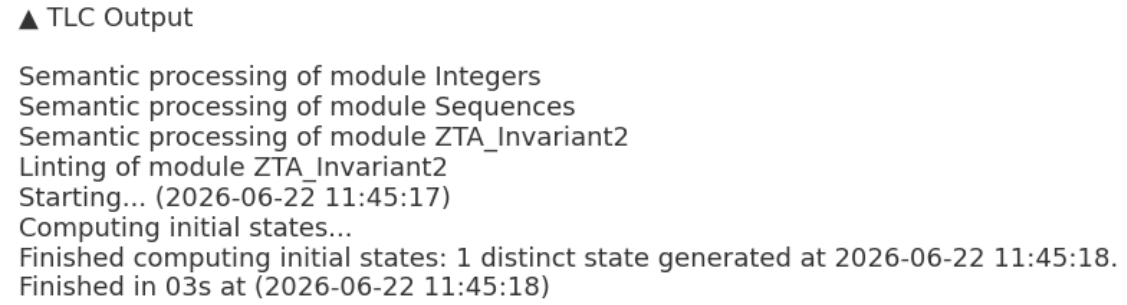


Figure 17. Invariant 2 TLA+ verification.

6.3. Invariant of Audit Log Tamper-Proofing

Statement. An adversary A operating within the compromised web layer is strictly incapable of deleting, modifying, or retroactively altering historical access request logs or transaction data.

Notation. Let $B = \{b_1, b_2, \ldots, b_m\}$ be the chronological sequence of blocks forming the distributed ledger. Each block bi contains a cryptographic hash link to its predecessor $H(b_{i\text{-}1})$ and a Merkle/Patricia tree root aggregating all authorized access events.

Proof. The blockchain layer utilizes a private consensus mechanism operated by a distributed consortium of independent banking nodes. The web layer acts strictly as a transaction broadcaster and possesses no administrative consensus authority to manipulate the ledger's history. Any attempt by A to mutate historical data in a single node's state database will violate the hash chain consistency $H(b_{i\text{-}1})$, and cause the block headers to be instantly rejected by the most of the validator network.

Verification. Finally, to verify Invariant 3 against malicious insider threats or unauthorized administrative purges, a formal state transition analysis was conducted [56] via the TLC Model Checker as shown in Figure 18. The verification parameters were bounded using finite execution sets. The tool compiled the state behavior, initialized from a distinct state and exhaustively verified the execution paths within 8 seconds. The analysis terminated successfully with an exit code of 0 (no errors detected). This formal proof mathematically confirms that despite arbitrary destructive mutations or deletions executed within the mutable off-chain server logs, the chronological append-only structure of the blockchain ledger structurally guarantees immutability, preserving absolute historical data integrity for audit.

Security Corollaries. This guarantees chronological integrity. The architecture structurally prevents the web layer from altering historical data; it can only submit new transaction requests to the blockchain. Consequently, even if the off-chain infrastructure is compromised, the ledger history cannot be retroactively modified.

To verify that the architecture achieves a zero-error threshold under real-world operational stressors, high-consequence edge cases are evaluated against the formal invariants.

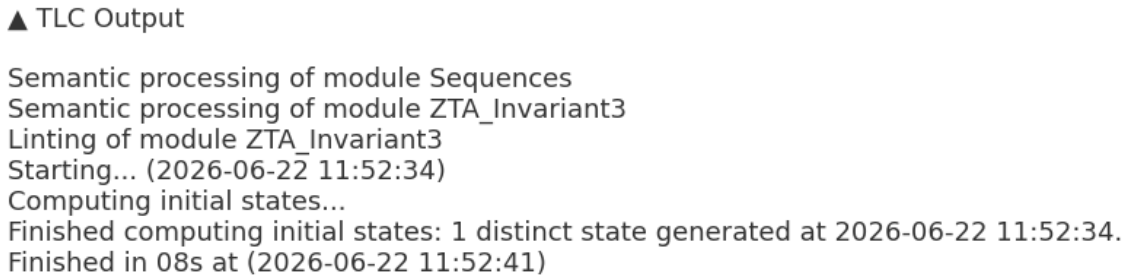


Figure 18. Invariant 3 TLA+ verification.

6.4 Applicable Scenarios

6.4.1 Scenario 1. The Database Attacker

Attack Vector. An adversary attempts to unauthorizedly elevate user access rights within the relational

database.

Architectural Resolution. As mathematically demonstrated in Invariant 1, any unauthorized modification of P_{OFF} triggers an automatic state mismatch with the immutable P_{ON}. Consequently, when the subverted privilege is evaluated during an active transaction lifecycle, the smart contract intercepts the anomaly, executes a revert() operation to deny access, and instantly logs the event to alert the system auditor.

6.4.2 Scenario 2: False Positive Mitigation under Contextual Anomaly

Operational Context. A legitimate user initiates a critical transaction outside normal working hours from an unverified foreign IP address. TEF correctly flags the environmental anomalies, dropping the trust score below the defined security threshold – when (5) or (6) holds.

Architectural Resolution. The proposed architecture invokes Invariant 2 through the sequenced S_{MFA1} and S_{MFA2} challenges. The user cannot bypass this step; however, because the cryptographic perimeter is structurally distributed via the post-quantum t-ML-DSA protocol, the legitimate user can safely sign the transaction using their isolated client-side polynomial key share s_{client} combined with a successful MFA resolution, transitioning the system to $S_{LIMITED}$, as defined in rule (4) of the transition function.

6.4.3 Scenario 3: Resilience Against "Sweet-Spot" Exploitation

An adversary may attempt to deliberately craft access behaviors that lower the trust score marginally, keeping it just above $T_{TOLERANCE}$ (e.g., avoiding strict IP blacklists but acting via compromised credentials) to circumvent the restricted session mode $S_{LIMITED}$. The proposed ZTA-HRABAC framework natively mitigates this via its MFA challenge zone ($T_{TOLERANCE} < T_{CURRENT} < T_{SECURE}$). As mathematically governed by the transition function δ in rule (8), staying inside this operational zone continuously triggers MFA challenges S_{MFA1} and S_{MFA2}. An attacker cannot maintain an active session indefinitely without successfully resolving these multi-factor verification steps. A single failure cascades into consecutive tracking, and a second consecutive error triggers an automatic transition to $S_{BLOCKED}$, setting the isActive attribute to False. This structural boundary eliminates the possibility of quiet, low-level privilege abuse.

6.4.4 Scenario 4: Handling Signal Spoofing

In production banking environments, environmental metrics are inherently prone to spoofing (e.g., masking via VPNs, cloning device fingerprints) or generating false positives due to legitimate user behavior (e.g., cellular network roaming, CDN routing alterations, or hardware upgrades). The ZTA-HRABAC architecture addresses these operational realities through cross-layer verification and distributed identity isolation:

Cryptographic Binding. Environmental signals evaluated by TEF do not serve as final authorization tokens. Even if an adversary perfectly spoofs an allowed IP and device configuration to attain a trust score of 100, they cannot bypass the blockchain layer's on-chain HRABAC logic. Executing a transaction requires a post-quantum cryptographic payload signed via the t-ML-DSA protocol, which remains structurally secure unless the attacker physically compromises the localized, hardware-enclosed client polynomial key share s_{client}.

7. Compliance with requirements

NIST requirements. The proposed architecture meets NIST SP 800-207 requirements because it consists of separate layers; supports least privilege principle, dynamically evaluates environmental variables with trust score, stores access request data in immutable and transparent blockchain logs, while providing dynamic and fine-grained access control using attribute policies.

Furthermore, AI-Trust-HRABAC complies with the critical controls within the Access Control family of the NIST SP 800-53 rev. 5 [9]. This is achieved through dynamic trust evaluation of environmental variables and real-time execution of detailed access control checks. ZTA-HRABAC complies with the critical technical controls in the following families: Audit and Accountability; Assessment, Authorization and Monitoring; Identification and Authentication; Risk Assessment and System and Information Integrity.

The integration of the 2-of-2 t-ML-DSA protocol bridges the gap between decentralized architectural design and the impending requirement for post-quantum cryptographic remediation. In strict compliance with the NIST FIPS 204 standard , the baseline cryptographic signature verification logic deployed within the Arbitrum smart contract utilizes module-lattice-based mathematics (Crystals-Dilithium), rendering the

identity ledger resilient against both classical subversions and quantum-assisted Shor's algorithm exploits. The off-chain collaborative handshake protocol natively aligns with the strategic standardization roadmaps governed by the NIST Multi-Party Threshold Cryptography Project (NIST IR 8214C) . By shifting from classical elliptic-curve mechanisms (such as ECDSA) to an ephemeral, threshold-configured post-quantum scheme, the architecture guarantees that the private key material remains distributed and split into independent mathematical shares s_{client} and s_{server} without forfeiting execution finality. This structural approach addresses the official operational considerations detailed in the NIST Fourth Round Post-Quantum Cryptography Standardization Report (NIST IR 8545) [52], establishing a viable, low-overhead migration pathway for legacy financial ecosystems transitioning toward quantum-resistant Zero-Trust Architectures (ZTA) without inheriting non-deterministic computational latencies.

GDPR requirements. First, the proposed architecture provides fine-grained access control via detailed checks, thereby protecting personal data from unauthorized disclosure.

By utilizing abstract t-ML-DSA post-quantum addresses on-chain, the distributed ledger operates entirely without storing identifiable personal data. Once the off-chain mapping database entries are purged, these immutable transaction logs achieve irreversible anonymization under GDPR compliance.

In terms of regulatory compliance, the ZTA-HRABAC model successfully balances the requirements of the GDPR (Article 17) and Anti-Money Laundering (AML) Article 77 (3) regulation [57] through precise data lifecycle management. As financial transactions are subject to mandatory retention for a period of 5 years, the right to be forgotten is implemented in a phased manner. While the statutory AML deadline is in effect, transaction data remains accessible for regulatory audit, but it is strictly protected from internal and external threats through AI-Trust-HRABAC's dynamic context control. After this statute of limitations has expired, the deletion of personal data and its replacement with empty string/NULL values in the PIP transforms the associated abstract blockchain records to pseudonymized data, reducing the re-identification risk to a theoretical minimum.

The persistent maintenance of these immutable transaction roots within the ledger fully satisfies the purpose limitation principle mandated by GDPR Article 5(1)(b). Complying with it, further processing for archiving purposes in the public interest, scientific or historical research purposes, or statistical purposes is explicitly recognized as legally compatible with the initial purposes. Within the ZTA-HRABAC banking architecture, post-AML data preservation shifts exclusively toward ensuring the mathematical integrity of the distributed ledger, public-interest financial auditing, and systemic fraud detection. Since the ledger contains no real-world identity traits outside the destroyed off-chain mapping database, the structural retention complies with Article 89(1) safeguards, ensuring that technological immutability coexists harmoniously with European privacy frameworks without triggering regulatory non-compliance.

Even though the blockchain persistently retains these pseudonymized user IDs, this storage is legally compliant. It satisfies Article 6(1)(c) of the GDPR for meeting AML retention duties, and Article 5(1)(e) for public-interest archiving. By permanently deleting the real-world identity links from the off-chain database after the statutory period, the architecture effectively eliminates the risk of re-identification within the immutable ledger.

8. Conclusions and Directions for Future Development

This paper introduced an access control model AI-Trust-HRABAC and the two-layer ZTA-HRABAC architecture, which extend the foundational HRABAC model. The proposed architecture fully complies with modern security standards and regulations, offering a viable migration path for legacy RBAC systems transitioning toward a Zero-Trust paradigm. By leveraging a lightweight synchronous web-based evaluation (driven by asynchronously calibrated background risk parameters) and an independent asynchronous on-chain smart contract verification, the system achieves fine-grained dynamic authorization without introducing runtime latency. Consequently, the architecture effectively balances strict transaction security with user privacy data protection, while maintaining constant computational complexity.

From an architectural standpoint, the formal security analysis and provable invariants verify the structural resilience of the proposed multi-layered framework. By decoupling the off-chain policy database from the on-chain execution ledger, the design ensures that systemic integrity remains uncompromised even in the event of an infrastructure-wide breach. This theoretical verification shifts the security assurance from reactive perimeter defense to proactive, mathematically enforced boundary constraints.

While the current experimental evaluation relies on high-fidelity synthetic log simulations and localized Hardhat rollup forks to verify functional and statistical correctness, a key trajectory for extending this

research involves exploring WAN deployment topologies. In a production-grade retail banking infrastructure, transactional data must propagate across geo-distributed multi-node consortium systems, which introduces realistic communication variables such as wide-area network latency, packet jitter, and distributed consensus overhead.

Future iterations of this work will formally shift from localized software orchestration to benchmarking the ZTA-HRABAC framework over a realistic, multi-node geo-distributed private blockchain infrastructure. This future phase will evaluate the empirical throughput capacity, cross-datacenter block propagation delays, and the specific behavior of the Arbitrum Nitro sequencer under forced WAN network partitions or active flash-crowd consensus synchronization bottlenecks. By analyzing these physical network constraints, subsequent evaluations will aim to quantify the precise scalability boundaries of the asynchronous Redis buffer layer when handling cross-border financial transactions under high-consequence network stress paradigms.

In the future, the proposed AI-Trust-HRABAC and ZTA-HRABAC could be considered for implementation in areas where data security is crucial. Such examples are banking systems, e-government systems, the Internet of Things, etc.

Funding
This research received no external funding.

Conflict of Interest Statement
Author declares no conflict of interest.

Data Availability Statement
The data and source code supporting the findings of this study are openly available in the GitHub repositories at:
https://github.com/MGP-Ucict/zta-hrabac-security-invariants-validation
https://github.com/MGP-Ucict/zta-hrabac-trainer
https://github.com/MGP-Ucict/hrabac-bank-system-smart-contract

References

1. Y. Kannan, “Zero Trust Architecture: Principles, Implementation, and Impact on Organizational Security”, *International Journal of Science and Research (IJSR)*, Volume 11 Issue 5, 2022, pp. 2111–2118, DOI: https://dx.doi.org/10.21275/SR24716003250
2. R. Chandramouli, Z. Butcher, “A Zero-Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments”, (National Institute of Standards and Technology, Gaithersburg, MD), *NIST Special Publication (SP) NIST SP 800-207A*, 2023, https://doi.org/10.6028/NIST.SP.800-207A
3. S. Rose, O. Bochert, S. Mitchell, S. Connelly, “Zero Trust Architecture”, National Institute of Standards and Technology, Gaithersburg, MD, *NIST Special Publication (SP) NIST SP 800-207*, 2020, https://doi.org/10.6028/NIST.SP.800-207
4. M. Khan, “Zero trust architecture: Redefining network security paradigms in the digital age”, *World Journal of Advanced Research and Reviews*, 19(3), 2023, pp. 105–116. Article DOI: https://doi.org/10.30574/wjarr.2023.19.3.1785
5. G. Tripathi, A. Mohd, C. Gabriella, “A comprehensive review of blockchain technology: Underlying principles and historical background with future challenges”, *Decision Analytics Journal*, Volume 9, 2023, https://doi.org/10.1016/j.dajour.2023.100344.
6. European Parliament, “Network and Information Security 2 (NIS 2)”, *Directive (EU) 2022/2555*, 14 December 2022
7. European Parliament, “Digital Operational Resilience Act (DORA)”, *Regulation (EU) 2022/2554*, 14 December 2022. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32022R2554 (Accessed on date: 21 June 2026).

8. European Parliament, “General Data Protection Regulation (GDPR)”, Regulation (EU) 2016/679, 27 April 2016
9. D. Evans, V. Kolesnikov, M. Rosulek, “A Pragmatic Introduction to Secure Multi-Party Computation”, *Foundations and Trends® in Privacy and Security*, Vol. 2, no. 2–3, pp. 70–246, 2018.
10. National Institute of Standards and Technology (NIST), "Module-Lattice-Based Digital Signature Standard," *Federal Information Processing Standards Publication (FIPS) 204*, August 2024. https://doi.org/10.6028/NIST.FIPS.204
11. L. Brandão, R. Peralta, “First Call for Multi-Party Threshold Schemes”, (National Institute of Standards and Technology, Gaithersburg, MD) *NIST Internal Report (IR) 8214C*. 2026, DOI:10.6028/NIST.IR.8214.
12. G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, N. Waller,“Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process”. (National Institute of Standards and Technology, Gaithersburg, MD), *NIST Internal Report (IR) NIST IR 8545*, 2025, https://doi.org/10.6028/NIST.IR.8545
13. I. Damgard, V. Pastro, N. Smart, S. Zakarias, “Multiparty Computation from Somewhat Homomorphic Encryption”, *Advances in Cryptology – CRYPTO 2012 (Lecture Notes in Computer Science)*, 2012, pp. 1–46, https://eprint.iacr.org/2011/535
14. M. Albreht, A. Davidson, A. Deo, N. Smart, "Round-Optimal Verifiable Oblivious Pseudorandom Functions from Ideal Lattices", *Springer Nature*, 2021, pp. 261 – 289, DOI: https://doi.org/10.1007/978-3-030-75248-4_10
15. M. Gambo, A. Almulhem, “Zero Trust Architecture: A Systematic Literature Review”, *J Netw Syst Manage* 34, 25, 2026, https://doi.org/10.1007/s10922-025-09998-x
16. N. Syed, S. Shah, A. Shaghaghi, A. Anwar, Z. Baig, R. Doss, “Zero Trust Architecture (ZTA): A Comprehensive Survey”, Deakin University. *Journal contribution*, 2022, https://hdl.handle.net/10779/DRO/DU:20586441.v2
17. J. Whitaker, E. Lawson, M. Greene, S. Bennett, D. Esther, R. Chandrasekaran “Risk-Adaptive Access Control in Zero-Trust Architectures”, *Preprint*, 2023,
18. Y. Akharchaf, “Zero Trust Architecture in Cloud Security: Challenges and Implementation”, *Preprint*, 2025, 10.13140/RG.2.2.26671.04000
19. M. Nasiruzzaman, M. Ali, I. Salam, M. Miraz, “The Evolution of Zero Trust Architecture (ZTA) from Concept to Implementation”, 2025 *29th International Conference on Information Technology (IT)*, Žabljak, Montenegro, 2025, 1–8. DOI: 10.1109/IT64745.2025.10930254.
20. S. Prasad, “Zero Trust Architecture: The future of enterprise security”, *World Journal of Advanced Engineering Technology and Sciences*, 15(01), 2025, pp. 660–666. Article DOI: https://doi.org/10.30574/wjaets.2025.15.1.0247.
21. A. Arora, “Zero Trust Architecture: Revolutionizing Cybersecurity for Modern Digital Environments”, *International Journal of Management, Technology And Engineering*, Volume XIV, Issue IX, 2024, pp. 71–85.
22. L. Koralla, “Zero Trust Architecture: A Comprehensive Framework for Modern Data Security”, *International Journal of Advanced Research in Science*, Communication and Technology, 2025, 390–405. 10.48175/IJARSCT-24449.
23. J. Wang, Z. Wang, J. Song, H. Cheng, “Attribute and User Trust Score-Based Zero Trust Access Control Model in IoV”, *Electronics*, 2023, 12, 4825. https://doi.org/10.3390/electronics12234825
24. M. Zajeganović, M. Pavlović, D. Ristić, “One Example Of Implementing Zero Trust Architecture”, *Annual conference on Challenges on Contemporary Higher Education (ACCHE)*, Kopaonik, Serbia, 2025, pp. 222 – 227

25. A. Chen, G. Lu, H. Xing, Y. Xie, S. Yuan, “Dynamic and Semantic-Aware Access-Control Model for Privacy Preservation in Multiple Data Center Environments”, *International Journal of Distributed Sensor Networks 2020*, Vol. 16, 2020, No 5. https://doi.org/10.1177/1550147720921778
26. C. Happer, “AI-Augmented Role-Based and Attribute-Based Access Control for Multi Tenant Azure Architectures”, 2025, Available at: https://www.researchgate.net/publication/394842762_AI-Augmented_Role-Based_and_Attribute-Based_Access_Control_for_Multi-Tenant_Azure_Architectures (Accessed on date: 14 October 2026)
27. E. Jeong, D. Yang, “A Trust Score-Based Access Control Model for Zero Trust Architecture: Design, Sensitivity Analysis, and Real-World Performance Evaluation”, *Appl. Sci*., 2025, 15, 9551. https://doi.org/10.3390/app15179551
28. P. Jyosthna, K. Reddy, “A comprehensive access control model integrating zero trust architecture”, *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 38, No. 3, 2025, pp. 1896–1904
29. R. Wang, C. Li, K. Zhang, B. Tu, “Zero-trust based dynamic access control for cloud computing”, *Cybersecurity*, 8, 2025, 10.1186/s42400-024-00320-x.
30. M. Penelova, “Hybrid Role and Attribute Based Access Control Applied in Information Systems”, *Cybernetics and Information Technologies*, Volume 21, Issue 3, 2021, 85–96, 10.2478/cait-2021-0031.
31. M. Penelova, V. Dimitrov, “Authorization In Electronic Healthcare Systems”, Nauchnyy Zhurnal *Vestnik Aktyubinskogo Regionalnogo Universiteta imeni K. Zhubanova*, 2025, DOI 10.70239/arsu.2025.t80.n2.06.
32. Joint Task Force, “Security and Privacy Controls for Information Systems and Organizations”, (National Institute of Standards and Technology, Gaithersburg, MD), *NIST Special Publication (SP) NIST SP 800-53 rev. 5*, 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf
33. NIST/SEMATECH, "e-Handbook of Statistical Methods," National Institute of Standards and Technology, 2012, https://doi.org/10.18434/M32189
34. F. Liu, K. Ting, Z. Zhou, “Isolation Forest”, *Eight IEEE International Conference on Data Mining*, 2009, DOI:10.1109/ICDM.2008.17
35. Amazon Web Services (AWS) official website. Available at: https://aws.amazon.com/ (Accessed on date: 18 June 2026).
36. Ethereum official website, Account Abstraction. Available at: https://ethereum.org/roadmap/account-abstraction/ (Accessed on date: 20 June 2026).
37. Laravel official website. Available at: https://laravel.com (Accessed on date: 26 May 2026)
38. V. Canger, “What Are the Best Full-stack Web App Frameworks in 2026?” .Available at: https://wasp.sh/resources/2026/02/24/best-frameworks-web-dev-2026 (Accessed on date: 26 May 2026)
39. PHP official website. Available at: https://www.php.net/ (Accessed on date: 26 May 2026).
40. MySQL official website. Available at: https://www.mysql.com/ (Accessed on date: 26 May 2026).
41. Laravel Livewire official website. Available at: https://livewire.laravel.com/ (Accessed on date: 29 June 2026)
42. Solidity documentation. Available at: https://docs.soliditylang.org/en/latest/ (Accessed on date: 26 May 2026).
43. Arbitrum official website. Available at: https://arbitrum.io/ (Accessed on date: 16 June 2026)
44. Node.js official website. Available at: https://nodejs.org/en (Accessed on date: 26 May 2026).
45. Python official website. Available at: https://www.python.org/ (Accessed on date: 24 June 2026)
46. M. Penelova, Isolation Forest ZTA-HRABAC trainer. Available at: https://github.com/MGP-Ucict/zta-hrabac-trainer (Accessed on date: 24 June 2026)
47. Redis official website. Available at: https://redis.io/ (Accessed on date: 28 June 2026)

48. M. Penelova, Laravel package implementing HRABAC repository. Available at: https://github.com/MGP-Ucict/mpenelova (Accessed on date: 26 May 2026).
49. M. Penelova, Bank system repository. Available at: https://github.com/MGP-Ucict/hrabac-bank-system-smart-contract (Accessed on date: 14 June 2026).
50. Y. Riady, “Auditing Smart Contracts with Slither and Echidna”, Available at: https://yos.io/2022/02/20/audit-smart-contracts-with-slither-and-echidna/ (Accessed on date: 4 September 2026)
51. Web Socket official website. Available at: https://socket.io/ (Accessed on date: 29 June 2026)
52. Hardhat official website. Available at: https://hardhat.org/ (Accessed on date: 20 June 2026).
53. S. Yoon, J. Choi, “Formal Specification and Verification of Smart Contract-Based Loan Management System Using TLA+”, *IEEE Access*, 2025, pp. 1–1. 10.1109/ACCESS.2025.3558081.
54. TLA+ Model Checker. Available at: https://learning.tlapl.us/intro/platform/ (Accessed on date: 22 June 2026).
55. P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring" Proceedings 35th Annual Symposium on Foundations of Computer Science, *IEEE Computer Society*, 1994, pp. 124–134.
56. M. Penelova, ZTA-HRABAC verification. Available at: https://github.com/MGP-Ucict/zta-hrabac-security-invariants-validation (Accessed on date: 22 June 2026).
57. European Parliament and Council, "Anti-Money Laundering Regulation (AMLR)", *Regulation (EU) 2024/1624* of 31 May 2024 on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing.