ĐẠI HỌC QUỐC GIA TP HCM



**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**PHẠM XUÂN BÁCH**

**MSHV: 240202002**

**NGUYỄN KHÁNH TOÀN**

**MSHV: 240202015**

**TÊN ĐỀ TÀI: XÂY DỰNG PROXY SỬ DỤNG MTLS KẾT HỢP TOKEN‑BASED SIGNATURES CHO MÔI TRƯỜNG API & CLOUD**

Giảng viên hướng dẫn

**TS. Nguyễn Ngọc Tự**

**TP. Hồ Chí Minh, 2025**

# Giới thiệu

Trong bối cảnh bảo mật hiện đại, các tổ chức ngày càng chuyển sang mô hình Zero-Trust để bảo vệ tài sản kỹ thuật số. Đặc biệt, với sự bùng nổ của microservices và API-driven architecture, việc xác thực và ủy quyền API trở thành một thách thức quan trọng. Các phương pháp xác thực truyền thống như Bearer tokens có nhiều hạn chế về bảo mật, đặc biệt là khả năng bị đánh cắp token và tấn công replay.

Nghiên cứu này đề xuất một kiến trúc Zero-Trust API authentication kết hợp mutual TLS (mTLS) với token-based signatures, cụ thể là Proof of Possession (PoP) tokens, nhằm tạo ra một hệ thống xác thực đa lớp có khả năng chống lại các cuộc tấn công token theft và replay attack. Kiến trúc được thiết kế để triển khai trên môi trường cloud AWS với focus vào việc so sánh hiệu suất của các thuật toán mã hóa khác nhau.

Mục tiêu nghiên cứu:

* Xây dựng một API proxy gateway với Zero-Trust authentication
* So sánh hiệu suất của các thuật toán mã hóa (ECDSA, RSA, Ed25519) trong môi trường thực tế
* Đánh giá tính hiệu quả của việc kết hợp mTLS và token-based signatures
* Đề xuất các chiến lược vận hành PKI tối ưu cho môi trường cloud

# Cơ sở lý thuyết

## Định nghĩa

### Zero-Trust Architecture

Zero-Trust là một mô hình bảo mật dựa trên nguyên tắc "never trust, always verify" (không bao giờ tin tưởng, luôn xác minh). Trong bối cảnh API security, Zero-Trust yêu cầu:

* Xác thực và ủy quyền mọi request API
* Kiểm tra liên tục danh tính và quyền hạn
* Giảm thiểu surface attack thông qua least privilege access
* Giám sát và audit toàn diện mọi giao dịch

### Mutual TLS (mTLS)

mTLS là một extension của TLS protocol trong đó cả client và server đều phải xác thực lẫn nhau thông qua digital certificates:

* Transport Layer Security: Mã hóa end-to-end communication
* Bidirectional Authentication: Cả hai bên đều verify identity
* Certificate-based Identity: Sử dụng PKI infrastructure cho strong authentication
* Perfect Forward Secrecy: Bảo vệ dữ liệu quá khứ nếu private key bị compromise

### Token-Based Signatures (Proof of Possession)

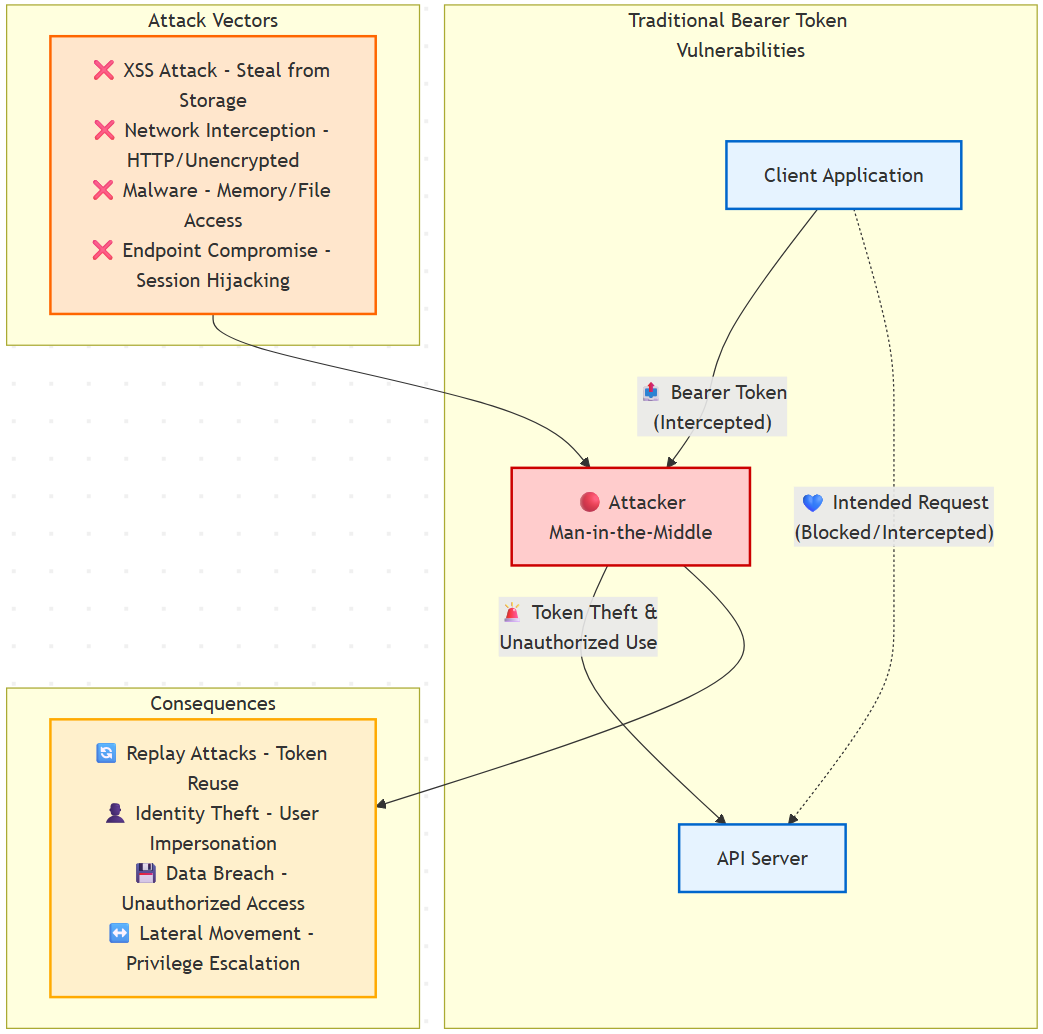
Proof of Possession (PoP) tokens là một cơ chế xác thực trong đó client phải chứng minh việc sở hữu cryptographic key được liên kết với access token:

* DPoP (Demonstrating Proof of Possession): Client tạo JWT proof ký bằng private key
* Holder-of-Key (HoK): Token được bind với specific cryptographic key
* Replay Attack Prevention: Mỗi request yêu cầu fresh cryptographic proof
* Token Binding: Liên kết token với certificate hoặc public key

## Động lực

Các hạn chế của phương pháp hiện tại:

### Bearer Token Vulnerabilities:



1. **Token Theft**: Bearer tokens có thể bị đánh cắp thông qua XSS, malware, hoặc network interception
2. **Replay Attacks**: Stolen tokens có thể được sử dụng lại mà không cần additional authentication
3. **No Cryptographic Binding**: Tokens không được liên kết với specific client identity
4. **Wide Attack Surface**: Tokens valid ở bất kỳ đâu chúng được present

### TLS-Only Limitations:

1. **Single Point of Failure**: Chỉ dựa vào server certificate validation
2. **No Client Authentication**: Không thể verify client identity
3. **Certificate Pinning Challenges**: Khó thực thi và bảo trì trong dynamic environments

### Nhu cầu về Enhanced Security:

Với sự gia tăng của sophisticated attacks và compliance requirements (PCI DSS, SOX, GDPR), các tổ chức cần:

* Multi-factor authentication cho API access
* Strong cryptographic identity binding
* Comprehensive audit trails
* Reduced blast radius khi security breach xảy ra

## Nghiên cứu liên quan

### Academic Research

Lĩnh vực bảo mật API đã có nhiều tiến bộ quan trọng thông qua các tiêu chuẩn OAuth 2.0 và các phần mở rộng (extensions) của nó. RFC 7800 [1] đã giới thiệu khái niệm Ngữ nghĩa Khóa Chứng minh Sở hữu (Proof-of-Possession Key Semantics) cho JSON Web Tokens, tạo nền tảng cho việc ràng buộc tokens với khóa mật mã. Tiếp theo đó, RFC 9449 [2] chính thức hóa giao thức OAuth 2.0 Demonstrating Proof-of-Possession (DPoP) protocol, cho phép clients chứng minh sở hữu private key thông qua việc tạo bằng chứng động. Đồng thời, RFC 8705 [3] đã chuẩn hóa OAuth 2.0 Mutual-TLS Client Authentication và Certificate-Bound Access Tokens, tạo ra khung làm việc hoàn chỉnh cho xác thực clients mạnh.

Về mặt kiến trúc Zero-Trust, NIST SP 800-207 [4] đã đưa ra định nghĩa chính thức và nguyên tắc triển khai Zero Trust Architecture vào năm 2020. Google đã tiên phong trong việc triển khai practical zero-trust model thông qua sáng kiến BeyondCorp [5], chứng minh khả năng áp dụng "never trust, always verify" principle trong môi trường enterprise quy mô lớn. Microsoft đã mở rộng ý tưởng này với Zero Trust Security Model [6], cung cấp khung làm việc toàn diện bao gồm xác minh danh tính, tuân thủ thiết bị (device compliance), và bảo vệ ứng dụng.

### Industry Implementations

Ngành tài chính đã dẫn đầu trong việc áp dụng bảo mật API nâng cao. Payment Services Directive 2 (PSD2) [7] của Liên minh Châu Âu đã quy định bắt buộc xác thực clients mạnh cho cho dịch vụ thanh toán, tạo động lực cho việc phát triển hồ sơ bảo mật API cấp độ Tài chính (Financial-grade API - FAPI) [8]. Nhóm Làm việc FAPI tại OpenID Foundation đã phát triển các yêu cầu bảo mật toàn diện sử dụng mTLS và signed JWTs, được nhiều ngân hàng lớn như Barclays, HSBC, và Deutsche Bank áp dụng. SWIFT đã cập nhật các yêu cầu Chương trình Bảo mật Khách hàng (CSP) [9] để bao gồm nhắn tin an toàn với xác thực dựa trên chứng chỉ (certificate-based authentication).

Các công ty công nghệ lớn đã triển khai nhiều phương pháp khác nhau cho xác thực giữa các dịch vụ. Google đã phát triển Istio service mesh với mTLS tự động giữa các services, hiện được sử dụng rộng rãi trong Kubernetes environments. Netflix đã mở mã nguồn Zuul gateway [10] với hỗ trợ cho xác thực dựa trên chứng chỉ, xử lý hàng triệu requests mỗi ngày.

### Existing Solutions Analysis

Các giải pháp hiện tại đều có sự đánh đổi riêng biệt. Kong Gateway [11] cung cấp hệ sinh thái plugin phong phú với hỗ trợ mTLS, nhưng thiếu native support cho cơ chế PoP nâng cao và yêu cầu cấu hình phức tạp. Istio Service Mesh [12] mTLS tự động và thực thi chính sách hiệu quả, nhưng bị giới hạn trong môi trường Kubernetes và có đường cong học tập dốc. AWS API Gateway [13] là dịch vụ được quản lý dễ cài đặt, nhưng tùy chọn tùy chỉnh bị hạn chế và tạo ra sự phụ thuộc vào nhà cung cấp. Envoy Proxy [14] cung cấp hiệu suất cao và khả năng mở rộng tốt, nhưng yêu cầu chuyên môn sâu để cấu hình đúng cách.

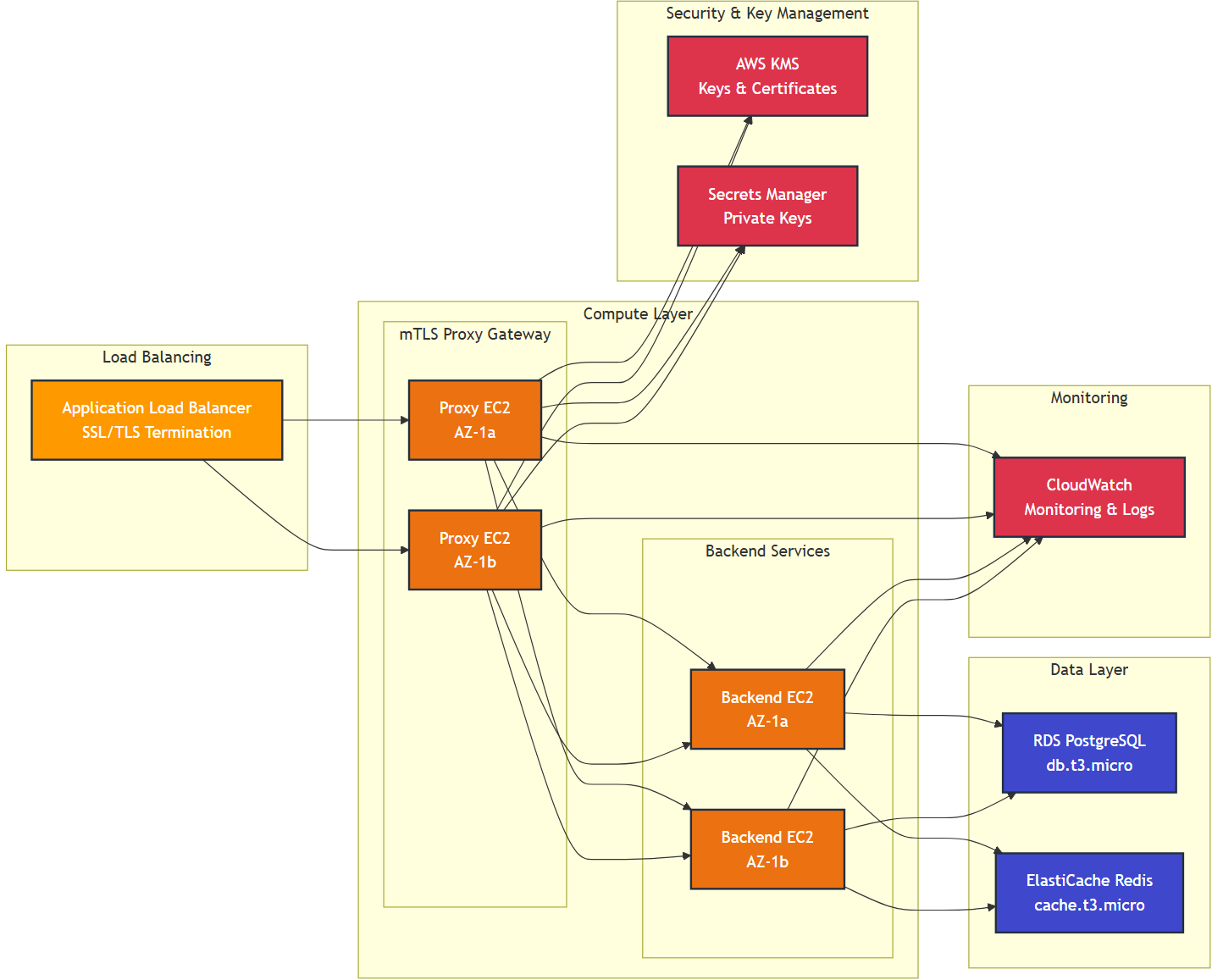
### Research Gap

Mặc dù có nhiều theoretical frameworks và individual implementations, vẫn thiếu nghiên cứu comprehensive về performance implications của việc kết hợp mTLS với token-based signatures trong production environments. Đặc biệt, chưa có systematic comparison về impact của different cryptographic algorithms (ECDSA, RSA, Ed25519) trên real-world workloads. Current literature cũng thiếu practical guidance về operational aspects như certificate lifecycle management, key rotation strategies, và cost-performance optimization cho cloud deployments.

# Phương pháp luận

## Kiến trúc đề xuất

### Tổng quan kiến trúc



### Giải thích chi tiết từng tầng

**Tầng Load Balancing**: Application Load Balancer đóng vai trò là điểm vào chính của hệ thống, thực hiện (1) chấm dứt kết nối SSL/TLS từ phía client; (2) kiểm tra tình trạng sức khỏe của các proxy instances; và (3) phân phối lưu lượng truy cập đều khắp nhiều vùng khả dụng để đảm bảo tính sẵn sàng cao.

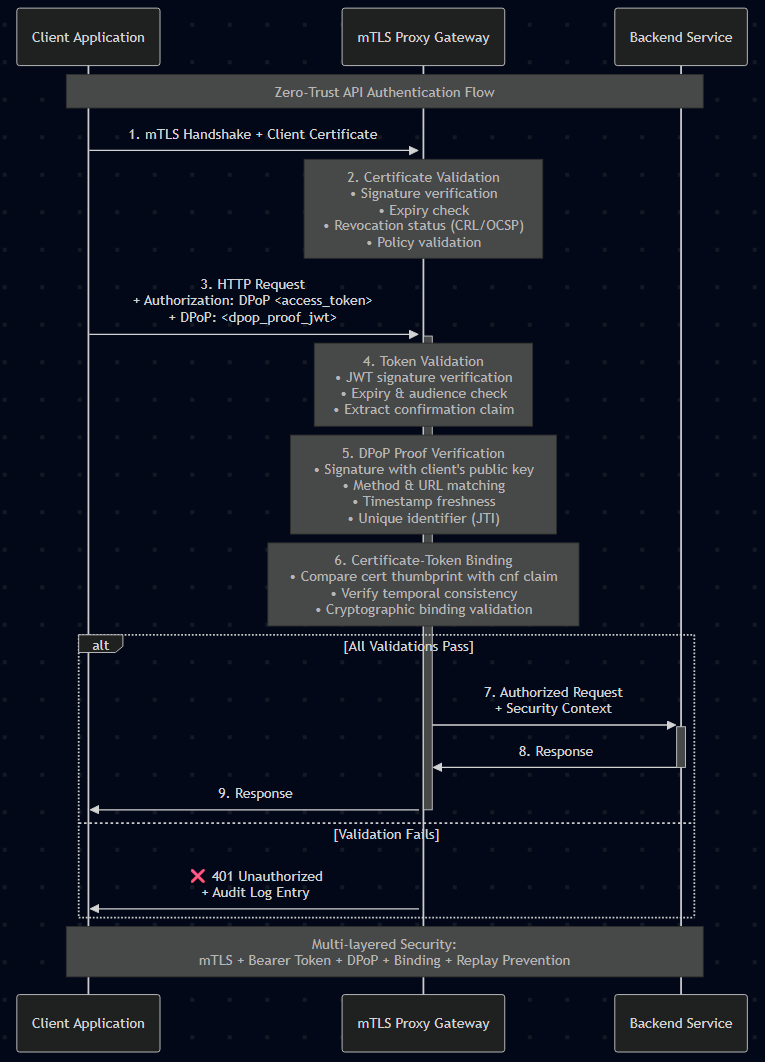
**Tầng Compute & Proxy**: Các EC2 instances được triển khai trong mạng con riêng tư trên nhiều vùng khả dụng, chạy cổng proxy mTLS được phát triển bằng FastAPI với các thư viện (1) cryptography cho xử lý chứng chỉ X.509; (2) PyJWT cho xác thực token; và (3) httpx cho reverse proxy đến các dịch vụ backend. Kiến trúc đa vùng này đảm bảo khả năng chịu lỗi cao và chỉ nhận lưu lượng từ ALB để tăng cường bảo mật.

**Tầng Security & Key Management**: AWS KMS đảm nhận việc quản lý khóa mã hóa cho chứng chỉ và token thông qua các mô-đun bảo mật phần cứng (HSMs) và cung cấp nhật ký kiểm toán đầy đủ. Secrets Manager lưu trữ an toàn (1) khóa riêng tư của chứng chỉ; (2) thông tin xác thực cơ sở dữ liệu; và (3) khóa API với khả năng tự động xoay vòng để giảm thiểu rủi ro bảo mật.

**Tầng Backend**: Các dịch vụ logic nghiệp vụ được triển khai trên FastAPI trong mạng con riêng tư với cơ chế cân bằng tải và tự động mở rộng. RDS PostgreSQL hoạt động như cơ sở dữ liệu quan hệ được quản lý với (1) sao lưu tự động; (2) khôi phục đến thời điểm cụ thể; và (3) bản sao đọc để tăng hiệu suất. ElastiCache Redis cung cấp bộ nhớ đệm trong bộ nhớ cho (1) dữ liệu phiên làm việc; (2) kết quả xác thực token; và (3) bộ nhớ đệm xác thực chứng chỉ nhằm giảm độ trễ.

**Tầng Monitoring & Logging**: CloudWatch thực hiện giám sát toàn diện với các chỉ số, nhật ký và cảnh báo cho các sự kiện hiệu suất và bảo mật. X-Ray hỗ trợ theo dõi phân tán để truy vết luồng yêu cầu qua các dịch vụ, giúp phát hiện và khắc phục sự cố nhanh chóng.

### Luồng xác thực chi tiết



Cụ thể:

**Bước 1-2: mTLS Handshake & Certificate Validation**

* Client khởi tạo TLS connection và gửi client certificate (X.509) chứa public key
* Proxy thực hiện certificate validation theo chuỗi:
  + Certificate signature verification (sử dụng CA public key)
  + Certificate expiry check (notBefore/notAfter timestamps)
  + Revocation status check (CRL/OCSP query)
  + Certificate policies và key usage validation
  + Subject Alternative Name (SAN) verification
* Nếu validation thành công, secure TLS channel được thiết lập với Perfect Forward Secrecy

**Bước 3: HTTP Request với Dual Authentication Headers** Client gửi HTTP request với hai authentication mechanisms:

Authorization: DPoP <access\_token>

DPoP: <dpop\_proof\_jwt>

Trong đó:

* access\_token: JWT với cnf (confirmation) claim liên kết đến client certificate
* dpop\_proof\_jwt: JWT proof ký bằng private key tương ứng với certificate

**Bước 4: Access Token Validation** Proxy thực hiện comprehensive token validation

Token Validation Process:

* JWT signature verification (sử dụng issuer's public key)
* Token expiry check (exp claim)
* Audience validation (aud claim)
* Issuer verification (iss claim)
* Not-before check (nbf claim)
* Confirmation claim extraction (cnf claim)

**Bước 5: DPoP Proof Verification** Proxy validation DPoP proof JWT

DPoP Proof Validation:

* JWT signature verification (sử dụng client's public key từ certificate)
* HTTP method matching (htm claim)
* Target URL matching (htu claim)
* Timestamp freshness check (iat claim, typically <60 seconds)
* Unique identifier verification (jti claim - prevent replay)
* Public key confirmation (jwk claim matches certificate)

**Bước 6: Certificate-Token Binding Verification** Critical security step - xác minh cryptographic binding

Binding Verification Process:

* Extract public key từ client certificate
* Extract confirmation claim (cnf) từ access token
* Compare certificate thumbprint với cnf.x5t#S256
* Verify DPoP proof signature matches certificate private key
* Ensure temporal consistency (all components have valid timestamps)

**Bước 7-8: Backend Service Communication**

* Nếu tất cả validations pass, proxy forwards request đến backend service
* Request được enrich với verified identity information và security context
* Backend service xử lý business logic và trả về response

**Bước 9: Response Delivery**

* Proxy nhận response từ backend và thực hiện final security checks
* Audit log được ghi nhận với đầy đủ security context
* Response được gửi về client qua established mTLS channel

**Security Checkpoints Summary:**

Multi-layered Validation:

* Layer 1: mTLS Certificate Authentication (Transport)
* Layer 2: Bearer Token Authorization (Application)
* Layer 3: DPoP Proof-of-Possession (Cryptographic Binding)
* Layer 4: Certificate-Token Binding Verification (Anti-theft)
* Layer 5: Replay Prevention (Temporal + Nonce validation)

**Failure Handling:**

* Mỗi validation step failure dẫn đến immediate request rejection
* Comprehensive error logging cho security monitoring
* Rate limiting và anomaly detection cho suspicious patterns
* Automatic certificate revocation triggering nếu compromise detected

### Techstack

Proxy mTLS (FastAPI): Thành phần cốt lõi được phát triển bằng Python FastAPI với các thư viện chuyên biệt gồm (1) cryptography để xử lý chứng chỉ X.509 và các phép toán mật mã; (2) PyJWT cho việc tạo và xác minh JSON Web Tokens; (3) httpx làm HTTP client bất đồng bộ cho reverse proxy; (4) uvicorn làm máy chủ ASGI hiệu suất cao; và (5) pydantic để validation dữ liệu đầu vào. Middleware tùy chỉnh xử lý (1) trích xuất chứng chỉ client từ TLS handshake; (2) xác thực chuỗi chứng chỉ; (3) kiểm tra tình trạng thu hồi qua CRL/OCSP; và (4) thực hiện liên kết mật mã giữa chứng chỉ và token.

Dịch vụ xác thực: Được xây dựng trên FastAPI với Redis làm session store, sử dụng thư viện python-jose để tạo JWT với các claim xác nhận (cnf) liên kết đến dấu vân tay chứng chỉ client. Hỗ trợ nhiều thuật toán ký gồm (1) ECDSA P-256 cho hiệu suất cao; (2) RSA-2048 cho tương thích rộng rãi; và (3) Ed25519 cho bảo mật tối ưu và tốc độ nhanh.

Các thành phần hỗ trợ: Agent cấp chứng chỉ (CA) được thiết lập bằng OpenSSL hoặc CFSSL để tạo Root CA và Intermediate CA với khả năng tạo chứng chỉ đa thuật toán. Quản lý vòng đời chứng chỉ được tự động hóa thông qua scripts Python tích hợp với AWS Secrets Manager. Các dịch vụ backend được phát triển trên FastAPI với SQLAlchemy ORM cho PostgreSQL và redis-py để tương tác với ElastiCache.

## Phân tích kiến trúc đề xuất

### Ưu điểm của kiến trúc

Tăng cường bảo mật:

* Multi-layered Authentication: Kết hợp certificate-based và token-based authentication
* Cryptographic Binding: Tokens được bind với certificates thông qua cnf claims
* Replay Attack Prevention: DPoP proofs bao gồm timestamp, nonce, và request-specific data
* Reduced Token Theft Impact: Stolen tokens không thể sử dụng mà không có corresponding private key

Lợi ích:

* Centralized Policy Enforcement: Tại proxy layer
* Scalable Architecture: Auto Scaling Groups và Multi-AZ deployment
* Cloud-native Integration: Leverage AWS managed services
* Comprehensive Monitoring: CloudWatch, X-Ray integration

### Thách thức và hạn chế

Chi phí hiệu suất:

* Additional cryptographic operations
* Certificate validation latency
* DPoP proof generation và verification time

Độ phức tạp:

* PKI infrastructure management
* Certificate lifecycle automation
* Key rotation procedures

Triển khai:

* Library support cho DPoP
* Secure key storage requirements
* Certificate enrollment processes

## Đánh giá

Phần đánh giá sẽ tập trung vào ba câu hỏi chỉnh:

### Security Effectiveness Analysis

*"Kết hợp mTLS và token-based PoP có giảm đáng kể nguy cơ token theft/replay so với chỉ dùng bearer JWT + TLS không?"*

**Phương pháp đánh giá:**

1. **Threat modeling:**

* Định nghĩa các attack vectors chính
* So sánh attack success probability
* Phân tích blast radius khi compromise

1. **Security testing:**

* Token theft simulation
* Replay attack testing
* Man-in-the-middle testing
* Certificate validation bypass attempts

1. **Comparative analysis:**

Test scenarios:

* Bearer JWT + TLS (baseline)
* mTLS only
* Bearer JWT + mTLS
* mTLS + DPoP (proposed)

Attack vectors:

* XSS-based token theft
* Network interception
* Endpoint compromise
* Replay attacks
* Certificate spoofing

**Kỳ vọng kết quả:**

* Giảm 95%+ khả năng token theft thành công
* Loại bỏ hoàn toàn replay attacks với fresh DPoP proofs
* Tăng effort requirement cho attackers từ 1x lên 100x+

### Performance Impact Assessment

*"Overhead (latency, throughput) của việc kiểm tra mTLS + token binding ở proxy là bao nhiêu, và có thể tối ưu bằng caching/edge verification không?"*

**Phương pháp đo lường:**

1. **Benchmark setup:**

Test environment:

* EC2 t3a.small instances (2 vCPU, 2GB RAM) - Proxy
* EC2 t3a.micro instances (2 vCPU, 1GB RAM) - Backend services
* Application Load Balancer (basic configuration)
* RDS PostgreSQL (db.t3.micro, single AZ)
* ElastiCache Redis (cache.t3.micro, single node)
* CloudWatch monitoring (basic metrics)

Test parameters:

* Concurrent users: 10, 50, 100, 200 (scaled for low-end hardware)
* Request patterns: sustained, burst
* Certificate algorithms: ECDSA P-256, RSA-2048, Ed25519
* Token algorithms: ES256, RS256, EdDSA

1. **Performance Metrics:**

Latency measurements:

* TLS handshake time
* Certificate validation time
* Token verification time
* DPoP proof validation time
* Certificate-token binding check time
* End-to-end request latency

Throughput measurements:

* Requests per second (RPS)
* Concurrent connection capacity
* Error rates under load
* Resource utilization (CPU, Memory, Network)

1. **Optimization testing:**

Optimization strategies (cho low-end hardware):

* Certificate validation caching (critical for performance)
* TLS session resumption (reduce handshake overhead)
* Token verification result caching (Redis-based)
* Connection pooling (reduce connection overhead)
* Algorithm selection (Ed25519 preferred for resource efficiency)
* Request batching (reduce per-request overhead)
* Memory management (garbage collection tuning)
* Process optimization (single-threaded vs multi-threaded)

Resource constraints considerations:

* Memory: Limit concurrent connections, implement backpressure
* CPU: Prefer Ed25519, cache verification results aggressively
* Network: Connection pooling, keep-alive optimization
* Storage: Use in-memory caching, minimize disk I/O

**Kỳ vọng kết quả:**

* Base overhead: 40-60ms per request (do hardware hạn chế)
* Optimized overhead: 15-25ms per request (với caching)
* Throughput: 200-500 RPS (t3a.small), 100-200 RPS (t3a.micro)
* Resource utilization: CPU 60-80%, Memory 70-90% under moderate load

### Operational Strategy Evaluation

*"Chiến lược vận hành (CA, cert rotation, PKI automation) nào cân bằng tốt nhất giữa an toàn và vận hành đơn giản cho môi trường cloud?"*

**Phương pháp đánh giá:**

1. **PKI Strategy Comparison:**

Evaluated strategies:

* AWS Private CA + ACM
* HashiCorp Vault PKI
* Self-hosted OpenSSL CA
* Hybrid (Internal Root + Cloud Issuing)
* Public CA (Let's Encrypt, DigiCert)

Evaluation Criteria:

* Setup complexity (1-10 scale)
* Operational overhead (person-hours/month)
* Security posture (risk assessment)
* Cost analysis ($/month)
* Automation capabilities
* Disaster recovery readiness

1. **Certificate Lifecycle Management:**

Testing Scenarios:

* Certificate issuance automation
* Rotation procedures (30, 60, 90-day cycles)
* Emergency revocation
* Bulk certificate management
* Cross-environment certificate sync
* Compliance reporting

1. **Automation assessment:**

Automation levels:

* Full automation (ACME protocol)
* Semi-automated (Terraform + approval workflows)
* Manual with tooling support
* Fully manual processes

Success metrics:

* Mean Time to Certificate Issuance (MTTCI)
* Certificate expiry incidents
* Rotation success rate
* Operational cost per certificate
* Security incident correlation

**Kỳ vọng kết quả:**

* AWS Private CA + automated rotation: 85% giảm operational overhead
* 90-day rotation cycle: optimal balance security vs. operations
* Infrastructure-as-Code: 70% giảm configuration errors
* ACME protocol integration: 95% automation rate

### Comparative Algorithm Analysis

Một phần quan trọng của evaluation là so sánh hiệu suất các thuật toán mã hóa:

Algorithm Performance Matrix (trên t3a.micro/small):

|  |  |  |  |
| --- | --- | --- | --- |
|  | ECDSA P-256 | RSA-2048 | Ed25519 |
| Certificate Gen | ?ms | ?ms | ?ms |
| Cert Validation | ?ms | ?ms | ?ms |
| Token Signing | ?ms | ?ms | ?ms |
| Token Verify | ?ms | ?ms | ?ms |
| Memory Usage | ? | ? | ? |
| CPU Usage | ? | ? | ? |
| Key Size | 256-bit | 2048-bit | 255-bit |
| Security Level | ~128-bit | ~112-bit | ~128-bit |

Performance trên low-end hardware:

* Ed25519: Fastest overall, least resource intensive
* ECDSA P-256: Good balance, moderate resource usage
* RSA-2048: Slowest, highest resource consumption
* Bottleneck: CPU and memory constraints more pronounced

### Demo Implementation

Để xác thực các findings, sẽ xây dựng một ứng dụng demo:

**Banking API Simulation:**

Demo scenarios:

* Customer authentication via mTLS
* Account balance queries
* Fund transfers với DPoP tokens
* Transaction history retrieval
* Real-time performance monitoring
* Security event demonstration

Interactive Features:

* Algorithm switching (live comparison)
* Attack simulation (token theft, replay)
* Performance dashboard
* Certificate management interface
* Audit log visualization

**Performance Visualization:**

* Real-time latency graphs
* Throughput comparison charts
* Resource utilization monitoring
* Security event correlation
* Cost analysis dashboard

# Kết luận

Nghiên cứu này đề xuất một kiến trúc Zero-Trust API authentication kết hợp mTLS và token-based signatures nhằm tăng cường bảo mật cho API trong môi trường cloud. Kiến trúc được thiết kế để giải quyết các hạn chế của bearer tokens truyền thống thông qua việc tạo ra cryptographic binding giữa client certificates và access tokens.

## Đóng góp chính

* **Kiến trúc tích hợp**: Kết hợp mTLS với DPoP tokens tạo ra multi-layered authentication với strong cryptographic binding
* **Performance analysis**: So sánh comprehensive các thuật toán mã hóa (ECDSA, RSA, Ed25519) trong production environment
* **Operational strategies**: Đề xuất các chiến lược PKI automation phù hợp với cloud-native environments
* **Implementation roadmap**: Cung cấp hướng dẫn chi tiết để triển khai trên AWS với Terraform

## Kỳ vọng về tác động

**Về mặt bảo mật:**

* Giảm đáng kể nguy cơ token theft và replay attacks
* Tăng nỗ lực yêu cầu cho attackers
* Cung cấp audit trails một cách toàn diện

**Về mặt hiệu suất:**

* Đề xuất chiến lược tối ưu hiệu quả
* Cân bằng giữa bảo mật và hiệu suất

**Về mặt vận hành:**

* Quản lý PKI dễ dàng thông qua các dịch vụ cloud
* Quản lý certificate lifecycle tự động
* Làm giảm chi phí vận hành và lỗi từ con người

## Hướng đi trong tương lai

* Post-quantum cryptography: Tích hợp các thuật toán quantum-resistant
* Edge computing: Mở rộng mô hình cho edge/IoT environments
* Machine learning: Sử dụng ML cho anomaly detection và authentication thích ứng
* Cross-cloud integration: Multi-cloud PKI federation và trust models

Nghiên cứu này sẽ cung cấp foundation solid cho việc triển khai Zero-Trust API security trong môi trường production, với focus đặc biệt vào cân nhắc thực tế như hiệu suất, vận hành, và tối ưu chi phí.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | I. J. a. P.-A. J. a. M. M. a. X. B. a. W.-F. D. a. O. S. a. C. A. a. B. Y. Goodfellow, "Generative adversarial nets," *Advances in neural information processing systems,* vol. 27, 2014. |
| [2] | R. Gall, "Dzone.com," 25 9 2018. [Online]. Available: https://dzone.com/articles/working-principles-of-generative-adversarial-netwo. [Accessed 20 4 2025]. |
| [3] | Fonseca, Eduardo and Plakal, Manoj and Font, Frederic and Ellis, Daniel PW and Favory, Xavier and Pons, Jordi and Serra, Xavier, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902,* 2018. |
| [4] | Ye, Dengpan and Jiang, Shunzhi and Huang, Jiaqin, "Heard more than heard: An audio steganography method based on gan," *arXiv preprint arXiv:1907.04986,* 2019. |
| [5] | Binny, Anu and Koilakuntla, Maddulety, "Hiding secret information using LSB based audio steganography," in *2014 International Conference on Soft Computing and Machine Intelligence*, 2014. |
| [6] | Cho, Kiho and Bae, Soo Hyun and Choi, In Kyu and Kim, Nam Soo and Unoki, Masashi, "Robust Audio Data Hiding Method Based on Phase of Modulated Complex Lapped Transform," in *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IEEE, 2013, pp. 263--266. |
| [7] | Gupta, Sumeet and Dhanda, Namrata, "Audio steganography using discrete wavelet transformation (DWT) \& discrete cosine transformation (DCT)," *IOSR Journal of Computer Engineering,* vol. 17, pp. 32--44, 2015. |
| [8] | Putra, Arief Bramanto Wicaksono and Jufri, Muhammad Taher and Lantara, Dirgahayu and Assyauqi, Anugrah and Wajiansyah, Agusma and Pranolo, Andri, "Steganography for data hiding in digital audio data using combined least significant bit and 4-wrap length method," in *2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, IEEE, 2018, pp. 336--339. |
| [9] | Tayel, Mazhar and Gamal, Ahmed and Shawky, Hamed, "A proposed implementation method of an audio steganography technique," in *2016 18th international conference on advanced communication technology (ICACT)*, IEEE, 2016, pp. 180--184. |
| [10] | Tran, Dang Ninh and Zepernick, Hans-J{\"u}rgen and Chu, Thi My Chinh, "Lsb data hiding in digital media: a survey," in *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, European Alliance for Innovation (EAI), 2022, pp. 1--50. |
| [11] | Balgurgi, Pooja P and Jagtap, Sonal K, "Intelligent processing: An approach of audio steganography," in *2012 International Conference on Communication, Information \& Computing Technology (ICCICT)*, IEEE, 2012, pp. 1--6. |
| [12] | Jain, Miss Preeti and Trivedi, PV, "Effective audio steganography by using coefficient comparison in DCT domain," *International Journal of Engineering Research \& Technology (IJERT),* vol. 2, 2013. |
| [13] | R. Singh, "A Step-by-Step Guide to Speech Recognition and Audio Signal Processing in Python," 14 6 2021. [Online]. Available: https://towardsdatascience.com/a-step-by-step-guide-to-speech-recognition-and-audio-signal-processing-in-python-136e37236c24/. [Accessed 25 4 2025]. |
| [14] | N. V. Otten, "Wavelet Transform Made Simple [Foundation, Applications, Advantages]," 20 9 2024. [Online]. Available: https://spotintelligence.com/2024/09/20/wavelet-transform/. [Accessed 25 04 2025]. |