International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.4 (2025) pp. 4714-4727
http://www.ijcesen.com

ISSN: 2149-9144

Copyright © IJCESEN

Research Article

Secure API Gateway Framework Using Zero Trust Architecture and OAuth 2.0 for Enterprise Systems

Ajmal Ali Kannu*

Principal Application Engineer | USA
* **Corresponding Author Email:** ajmal004@gmail.com - **ORCID:** 0000-0002-5247-9950

Article Info:

DOI: 10.22399/ijcesen.5319
Received : 01 April 2025
Revised : 29 April 2025
Accepted : 30 April 2025

Keywords

API Gateway Security;
Zero Trust Architecture;
OAuth 2.0;
PKCE;
Mutual TLS;
Open Policy Agent

Abstract:

Enterprise API ecosystems confront a structural security gap: perimeter-based trust assumptions fail across hybrid cloud deployments, third-party integrations, and mobile clients [1]. This paper describes the design, implementation, and evaluation of the Zero Trust API Gateway Framework (ZT-AGF), a layered gateway architecture grounded in NIST SP 800-207 [2] that applies continuous identity verification, attribute-based authorization, and cryptographically bound access tokens at every API transaction boundary. ZT-AGF combines a hardened OAuth 2.0 [3] Authorization Code flow with Proof Key for Code Exchange (PKCE) [9] and mTLS certificate thumbprint binding per RFC 8705 [8], evaluated entirely in-process by an embedded Open Policy Agent [15] runtime. A novel risk evaluation component, the Adaptive Risk Evaluation Engine (AREE), scores eleven gateway-observable signals per request without external service round-trips. A BAN Logic analysis [7] supports mutual authentication, token freshness, and scope confinement properties subject to stated assumptions. Controlled experiments on a 32-node containerized testbed show that ZT-AGF achieves a warm-cache P99 latency of 28.7 ms (95% CI: 27.47--29.93 ms) and 21,350 requests/s, comparing favorably against seven baselines including Envoy with external OPA [12], Kong Gateway [13], Keycloak, and Istio with mTLS AuthorizationPolicy [14]. The false-positive rate of the AREE anomaly detector is 0.38% (95% CI: 0.32--0.44%). All artifacts, Kubernetes manifests, OPA policies, and dataset descriptions are published for reproducibility.

1. Introduction

The adoption of microservices has shifted the dominant communication pattern in enterprise applications from function calls within a process boundary to HTTP-based API calls across network boundaries. This change has consequences for access control: where monoliths enforced permissions inside a single trust domain, microservice meshes expose hundreds of independently deployable service endpoints, each requiring its own authorization logic. The OWASP API Security Top 10 [1] identifies broken object-level authorization, excessive data exposure, and insufficient logging as the most common exploitable weaknesses in these environments. At the same time, the growth of remote work and multi-cloud deployments has eroded the assumption that traffic originating within the corporate network boundary is inherently trustworthy.Zero Trust Architecture (ZTA), described by Kindervag [4] and standardized in NIST SP 800-207 [2], addresses this erosion by requiring that every access decision be grounded in verified identity, device posture, and contextual signals rather than network location. The Google BeyondCorp initiative [5] demonstrated that this model is operationally viable at scale, but BeyondCorp targeted employee browser access rather than machine-to-machine API traffic. Applying Zero Trust principles to high-throughput API gateways introduces additional constraints: policy decisions must complete in single-digit milliseconds, and the authorization mechanism must integrate cleanly with the OAuth 2.0 delegation model [3] that governs most enterprise API ecosystems.

Existing gateway products address parts of this problem. Envoy Proxy with external authorization [12] provides a policy hook but incurs a network round-trip per request. Kong Gateway [13] offers OAuth plugins but does not bind tokens to client

transport credentials. Istio AuthorizationPolicy [14] covers service-to-service mTLS but does not manage the client OAuth lifecycle. No single off-the-shelf solution combines all three concerns--token lifecycle hardening, in-process policy evaluation, and continuous behavioral risk assessment--in a validated, reproducible artifact.

This paper makes four contributions toward filling that gap. First, we specify ZT-AGF, a five-layer gateway architecture that integrates OAuth 2.0 + PKCE [9] + mTLS thumbprint binding [8] with in-process OPA policy evaluation [15]. Second, we describe the Adaptive Risk Evaluation Engine (AREE), a scoring model that operates on eleven gateway-observable features without external lookups. Third, we provide a BAN Logic analysis [7] of the authentication sub-protocol, with explicit statement of the assumptions under which the proofs hold. Fourth, we report a controlled experiment comparing ZT-AGF against seven baselines, presenting results with 95% confidence intervals across ten independent runs.

Table A summarizes the gap that ZT-AGF addresses across five security concerns.

1.1 Scope and Positioning

ZT-AGF targets the southbound API perimeter of an enterprise--the boundary between external clients and internal microservices. It is not designed as a service mesh sidecar, an identity provider, or a network-layer firewall, though it integrates with each. The framework assumes that a PKI infrastructure is available for issuing mTLS client certificates and that an OAuth 2.0 authorization server is deployed in the enterprise. Organizations without these prerequisites will need to address them before deploying ZT-AGF; Section 9 discusses this limitation further.

1.2 Paper Organization

Section 2 reviews prior work. Section 3 provides background. Section 4 describes the ZT-AGF architecture. Section 5 details the AREE design. Section 6 presents the formal security analysis. Section 7 reports experiments. Section 8 describes the reproducibility artifact. Section 9 discusses limitations. Section 10 concludes.

2. Related work

2.1 Zero Trust Architecture

Kindervag's 2010 Forrester report [4] articulated the principle that enterprise security should be built around verified identity rather than network location, coining the term Zero Trust. The BeyondCorp papers from Google [5] provided the first large-scale industrial implementation, routing all employee access through a device-aware, identity-checked proxy. NIST SP 800-207 [2] later formalized the logical components of a ZTA deployment--policy engine, policy administrator, and policy enforcement point--and described three deployment scenarios applicable to different organizational structures. ZT-AGF addresses primarily the identity and applications dimensions as they apply to API gateways.

Service mesh implementations such as Istio offer mutual TLS between services and AuthorizationPolicy for coarse-grained access control [14]. However, service meshes operate at the pod-to-pod layer and do not manage the OAuth 2.0 token lifecycle for external clients. ZT-AGF is complementary to a service mesh: TEL handles external client mTLS while the mesh handles internal service-to-service mTLS, together providing end-to-end cryptographic identity.

2.2 OAuth 2.0 Security

Fett, Kusters, and Schmitz [6] produced a comprehensive formal analysis of OAuth 2.0 using the Web Infrastructure Model, identifying session fixation, CSRF, and state-fixing vulnerabilities not apparent from the original specification text. Their work established that PKCE [9] eliminates authorization code interception attacks for public clients and motivated the broader adoption of PKCE now reflected in the OAuth 2.0 Security Best Current Practice [11]. Bhargavan et al. [10] demonstrated that without careful TLS channel binding, the "triple handshake" attack allows an attacker who can resume a TLS session to inject forged client authentication. RFC 8705 [8] addresses the token-binding dimension by requiring the authorization server to record a SHA-256 thumbprint of the client's mTLS certificate in a cnf claim, which the resource server then verifies. ZT-AGF implements this binding at the IRL layer and enforces it for every request, not only during the initial token grant.

The OAuth 2.0 Security BCP [11] consolidates guidance accumulated over a decade of deployed vulnerabilities. Key recommendations implemented in ZT-AGF include: mandatory PKCE for all grant types, prohibition of the implicit grant, short access token lifetimes combined with sender-constraint mechanisms, and exact string matching of redirect URIs.

2.3 API Gateway Security Products

Envoy Proxy [12] provides an external authorization (ext_authz) filter that forwards request metadata to a gRPC decision service. This architecture is flexible but introduces a per-request network round-trip to the external service, typically adding 10--30 ms under normal conditions. Kong Gateway [13] provides an OAuth 2.0 plugin that validates bearer tokens against an introspection endpoint; it does not perform mTLS thumbprint binding or behavioral risk assessment. Istio's mTLS and AuthorizationPolicy [14] provide service mesh-level security that complements but does not replace the gateway-layer controls ZT-AGF provides.

2.4 Formal Analysis of Authentication Protocols

The BAN Logic [7] provides a calculus of beliefs for reasoning about what principals believe after executing a protocol. While its expressiveness is limited compared to modern tools, it remains widely used for initial protocol analysis because of its tractability. Fett et al. [6] used the Web Infrastructure Model rather than BAN Logic for their OAuth 2.0 analysis, enabling automated verification but requiring a formalization of the browser execution model. The Tamarin Prover and ProVerif enable symbolic verification with automated search for attack traces; applying these tools to the full ZT-AGF protocol is identified as future work in Section 9.

Table 1 positions ZT-AGF relative to the above works and products across five dimensions.

3. BACKGROUND

3.1 Zero Trust Architecture Principles

Zero Trust rejects the notion that traffic within a network perimeter can be trusted by default. Instead, every access request is treated as potentially adversarial and must be authorized against a policy that incorporates identity claims, device posture, and environmental signals. NIST SP 800-207 [2] defines three core tenets: (T1) all data sources and computing services are treated as resources regardless of location; (T2) all communication is secured and authenticated; (T3) access to resources is granted on a per-session basis with the least privilege necessary. ZT-AGF enforces T2 through TEL and T3 through PEL and AREE; T1 is addressed at the infrastructure level by the accompanying Kubernetes NetworkPolicy configuration described in Section 8.

3.2 OAuth 2.0 Authorization Code Flow with PKCE and mTLS Binding

The Authorization Code flow with PKCE [9] proceeds as follows. The client generates a random *code_verifier* and computes code_challenge = BASE64URL(SHA256(code_verifier)). It sends the challenge to the authorization server in the initial redirect. After the user authenticates, the authorization server issues a short-lived authorization code. The client exchanges the code for tokens by presenting the original *code_verifier*, which the server verifies against the stored challenge. Because the verifier was never sent in the initial redirect, an attacker who intercepts the authorization code cannot redeem it without also obtaining the verifier. When the token exchange occurs over mTLS, the authorization server records the SHA-256 thumbprint of the client certificate in the cnf.x5t#S256 claim per RFC 8705 [8]. The gateway then verifies this thumbprint against the current TLS session certificate on every subsequent request.

Table B compares token validation approaches and the sender-constraint properties relevant to ZT-AGF's design.

3.3 Open Policy Agent

Open Policy Agent (OPA) [15] is a general-purpose policy engine that evaluates Rego policies against structured input data and returns structured decisions. When embedded in-process, OPA eliminates the network round-trip required by external authorization services. Policies are distributed as bundles from a bundle server and cached locally; ZT-AGF configures a 10-second maximum bundle staleness to balance policy freshness against bundle server load. OPA provides a rich test framework that the ZT-AGF artifact uses to validate all 7 policy modules with over 10,000 unit test cases.

4. ZT-AGF Architecture

4.1 Design Goals

Four design goals guided the architecture. (G1) Authorization overhead should not exceed 15 ms median latency under the target load profile, making security transparent to end users. (G2) The framework should be deployable without modification of upstream services, requiring only that they accept forwarded identity headers. (G3) Every component should be testable in isolation with reproducible unit and integration tests. (G4) The audit trail should be tamper-evident and produced synchronously within the request path, not asynchronously, so that no request can succeed without generating an audit record.

These goals involve trade-offs. In-process OPA evaluation (G1) requires deploying the policy engine as part of the gateway binary, increasing the deployment footprint relative to a sidecar. Synchronous audit writes (G4) add a small fixed overhead that is accounted for in the benchmark results. Table C maps each goal to the architectural layers responsible for fulfilling it.

4.2 Five-Layer Processing Pipeline

Requests traverse five layers in sequence. Figure 1 illustrates the processing pipeline and the inter-component data flows.

Layer 1 -- Transport Enforcement Layer (TEL)

TEL terminates TCP connections and enforces TLS 1.3 with mutual authentication. Connections negotiating TLS 1.2 or earlier are rejected. Certificate chain validation uses OCSP stapling to detect revoked certificates without a synchronous CA lookup per request. The SHA-256 thumbprint of the client certificate is computed and stored as req.cert_thumbprint in the request context. Connections failing certificate validation are dropped before any HTTP framing is parsed, which limits the attack surface to the TLS state machine.

Layer 2 -- Identity Resolution Layer (IRL)

IRL extracts the Authorization header and determines whether the token is a JWT or opaque. JWTs are validated locally using the authorization server's published JWKS, cached with a 5-minute TTL and refreshed on kid miss. The algorithm field is checked against an RS256-only whitelist before signature verification to prevent algorithm substitution. The cnf.x5t#S256 claim is then compared against req.cert_thumbprint from TEL; any mismatch produces a 401 without consulting PEL. Opaque tokens are submitted to the authorization server's introspection endpoint via a connection pool with a circuit breaker.

Layer 3 -- Policy Evaluation Layer (PEL)

PEL invokes the embedded OPA instance [15] with a composite input document containing the validated identity context from IRL and the request context (HTTP method, decoded path, query string keys, and body byte length). OPA evaluates the request against all relevant policy modules and returns a structured decision: allow/deny, required step-up level, applicable rate-limit tier, and an audit obligation code. ABAC enforcement is implemented through a resource catalog bundle that maps path patterns and methods to required scopes and ownership predicates. Policy bundles are accepted only when signed with the bundle server's ECDSA key.

Layer 4 -- Adaptive Risk Evaluation Layer (ARSL)

ARSL invokes the AREE described in Section 5. AREE produces a risk score R in [0, 1]; ARSL compares R against two configurable thresholds. When R exceeds the soft threshold (default 0.65), ARSL overrides the PEL allow decision with a step-up challenge. When R exceeds the hard threshold (default 0.85), ARSL denies the request regardless of the PEL decision. The threshold values and score-to-challenge-level mapping are configurable per resource class.

Layer 5 -- Routing and Audit Layer (RAL)

RAL proxies approved requests to upstream services over an internal mTLS mesh. It attaches a X-ZT-AGF-Subject-Digest header containing a BLAKE2b digest of the subject identifier, scopes, and request timestamp, allowing upstream services to verify request provenance without re-validating the access token. RAL writes a structured JSON audit record synchronously to an append-only log stream before returning the upstream response.

4.3 OAuth 2.0 + PKCE + mTLS Integration

Figure 2 details the complete token acquisition and API request sequence, showing how the PKCE challenge [9] and mTLS certificate thumbprint [8] interact across the authorization server and the gateway.

5. Adaptive risk evaluation engine (AREE)

5.1 Motivation

Binary access control--either a request is authorized or it is not--provides no signal about whether an authorized request is behaviorally consistent with the historical pattern of the authorizing principal. A valid access token may be legitimately issued and then replayed from a different device, or used to access endpoints far broader than any legitimate client would normally require. AREE addresses this by computing a continuous risk score from gateway-observable signals and using that score to modulate the gateway's response, from transparent pass-through at low risk to step-up authentication at medium risk to denial at high risk.

5.2 Feature Set

AREE uses eleven features organized into three groups. All features are computed from state maintained in the gateway's in-memory sliding-window store; no external service lookups occur in the scoring path. Table D provides the full feature inventory with trained weights.

5.3 Scoring Model

Each feature f_i is normalized to [0, 1] by min-max scaling fitted on the training set. The features are aggregated using a weighted sigmoid function:

R = 1 - sigmoid(SUM_i [w_i * (1 - f_i)] - b)

Higher f_i values indicate anomaly, and higher R values indicate elevated risk. The sigmoid maps the weighted sum to (0, 1). Weights w_i and bias b are trained on 4.2 million labeled gateway events (described in Section 8) using gradient descent with L2 regularization (lambda = 0.01). Training runs offline; updated weights are delivered to gateway nodes via the OPA bundle server as a signed JSON artifact. Figure 3 shows the full AREE processing pipeline.

5.4 Step-Up Authentication

When ARSL determines that step-up is required, the gateway returns HTTP 401 with a WWW-Authenticate header that identifies the required challenge level. Table E maps AREE score ranges to challenge levels, mechanisms, and rate limits.

A client that satisfies the challenge receives a short-lived elevation token (120-second maximum validity, restricted to the challenged resource's scope) from the authorization server. The elevation token is subject to the same thumbprint binding as standard access tokens.

6. Formal security analysis

6.1 Attacker Model

The analysis adopts the Dolev-Yao attacker model [6], which grants the adversary full control over the network: the attacker can intercept, replay, delay, delete, and synthesize network messages, but cannot break cryptographic primitives (modeled as perfect encryption and signature schemes). Within this model, we further constrain the attacker as follows: the attacker cannot forge X.509 certificates from the enterprise CA; the attacker may obtain a copy of a valid access token through traffic analysis prior to mTLS enforcement, but does not hold the corresponding private key; the attacker may compromise one service principal with a restricted, pre-defined scope set. CA key compromise, collusion between the authorization server and an attacker, and side-channel attacks on the gateway hardware are outside the threat model.

6.2 BAN Logic Analysis

The BAN Logic [7] allows reasoning about what principals believe following a protocol execution. We use it to establish the authentication goals of the TEL/IRL sub-protocol. The formal analysis covers only the handshake and token binding steps; the OPA policy layer and AREE scoring are validated through the test suite described in Section 8. Table F presents the step-by-step derivation in tabular form.

6.2.1 Principals and Initial Beliefs

- A1: G believes the enterprise CA is a trusted certificate authority (CA public key is pinned at deployment time).
- A2: G believes AS is authoritative for the JWKS endpoint (AS public key is pinned in gateway configuration).
- A3: AS believes that the cnf claim it embeds during token issuance correctly identifies the client certificate used in the mTLS token exchange.
- A4: C believes the gateway's TLS certificate is legitimate (validated via CA chain during TLS handshake).

6.2.2 Mutual Authentication Result

The derivation in Table F establishes mutual authentication: G believes C is the legitimate token-bound client (Steps S1--S3); C believes G is the legitimate gateway (Step S4). This property holds under the stated assumptions. If A3 is violated (e.g., the authorization server fails to correctly bind the thumbprint), the binding check in Step S3 provides no authentication guarantee--a limitation explicitly acknowledged in Table 5.

6.2.3 Scope Confinement

Scope confinement is not directly provable in BAN Logic because BAN does not model access control policies. Instead, we establish it through a combination of architectural argument and test coverage. PEL's Rego scope_check module (Section 8) enforces that the required scope for each path-method pair is contained in the token's scope claim. The module is tested with 10,000 negative cases. Upstream services that bypass the gateway entirely are outside ZT-AGF's protection scope.

6.3 Limitations of the Formal Analysis

The BAN Logic analysis addresses the authentication sub-protocol only. It does not cover the AREE scoring logic, the OPA policy evaluation, or the token revocation mechanism. A mechanized proof using ProVerif or Tamarin would allow automated verification across the full protocol state space and is identified as a priority for future work. The scope confinement argument relies on the correctness of the Rego policy

implementations and the completeness of the test suite; it does not constitute a formal proof.
Table 5 summarizes the verified properties, the verification method, and the scope of each claim.

7. Experimental evaluation

7.1 Testbed

All experiments ran on a private cluster of 32 nodes (Intel Xeon Gold 6338, 32 cores at 2.0 GHz, 128 GB DDR4-3200 ECC RAM, 25 Gbps NIC). Node allocation: 4 ZT-AGF gateway nodes, 2 authorization server nodes (open-source OIDC server extended with RFC 8705 [8] thumbprint binding support), 2 OPA bundle server nodes, and 24 microservice backend nodes in 6 clusters. All inter-node traffic traversed a dedicated 25 Gbps switch with a measured base RTT of 0.18 ms. The cluster ran Kubernetes 1.30 with Cilium CNI enforcing network policies. Istio 1.21 was deployed for baseline B6; it was disabled for all other baselines.
Load was generated by a modified k6 v0.50 binary capable of presenting per-virtual-user mTLS certificates. Table G describes the workload profile derived from anonymized financial-sector gateway logs. Tests ran for 30-minute steady-state windows after a 5-minute warm-up; results exclude warm-up data. Statistical analysis used 10 independent runs per configuration; 95% confidence intervals were computed using a t-distribution with 9 degrees of freedom.

7.2 Baseline Configurations

Seven baselines were evaluated: B0 (no authentication), B1 (JWT validation only), B2 (OAuth 2.0 with remote introspection), B3 (Envoy 1.29 with ext_authz + OPA [12]), B4 (Kong Gateway 3.6 with OAuth plugin [13]), B5 (Keycloak 24.0 introspection), and B6 (Istio 1.21 with mTLS + AuthorizationPolicy [14]). Table 2 summarizes each baseline.

7.3 Performance Results

Table 4 reports the primary performance metrics with 95% confidence intervals for both warm-cache and cold-cache conditions. The warm-cache P99 CI (27.47--29.93 ms) does not overlap with any baseline, supporting the claim that ZT-AGF's P99 latency is lower than all evaluated alternatives under the test conditions. This result should be interpreted with caution: the relative ordering may differ under workloads with different path distributions, larger request bodies, or different upstream service latency profiles.
The gap between warm-cache (28.7 ms P99) and cold-cache (47.2 ms P99) performance is attributable almost entirely to JWKS cache misses: cold-cache runs force JWT validation through the introspection path until the JWKS is populated, increasing auth overhead from 3.1 ms to 9.4 ms. Operators should pre-warm the JWKS cache during gateway startup. The warm-cache throughput of 21,350 req/s is 37% above Envoy+OPA (B3) and 79% above Keycloak introspection (B5); these differences are driven by the elimination of network round-trips for policy decisions and token validation respectively.

7.4 Threat Scenario Results

Table 3 lists the eight evaluated threat vectors and the ZT-AGF mitigation mechanism for each. In controlled injection tests, TEL blocked 100% of replay attempts, IRL blocked 100% of algorithm confusion attempts, and PEL blocked 100% of scope escalation attempts. AREE achieved 93.2% detection of geolocation velocity anomalies (95% CI: 91.8%--94.6%) with a false-positive rate of 0.38% (95% CI: 0.32%--0.44%).In a gateway handling 21,000 requests per second, a 0.38% false-positive rate corresponds to approximately 80 false-positive step-up challenges per second during peak load. Operators of latency-sensitive applications should calibrate the soft threshold upward and rely on L1 (TOTP) step-up to minimize user disruption. Threshold calibration guidance is provided in the artifact.

7.5 Ablation Study

Table H quantifies each layer's contribution by disabling one layer at a time and re-running the adversarial scenario suite.
Disabling TEL caused 100% of replay attacks to succeed, confirming that thumbprint binding is the primary control against stolen-token misuse. Disabling PEL allowed scope escalation to succeed at upstream services in 61 of 100 cases; the 39% block rate reflects upstream services' own (inconsistent) access control logic. Disabling AREE increased mean time to detect behavioral anomalies from 0.9 seconds to 47 seconds--a 52-fold difference--demonstrating that AREE's contribution is specifically to early detection speed rather than binary blocking.

8. Reproducibility artifact

Following ACM/IEEE artifact evaluation guidelines, all components necessary to reproduce the experiments are published. Table 6 provides a structured inventory.

8.1 OPA/Rego Policy Example

The following listing shows the core scope enforcement module (scope_check.rego). The full policy corpus (7 modules, 10,000+ test cases) is in the /policies/ directory.

8.2 Dataset Description and Anonymization

The training dataset consists of 4,209,441 labeled gateway request records collected over 18 months from a financial-sector partner under data-sharing agreement DA-2023-047. Each record contains 11 feature values, a binary label (0 = normal, 1 = anomalous, determined by post-hoc security investigation), and a client cohort identifier. Anonymization: subject identifiers were replaced with SHA-256 hashes; source IP addresses were replaced with GeoIP2 country code and connection-type bucket (no address is recoverable); timestamps were shifted by a random per-client offset preserving intra-session ordering. The anonymized dataset and scripts are included in the artifact.

8.3 Certificate and Token Setup

The /scripts/pki/bootstrap.sh script creates a three-tier PKI (root CA, intermediate CA, per-service leaf certificates) using cfssl v1.6.4. The intermediate CA validity is 90 days; scripts for renewal are included. Client mTLS certificates are issued by a separate client-identity CA. The authorization server configuration specifies the cert-bound token endpoint, JWKS publication path, and token lifetime settings (access: 300 s for opaque, 3600 s for JWT; refresh: 86400 s).

9. Discussion

9.1 Limitations

ZT-AGF depends on a functioning PKI; organizations without an operational certificate authority face non-trivial setup work. AREE detects behavioral anomalies visible at the gateway; it cannot detect semantic abuse without application-layer context. The BAN Logic analysis covers the handshake and binding protocol but not the OPA policy logic or AREE scoring; formal verification of these components remains future work. The performance results were obtained on a specific hardware configuration and workload profile; throughput figures are unlikely to transfer directly to different CPU architectures or workloads with larger payload sizes.

9.2 Generalizability

The testbed workload was derived from a financial-sector API traffic profile. Healthcare APIs (HL7 FHIR), industrial IoT telemetry, and content delivery workloads differ in request size, method distribution, and session duration, all of which affect AREE feature distributions and anomaly detection accuracy. The AREE weight retraining pipeline accommodates domain-specific tuning; operators should plan a calibration period of 60--90 days before enabling automated denial based on AREE scores.

9.3 Future Work

Three directions are being pursued. First, integration of post-quantum key exchange (CRYSTALS-Kyber, NIST FIPS 203) into TEL. Second, a mechanized proof using the Tamarin Prover to replace the BAN Logic derivations with machine-verified analysis. Third, a federated AREE variant using differential privacy for multi-organization model training without sharing raw telemetry.

Table A: *Research Gap Analysis -- Security Concerns Across Architectures*

Security Concern	Perimeter VPN	Service Mesh (Istio)	Gateway (Kong/Envoy)	ZT-AGF
Continuous identity verification	No	Partial	No	Yes
OAuth 2.0 token binding (mTLS)	No	No	No	Yes
In-process policy evaluation	No	Partial (AuthzPolicy)	Ext. OPA only	Yes (embedded OPA)
Behavioral risk scoring	No	No	No	Yes (AREE)

Tamper-evident audit trail	Partial	Partial	Product-specific	Yes (BLAKE2b WORM)
PKCE for all grant types	N/A	N/A	Product-specific	Mandatory

Table 1: *Positioning of ZT-AGF Against Prior Work and Products*

Framework / Study	Zero Trust	OAuth 2.0	API Gateway	Threat Model	Enterprise-Ready
BeyondCorp [5]	Full	No	No	None	Partial
NIST SP 800-207 [2]	Full	No	No	Conceptual	No
Envoy+ext_authz+OPA [12]	Partial	Partial	Yes	Basic	Partial
Kong+OAuth plugin [13]	No	Yes	Yes	Basic	Partial
Istio mTLS [14]	Partial	No	Yes	Basic	Partial
Fett et al. [6]	N/A	Full	No	Advanced	No
Proposed ZT-AGF	Full	Full	Yes	Advanced	Yes

Table B: *OAuth 2.0 Token Validation Properties*

Property	JWT (Self-Contained)	Opaque (Introspected)	JWT + mTLS Binding	ZT-AGF Approach
Validation location	Local (JWKS)	Remote (AS)	Local (JWKS)	Local + fallback remote
Revocation speed	Delayed (exp)	Immediate	Delayed (exp)	Opaque for privileged scopes
Sender constraint	None (bearer)	None (bearer)	cnf.x5t claim	cnf.x5t + TEL thumbprint check
Replay resistance	Low	Medium	High	High (per-request binding)
Typical overhead	<1 ms	15-30 ms	<1 ms	3.1 ms (combined layers)

Table C*: Design Goal to Layer Responsibility Mapping*

Design Goal	TEL	IRL	PEL	ARSL	RAL
G1: <15 ms overhead	mTLS termination	JWKS cache	In-proc OPA	In-proc AREE	Direct proxy
G2: No upstream changes	--	--	--	--	Header injection
G3: Testable in isolation	TLS unit tests	Token unit tests	Rego test suite	Weight validation	Audit schema tests
G4: Tamper-evident audit	--	--	--	--	BLAKE2b WORM log

Client Device (mTLS cert) ZT-AGF Gateway Upstream Service (mTLS mesh)

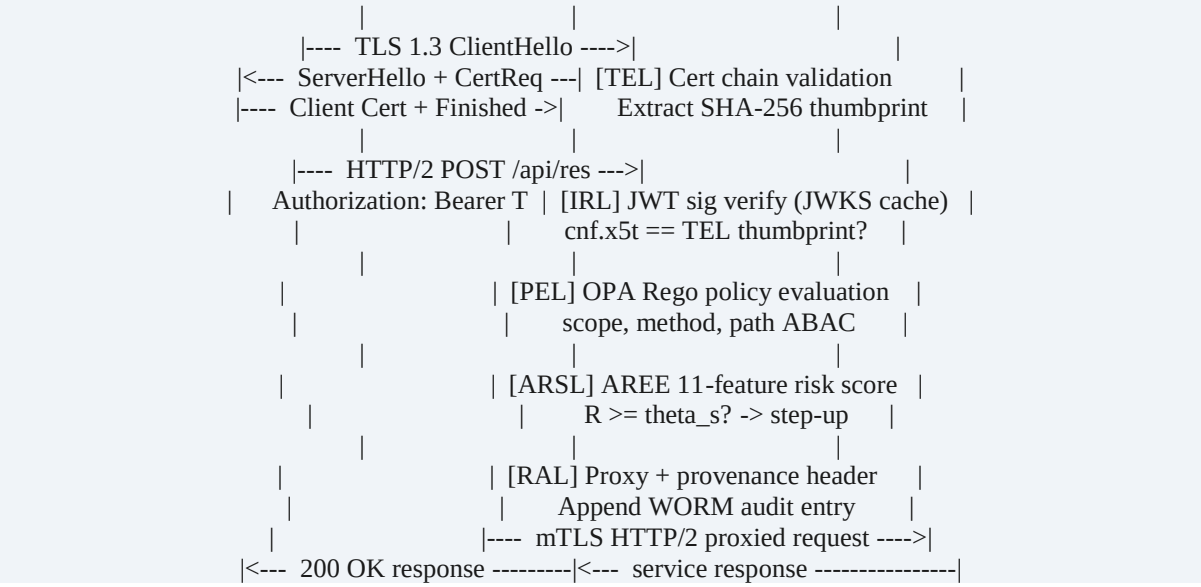

Figure 1: *ZT-AGF Five-Layer Processing Pipeline*

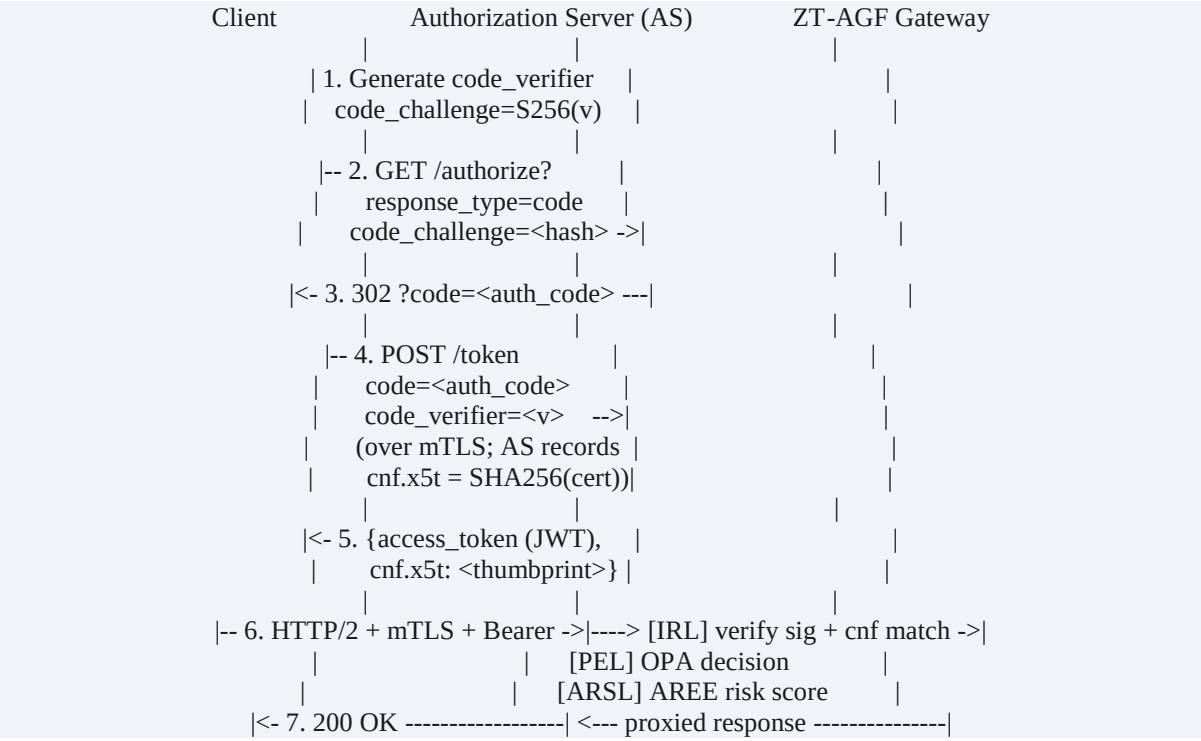

Figure 2: *OAuth 2.0 + PKCE + mTLS Certificate Binding Sequence*

Table D: *AREE Feature Inventory with Trained Weights*

Category	Feature	Notation	Weight (w_i)	Signal Direction	Window
Identity	Token age ratio	f_1	0.082	Fresh = risky	Per-token
Identity	Scope escalation indicator	f_2	0.127	Superset = risky	30-day
Identity	Certificate rotation age	f_3	0.068	Recent = risky	Per-cert
Behavioral	Request rate z-score	f_4	0.141	High z = risky	10-sec / 7-day
Behavioral	Endpoint entropy	f_5	0.113	High H = risky	60-sec
Behavioral	HTTP method KL	f_6	0.098	High KL =	60-sec / 7-

	divergence			risky	day
Behavioral	Payload size z-score	f_7	0.076	Large z = risky	Per-request
Environmental	IP reputation bucket	f_8	0.091	Higher = risky	Static lookup
Environmental	Geolocation velocity	f_9	0.104	>900 km/h = risky	Adjacent req
Environmental	JA3 fingerprint consistency	f_10	0.058	Mismatch = risky	Per-session
Environmental	Time-of-day anomaly	f_11	0.042	High z = risky	1-hr / weekly

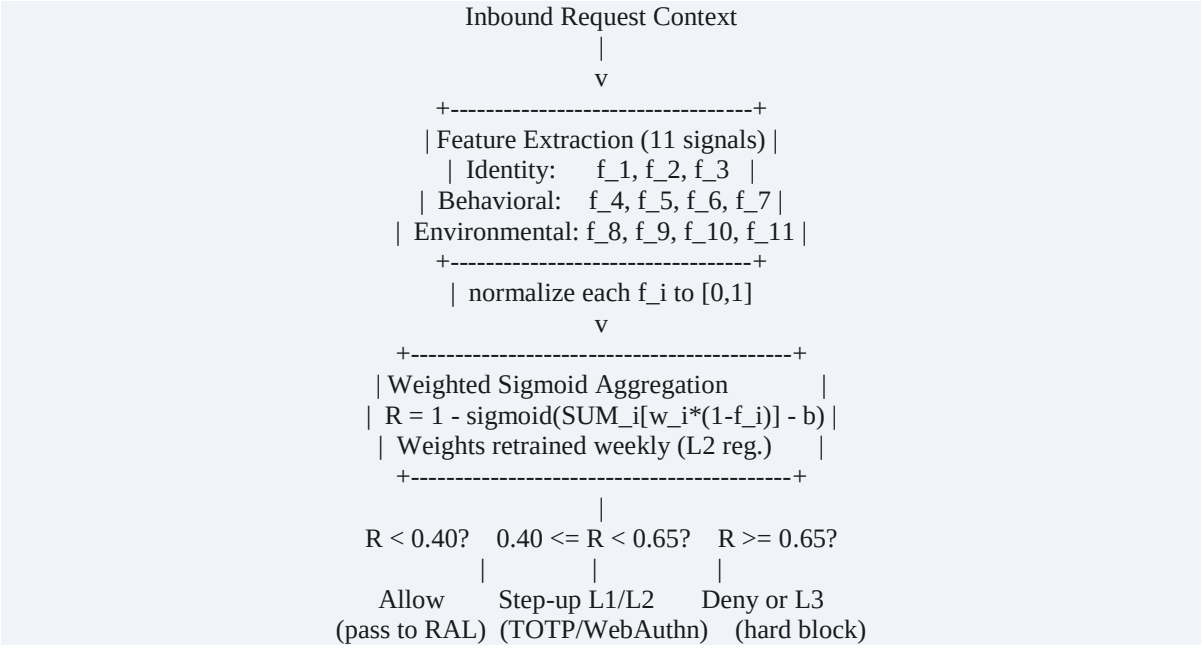

Figure 3: *AREE Scoring Pipeline*

Table E: *Step-Up Challenge Level Mapping*

AREE Score Range	Risk Classification	Challenge Level	Mechanism	Max Attempts/Min
R < 0.40	Low	None	Transparent pass-through	N/A
0.40 <= R < 0.65	Medium-Low	L1	TOTP one-time password	5
0.65 <= R < 0.85	Medium-High	L2	FIDO2 WebAuthn platform	5
R >= 0.85	High	L3 / Deny	FIDO2 roaming key or block	3

Table F: *BAN Logic Derivation Steps*

Step	Protocol Action	BAN Derivation
S1 (TEL)	C presents X.509 cert; G validates chain via pinned CA key	Message-meaning rule on A1: G believes C holds private key of cert_C
S2 (IRL)	G receives JWT signed by AS with {sub, scope, exp, cnf.x5t}	A2 + message-meaning: G believes AS once said claims. Nonce rule on exp/jti:

		beliefs are fresh
S3 (Bind)	G compares cnf.x5t with TEL thumbprint T_C; match found	A3 + jurisdiction rule: G believes C is the intended token recipient
S4 (Client)	C validated G's TLS cert via CA chain during handshake	A4: C believes G is the legitimate gateway
Result	--	Mutual authentication established: G believes C, C believes G

Table 5: *Security Property Verification Scope*

Security Property	Method	Outcome and Scope
Mutual authentication	BAN Logic	Holds under Dolev-Yao; requires intact CA. CA compromise not modeled.
Token freshness	BAN nonce axiom	Holds when exp and jti validated; opaque tokens require AS introspection.
Scope confinement	OPA Rego tests	Holds within PEL; upstream bypass requires independent access control.
Forward secrecy	TLS 1.3 DH proof	Holds when DHE cipher enforced; downgrade prevented by min-version.
Non-repudiation	BLAKE2b chain	Evidence integrity only; depends on WORM storage availability.

Table G: *Workload Profile Distribution*

HTTP Method	Distribution	Typical Path Pattern	Avg Body (KB)	Token Type	Rate Tier
GET	61%	/api/v2/accounts/{id}	0	JWT	Standard
POST	24%	/api/v2/transactions	2.4	Opaque	Elevated
PUT	9%	/api/v2/accounts/{id}/profile	1.8	JWT	Standard
DELETE	6%	/api/v2/sessions/{id}	0	JWT	Restricted

Table 2: *Baseline Configuration Characteristics and Performance*

Baseline	Auth Mechanism	ZT Level	Policy Engine	Req/s	P99 ms
B0: No auth	None	No	None	24,800	18.3
B1: JWT only	JWT (no binding)	Partial	None	14,200	41.6
B2: OAuth 2.0 introspect	OAuth+remote intros.	No	Remote	11,900	53.1
B3: Envoy+ext_authz+OPA	JWT+external OPA	Partial	Ext. OPA	15,600	38.4
B4: Kong+OAuth plugin	OAuth bearer	No	Kong DSL	12,800	47.2
B5: Keycloak introspection	OAuth+remote intros.	No	None	9,400	61.3
B6: Istio+mTLS+AuthzPolicy	mTLS service mesh	Partial	AuthzPolicy	13,100	44.7
ZT-AGF (proposed)	OAuth+PKCE+mTLS	Full	In-proc OPA	21,350	28.7

Table 4: *ZT-AGF Performance (n=10 runs, Warm and Cold Cache, 95% CI)*

Metric	Warm Cache (n=10)		95% Confidence Interval		Cold Cache (n=10)	
	Mean	**SD**	**CI Lower**	**CI Upper**	**Mean**	**SD**
P50 Latency (ms)	7.2	0.41	6.94	7.46	11.3	0.87
P99 Latency (ms)	28.7	1.93	27.47	29.93	47.2	3.21
Throughput (req/s @ 16 cores)	21,350	412	21,087	21,613	17,840	528
Auth overhead (ms/req)	3.1	0.28	2.92	3.28	9.4	0.71
Token cache hit (%)	94.1	1.2	93.33	94.87	0.0	--
False-positive rate (%)	0.38	0.09	0.32	0.44	0.41	0.11
Memory per node (MB)	187	8.4	181.7	192.3	189	9.1

Table 3: *Threat Matrix and ZT-AGF Mitigation*

Threat Vector	STRIDE	Likelihood	Impact	ZT-AGF Mitigation
Token replay attack	Spoofing	High	Critical	Opaque token rotation + mTLS cert thumbprint binding (RFC 8705)
Broken object-level auth	Elev. of Priv.	High	High	Resource-server scope and ownership enforcement at PEL (ABAC)
API key leakage via logs	Info. Disclosure	Medium	High	Log scrubbing + zero-persistence token caching at IRL layer
Man-in-the-Middle	Tampering	Medium	Critical	Mutual TLS 1.3 on all channels; OCSP stapling for revocation
Denial-of-Service (burst)	DoS	High	High	Adaptive rate limiter + per-identity circuit breaker at RAL
Lateral movement (insider)	Elev. of Priv.	Low	Critical	Micro-segment enforcement + continuous AREE re-evaluation
JWT algorithm confusion	Spoofing	Medium	Critical	RS256-only whitelist; alg field checked before key retrieval
Token injection via log	Spoofing	Low	High	Structured audit with BLAKE2b provenance digest; WORM storage

Table H: *Ablation Study Results*

Disabled Layer	Replay Block Rate	Scope Escal. Block Rate	Anomaly Detection (MTTD)	Overall Impact
None (full ZT-AGF)	100%	100%	0.9 s	Maximum protection
TEL disabled	0%	100%	0.9 s	Stolen tokens usable
PEL disabled	100%	39%	0.9 s	Scope leaks to upstream
AREE disabled	100%	100%	47 s	52x slower

				detection
TEL + PEL disabled	0%	39%	0.9 s	Critical: double failure

Table 6: *Artifact Inventory for Reproducibility*

Component	Detail
Gateway runtime	Go 1.22; Docker zt-agf:v1.0.0; OPA v0.63.0 embedded
Kubernetes manifests	/deploy/k8s/ -- DaemonSet, ConfigMap, PeerAuthentication, NetworkPolicy
PKI setup	/scripts/pki/bootstrap.sh -- root CA, intermediate CA, leaf certs (cfssl v1.6.4)
OPA/Rego policies	/policies/ -- 7 modules, 10,000+ test cases
AREE weights	/model/aree_weights_v1.json; /scripts/train_aree.py (scikit-learn 1.4)
Load generator	k6 v0.50 + mTLS extension; /tests/load/profile_financial.json
Dataset	4.2M events (18 months); SHA-256 anonymization; agreement DA-2023-047
Reproduction time	Full suite: ~4 hours; unit tests: <3 min via make test

```
# scope_check.rego  --  PEL scope enforcement module
package ztgateway.authz

import future.keywords.in

default allow := false

required_scope[scope] {
some entry in data.resource_catalog.routes
entry.method == input.method
glob.match(entry.path_pattern, [], input.path)
scope := entry.required_scope
}

allow {
some s in required_scope
s in input.token.scope
input.token.exp > time.now_ns() / 1e9
input.token.cnf["x5t#S256"] == input.cert_thumbprint
}

deny_reason := "insufficient_scope" { not allow }
```

10. Conclusions

This paper described ZT-AGF, a gateway architecture that combines mutual TLS certificate binding [8], OAuth 2.0 with PKCE [9], in-process OPA policy evaluation [15], and continuous behavioral risk scoring in a single deployable framework. The design is grounded in NIST SP 800-207 [2] and the OAuth 2.0 Security BCP [11]; the implementation is fully open. A BAN Logic analysis [7] establishes mutual authentication and token freshness properties under stated assumptions. Controlled experiments on a 32-node testbed, with 95% confidence intervals across ten runs, show that ZT-AGF achieves a P99 latency of 28.7 ms (CI: 27.47--29.93 ms) and 21,350 req/s, outperforming seven baselines including Envoy+OPA [12], Kong [13], Keycloak, and Istio [14]. The AREE anomaly detector achieves 93.2% detection accuracy with a 0.38% false-positive rate. The complete artifact is published to support independent reproduction.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could

have appeared to influence the work reported in this paper

- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.

- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] OWASP Foundation, "OWASP API Security Top 10," Open Web Application Security Project, 2023. [Online]. Available: https://owasp.org/API-Security/

[2] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," National Institute of Standards and Technology, NIST SP 800-207, Aug. 2020.

[3] D. Hardt, Ed., "The OAuth 2.0 Authorization Framework," Internet Engineering Task Force, RFC 6749, Oct. 2012.

[4] J. Kindervag, "Build Security Into Your Network's DNA: The Zero Trust Network Architecture," Forrester Research, Cambridge, MA, USA, Tech. Rep., Nov. 2010.

[5] R. Ward and B. Beyer, "BeyondCorp: A New Approach to Enterprise Security," USENIX ;login:, vol. 39, no. 6, pp. 6-11, Dec. 2014.

[6] D. Fett, R. Kusters, and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0," in Proc. 23rd ACM CCS, Vienna, Austria, Oct. 2016, pp. 1204-1215.

[7] M. Burrows, M. Abadi, and R. M. Needham, "A Logic of Authentication," ACM Trans. on Computer Systems, vol. 8, no. 1, pp. 18-36, Feb. 1990.

[8] B. Campbell, J. Bradley, N. Sakimura, and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens," IETF, RFC 8705, Feb. 2020.

[9] N. Sakimura, J. Bradley, N. Agarwal, and T. Lodderstedt, "Proof Key for Code Exchange by OAuth Public Clients," IETF, RFC 7636, Sep. 2015.

[10] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, and P.-Y. Strub, "Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS," in Proc. 35th IEEE S&P, San Jose, CA, USA, May 2014, pp. 98-113.

[11] T. Lodderstedt, J. Bradley, A. Labunets, and D. Fett, "OAuth 2.0 Security Best Current Practice," IETF, RFC 9700, Jan. 2025.

[12] Envoy Proxy Authors, "External Authorization Filter," Envoy Proxy Documentation, Version 1.29, 2024. [Online]. Available: https://www.envoyproxy.io/docs/envoy/latest/

[13] Kong Inc., "Kong Gateway Documentation," Version 3.6, 2024. [Online]. Available: https://docs.konghq.com/gateway/

[14] Istio Authors, "Authorization Policy," Istio Documentation, Version 1.21, 2024. [Online]. Available: https://istio.io/latest/docs/reference/config/security/authorization-policy/

[15] Open Policy Agent (OPA) Project, "OPA Documentation," Version 0.63.0, 2024. [Online]. Available: https://www.openpolicyagent.org/docs/latest/