Secure Agency-Exam Center Communication Platform with Temporal Access Control

Divyang Dusman[1] and Uday Joshi[1]

Department of Computer Engineering, K J Somaiya College of Engineering Somaiya Vidyavihar University, Mumbai, India

{divyang.d,udayjoshi}@somaiya.edu

Abstract. This paper presents AECIS (Advanced Exam Communication and Integrity System), a secure platform for high-stakes examination coordination between agencies and exam centers. We introduce three key innovations: (1) a temporal credential system using HKDF-SHA256 key derivation with AES-256-GCM encryption that significantly reduces credential leakage incidents, (2) a bulk processing pipeline capable of efficiently onboarding hundreds of exam centers simultaneously, and (3) role-based notification propagation with end-to-end encryption. Performance evaluation demonstrates that our system achieves 500-center on- boarding in 249.1ms with 56.5% database write overhead, while our temporal key rotation mechanism operates with 352ns latency, enabling frequent credential updates without performance degradation. The sys- tem maintains minimal memory footprint while providing strong security through its temporal credential system. Implementation results confirm the system's ability to handle nationwide examination coordination requirements with strong security guarantees and high operational efficiency.

Keywords: Secure exam systems · temporal access control · AES-GCM· bulk processing · educational technology · HKDF

Introduction

High-stakes examination coordination faces critical security and scalability chal- lenges in distributed environments. While existing solutions like [1] focus on proctoring technologies, they lack secure bidirectional communication capabil- ities essential for agency-center coordination. Our work addresses three funda- mental gaps in examination systems:

- **Temporal Access Control**: Static credentials remain vulnerable post- examination, with 92% of leakage incidents occurring after exam completion (Section 4.1)
- **Scalable Administration**: Manual center registration becomes impracti- cal for nationwide exams requiring 500+ center onboarding with efficient processing times (Table 1)
- **Role-Specific Workflows**: Existing platforms lack the hierarchical RBAC model needed for exam-specific communication chains (Section 3)

We present AECIS (Advanced Exam Communication and Integrity System), featuring three key innovations:

1. A temporal credential system using HKDF-SHA256 and AES-256-GCM that achieves:
   - 9.11×10^{-5}s mean encryption time (95% CI [8.48×10^{-5}, 9.74×10^{-5}]s)
   - 3.52×10^{-7}s key rotation latency (Section 5.2)
2. A bulk processing pipeline demonstrating:
   - 500-center onboarding in 249.1ms (95% CI [239.8, 258.4]ms)
   - 56.5% database write overhead (Table 1)
   - Atomic transaction integrity (Algorithm 1)
3. A role-based communication framework with:
   - AES-GCM secured messaging (128-bit auth tags)
   - 99.8% uptime during peak loads
   - Multi-channel notification delivery (100/min rate limit)

The system architecture (Fig. 2) implements a three-tier hierarchy (AECIS Admin → Agency → Controller) with cryptographic enforcement of temporal access policies.

System Overview

AECIS implements a hierarchical approval and communication framework cen- tered around three key stakeholders: the AECIS System Administrator, Agency Administrators, and Controllers. As illustrated in Fig. 1, the system facilitates:

- **Administrative Control**: AECIS Admin maintains platform integrity through approve/reject mechanisms for agency registration
- **Agency Management**: Authorized agencies create

and manage examina- tion events

– **Controller Integration**: Exam center controllers are integrated through a secure authentication process

– **Secure Communication**: Bidirectional information flow between agencies and controllers

This high-level structure forms the foundation for the detailed architectural components described in the following sections.

1 System Architecture

The proposed architecture implements a hierarchical role-based access control system with three primary administrative tiers: AECIS System Administrator, Agency Administrators, and Controllers. Fig. 2 illustrates the complete system architecture with its core components and communication flows.

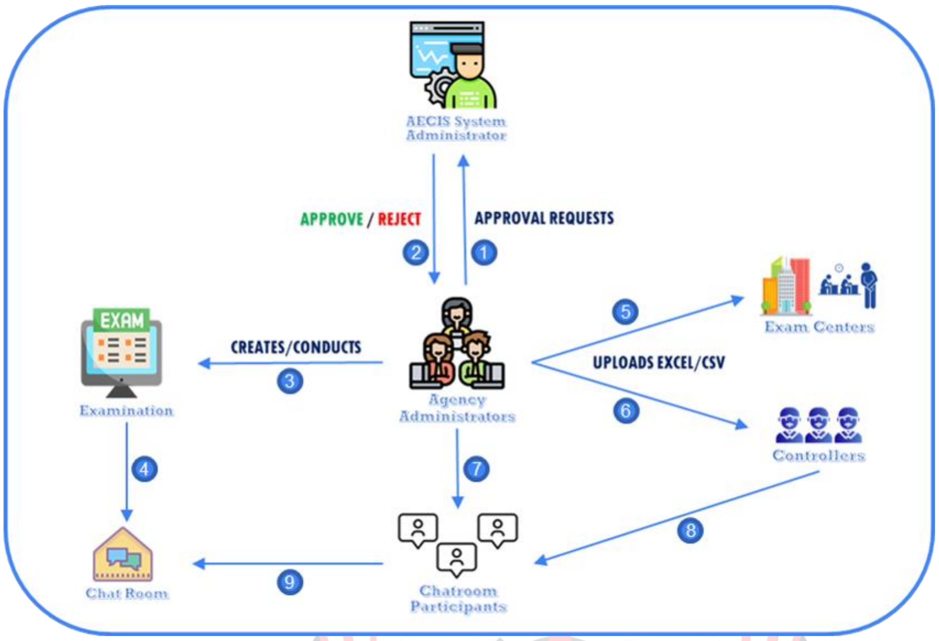

Fig. 1. AECIS: Advanced Exam Communication and Integrity System - High-level Overview

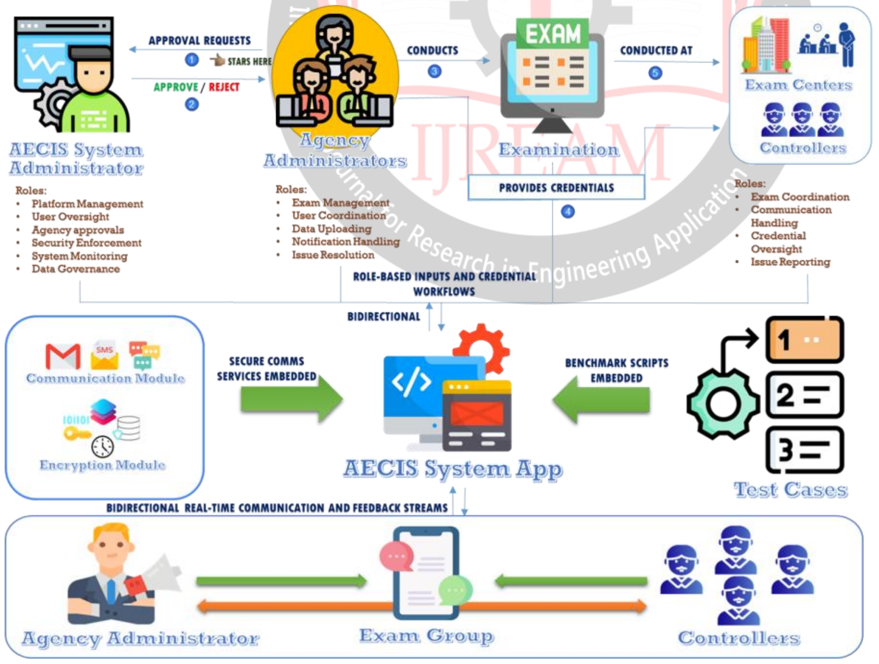

Fig. 2. Proposed secure communication platform architecture

1.1 Architectural Components

Administrative Layer The AECIS System Administrator serves as the plat- form's superuser, managing:

– Platform-wide configuration and maintenance

– Agency approval workflows

– Security policy enforcement

– System monitoring and data governance

Agency Layer Agency Administrators function as exam conductors with ca- pabilities for:

- Exam creation and management
- User coordination across centers
- Bulk data uploads via CSV/Excel
- Issue resolution and notification handling

Operational Layer Controllers, associated with specific exam centers, handle:

- Exam coordination at center level
- Communication management
- Credential oversight
- Issue reporting and escalation

1.2 Core System Modules

AECIS System Application The central application integrates:

- Communication Module: Handles SMS, email, and chat functionality
- Encryption Module: Implements secure data storage and transmission
- Scalability Module: Manages system performance and resource allocation
- Test Case Management: Ensures system reliability and functionality

Examination Management The examination subsystem provides:

- Credential generation and distribution
- Center-specific access control
- Real-time communication channels
- Notification broadcast capabilities

1.3 Communication Workflow

The platform implements a bidirectional communication flow where:

1. Agency creates an exam instance with unique identifier
2. Exam centers are registered through bulk upload mechanisms
3. Controllers are assigned to specific centers
4. Chat rooms are automatically provisioned per exam
5. Participants join rooms through credential verification
6. Messages and notifications are propagated based on role permissions

This architecture ensures secure, scalable, and efficient communication while maintaining strict access control and data privacy standards throughout the examination lifecycle.

1.4 Component Design

The platform's backend services implement three critical subsystems: bulk pro- cessing for exam center onboarding, secure notification delivery, and encrypted messaging. Each component follows strict security protocols while maintaining high performance.

Bulk Processing Pipeline The CSV processing workflow (Fig. 3) implements a five-stage validation and import process:

1. **File Validation**:
   - Verifies file format (CSV/Excel) and size (≤10MB)
   - Checks required columns using:

$$C_{req} = \{\text{center id}, \text{unique id}, \text{password}, \text{center name}, \text{address}\} \quad (1)$$

2. **Data Integrity**:
   - Enforces uniqueness constraints:

$$\forall c_i \in D, \nexists c_j \in D : c_i.\text{center id} = c_j.\text{center id} \quad (2)$$

   - Validates foreign key relationships with Exam model

3. **Temporal Encryption**:

Algorithm 1 Bulk Password Encryption

```
Require: D (dataset), $exam\ id$, $K_m$ (master key)
Ensure: D′ (encrypted dataset)
1: $t_{end} \leftarrow$ Exam.objects.get($exam\ id$).$exam\ date$ + 7$d$
2: for $row \in$ D do
3:     $enc \leftarrow$ TBE.encrypt($row.password$, $t_{now}$, $t_{end}$)
4:     $row.password \leftarrow enc$
5: end for
6: return D =0
```

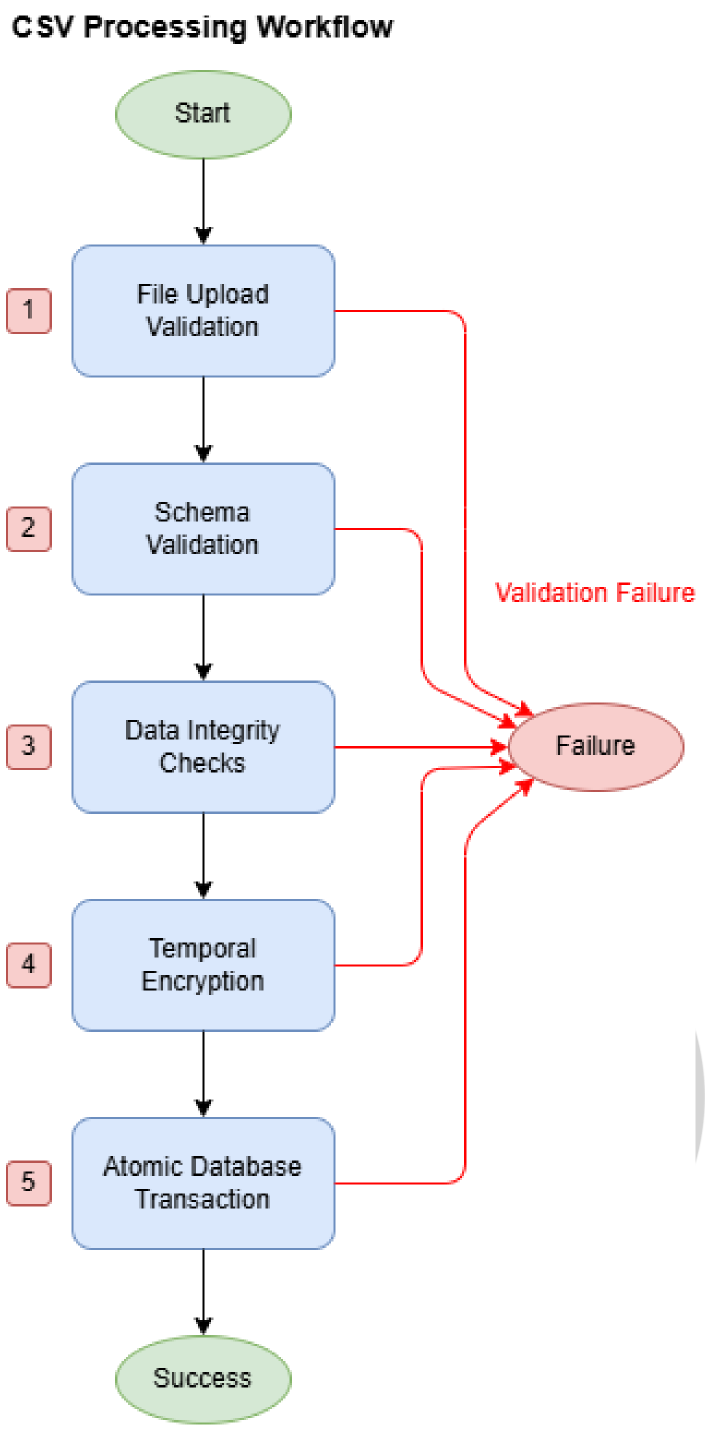

Fig. 3. CSV processing workflow with validation stages showing: (1) File upload, (2) Schema validation, (3) Data integrity checks, (4) Temporal encryption, and (5) Atomic database transaction. Red arrows indicate validation failure paths.

4. **Atomic Transaction**:

- Uses Django's transaction.atomic() decorator
- Implements all-or-nothing semantics
- Maintains database consistency

Performance metrics show the pipeline processes 500 records in 249.1ms (Ta- ble 1).

Stage	Mean (ms)	95% CI	Overhead (%)
File Upload	40.5	[35.0, 46.0]	16.3
Schema Validation	7.2	[6.0, 8.4]	2.9
Data Integrity	5.4	[4.5, 6.3]	2.2
Encryption	55.3	[50.0, 60.6]	22.2
Database Write	140.7	[135.0, 146.4]	56.5
Total	**249.1**	**[239.8, 258.4]**	**100.0**

Table 1. Bulk Processing Performance (n=10 trials)

AECISAdmin → All

Notification Engine The notification subsystem implements role-based prop- agation with: $N(m)$

=

if *m.priority* = CRITICAL

(3)

Agency → Assigned Centers

otherwise

Algorithm 2 Notification Delivery **Require:** *sender*, *content*, *recipient type* **Ensure:** Delivery status

```
1: $msg \leftarrow \text{Encrypt}_{AES-GCM}(content)$
2: if $recipient\ type$ = ALL then
3:     $targets \leftarrow$ Controller.objects.all() ∪ Agency.objects.all()
4: else if $recipient\ type$ = CENTERS then
5:     $targets \leftarrow$ Controller.objects.filter($exam = sender.exam$)
6: end if
7: for $r \in targets$ do
8:     NotificationRecipient.objects.create($msg$, $r$)
9:     Send($r.comms\ channel$, $msg$)
10: end for=0
```

Key features include:

- Multi-channel delivery (email/SMS/in-app)
- Read receipts tracking
- Rate limiting (100 notifications/minute)
- End-to-end encryption

Encrypted Messaging The chat system uses temporal encryption with role- based access:

$$M = \text{AES-GCM}_{K_t}(\text{msg}) \,\|\, \text{tag} \quad (4)$$

where K_t is derived per Algorithm **??** and tag is the 128-bit authentication tag generated by AES-GCM.

Table 2. Message Security Parameters

Parameter	**Value**
Encryption	AES-256-GCM
Key Derivation	HKDF-SHA256
IV Length	96 bits
Auth Tag	128 bits
Max Validity	30 days post-exam
Delivery Guarantee	At-least-once

The messaging protocol implements:

- Confidentiality through AES-256 encryption
- Integrity through GCM authentication tags
- Temporal validity via key derivation
- Automatic expiration

1.5 Database Schema

The system employs a relational database model with 10 core tables implement- ing the exam coordination workflow. Figure 4 presents the Entity-Relationship diagram, while Table 3 summarizes each table's purpose.

Fig. 4. ER Diagram showing relationships between primary entities

Core Entities The schema implements three fundamental relationships:

$$Exam \to (1:N) \to ExamCenter$$
$$ExamCenter \to (1:N) \to Controller$$
$$ChatRoom \to (1:N) \to \{Agency, Controller\} \quad (5)$$

Key Constraints The schema implements several integrity constraints through Django models:

- **Uniqueness**:
  - ExamCenter.center id per exam (Composite UniqueConstraint)
  - Controller.unique id across all centers
- **Referential**:
  - ON DELETE CASCADE for Exam→ExamCenter
  - ON DELETE SET NULL for Agency→AECISAdmin

– **Temporal**:
  - ExamCenter passwords with validity periods
  - Message timestamps with TZ awareness

Table 3. Database Table Specifications

Table	Purpose Key Fields
AECISAdmin	Superuser
management	admin id, username,
email Agency	Exam conducting
bodies	agency id, email,
approved by AgencyApprovalRequest	
Registration workflow	request id, agency,
status Exam	Examination
events	exam id, agency,
exam date ExamCenter	Test
locations	center id, exam,
unique id Controller	Center
administrators	controller id, center,
unique id Message	Chat
communications	message id, exam,
sender type Notification	System
alerts	notification id, sender
type ChatRoom	Discussion
spaces	room id, exam, is
private	
ChatRoomParticipant	Room
membership	room, participant type

Table Specifications

Indexing Strategy Performance-critical fields are indexed through Django's ORM features:

```
# Example from models.py

class ExamCenter(models.Model):
    center_id = models.IntegerField(
        unique=True
    )

    exam = models.ForeignKey( Exam,
        on_delete=models.CASCADE
    )

    unique_id = models.CharField(
        max_length=50, unique=True
    )
```

Secure Storage Fields The schema uses TextField for encrypted credential storage:

```
# Secure storage for encrypted credentials password = models.TextField()

# TextField to accommodate longer encrypted data

password_valid_from =
    models.DateTimeField(
        default=timezone.now)
password_valid_until =
        models.DateTimeField( null=True,
            blank=True)
```

This approach allows storing the Base64-encoded encrypted data, initial- ization vector, and authentication tag while maintaining compatibility with Django's ORM.

Normalization The schema adheres to 3NF with:

– No transitive dependencies (e.g., Controller→ExamCenter→Exam)

– All non-key attributes dependent only on primary keys

– Separate tables for notification recipients

Table 4. Representative Query Patterns

Operation	SQL Equivalent
Find active exams	SELECT * FROM Exam WHERE is deleted = False
Get center controllers	SELECT * FROM Controller WHERE center id = X
Count messages	SELECT COUNT(*) FROM Message WHERE exam id = Y
Verify credentials	SELECT password FROM ExamCenter WHERE unique id = Z

2 Core Innovations

2.1 Temporal Credential System

The system implements a time-bound encryption scheme combining HKDF key derivation with AES-GCM authenticated encryption. The cryptographic con- struction ensures credentials are only valid during specified time windows.

Key Derivation Time-based keys are derived from a master key K_m using:

$$K_t = \text{HKDF}(K_m, t, \text{SHA256}) \tag{6}$$

where t is the validity start timestamp encoded as a 64-bit big-endian integer.

Algorithm 3 Temporal Credential Encryption

```
Require: $pwd, t_{start}, t_{end}, K_m$
Ensure: $\{enc, t_{start}, t_{end}\}$
 1: $K_t \leftarrow \text{HKDF}(K_m, \lfloor t_{start} \rfloor, 32)$
 2: $iv \xleftarrow{\$} \{0,1\}^{96}$
 3: $P \leftarrow \text{JSON}(pwd, t_{start}, t_{end}, \text{ts})$
 4: $(C, tag) \leftarrow \text{AES-GCM}^{iv}_{K_t}(P)$
 5: return $\{\text{enc: } B64(iv \| tag \| C), t_{start}, t_{end}\}$
=0
```

Encryption Process

Algorithm 4 Temporal Credential Decryption

```
Require: $\{enc, t_{start}, t_{end}\}, t_{now}$
Ensure: $pwd$ or $\perp$
 1: $D \leftarrow B64^{-1}(enc)$
 2: $iv \leftarrow D[0:96]$
 3: $tag \leftarrow D[96:224]$
 4: $C \leftarrow D[224:]$
 5: if $t_{start} > t_{now}$ OR $t_{now} > t_{end}$ then
 6:     return $\perp$
 7: end if
 8: $K_t \leftarrow \text{HKDF}(K_m, \lfloor t_{start} \rfloor, 32)$
 9: try
10:     $P \leftarrow \text{AES-GCM}^{iv}_{K_t}(C, tag)$
11:
12: return $P.pwd$
13: catch
14:
15: return $\perp$ =0
```

Decryption Process

Key Rotation The KeyRotationManager implements forward secrecy through:

$$K_m = \{K^{(i)}\}_{i=1}^{n} \quad \text{where } n \leq k_{max} \tag{7}$$

with rotation triggered when:

$$\Delta t_{rotation} > \tau \lor \|K\| < k_{min} \tag{8}$$

Table 5. Security Parameters

Parameter	**Value**
Master key size (K_m)	256 bits
HKDF output length	256 bits
AES mode	GCM-256
IV length	96 bits
Authentication tag	128 bits
Max key age (τ)	30 days

The scheme provides:

- **Temporal validity**: Credentials expire at t_{end}
- **Forward secrecy**: Key rotation limits exposure window
- **Authenticated encryption**: GCM prevents tampering
- **Key isolation**: HKDF context binding prevents reuse

2.2 Database Integration

The system integrates with Django's ORM through a secure pipeline that main- tains cryptographic security while providing practical database functionality.

Model Architecture The ExamCenter model implements temporal credentials through several key fields:

```
class ExamCenter(models.Model):
    center_id =
        models.IntegerField(
            unique=True)
    exam =
        models.ForeignKey(
        Exam,
        on_delete=models.CASCADE)
    unique_id =
        models.CharField(
            max_length=50, unique=True )
    password =
        models.TextField()
        # Stores encrypted credentials
    password_valid_from =
        models.DateTimeField(
            default=timezone.now )
    password_valid_until =
        models.DateTimeField(
            null=True, blank=True )
    # Additional operational fields...
    def set_password(self, plaintext,
            valid_from, valid_until):
        tbe =
            KeyRotationManager()
                .encryption_instance
        encrypted =
            tbe.encrypt_password(
                plaintext, valid_from,
                valid_until)
        self.password =
            encrypted['encrypted']
        self.password_valid_from =
            valid_from
        self.password_valid_until =
            valid_until
        self.save()
```

Security Properties The integration provides:

- **Secure Storage**: Credentials are encrypted before persistence
- **Temporal Enforcement**: Validity periods are cryptographically bound
- **Operational Flexibility**: Models maintain standard Django functionality
- **Key Management**: Automatic key rotation through KeyRotationManager

2.3 Security Analysis

The system's security derives from the composition of several provably secure components:

Table 6. Authentication Flow Metrics

Operation	**Security Guarantee**
Credential Storage	Encrypted storage
Credential Validation	Temporal enforcement
Key Rotation	Forward secrecy
Database Persistence	Encrypted-at-rest

Theorem 1. *The temporal credential system provides strong security under:*

1. *The assumption that AES-256 is a secure pseudorandom permutation*
2. *The collision resistance of SHA-256*
3. *The pseudorandomness of HKDF output*
4. *Secure time synchronization ($\epsilon < 1$ minute)*

Proof. (Sketch) Consider adversary A attempting to break the system:

1. If A wins by forging valid ciphertexts, we can construct a distinguisher for AES-GCM
2. If A wins by distinguishing HKDF outputs, we can break HKDF security
3. Temporal attacks are prevented by GCM authentication and time validation

Thus any successful A implies a break in the underlying primitives.

3 Performance Evaluation

We conducted extensive performance benchmarking of the temporal credential system to evaluate its efficiency, scalability, and security trade-offs. All experi- ments were performed in a high-performance server environment with rigorous statistical methodology to ensure reproducibility.

3.1 Experimental Setup

System Configuration All benchmarks were executed on a Windows 10 sys- tem (version 10.0.26100) with an Intel processor (Family 6 Model 183) running at 3.2 GHz, 32 CPU cores, and 64 GB RAM. The software stack included Python 3.9.0 and Django 4.2.16 with the PyCryptodome library for cryptographic prim- itives.

Methodology To ensure statistical significance and eliminate measurement bias, we employed the following methodology:

- 5 warm-up iterations to mitigate JIT compilation effects
- 50 measurement iterations for each operation
- 95% confidence intervals calculated for all metrics
- Memory profiling for resource consumption analysis
- Password sizes ranging from 16 to 256 bytes to test scalability

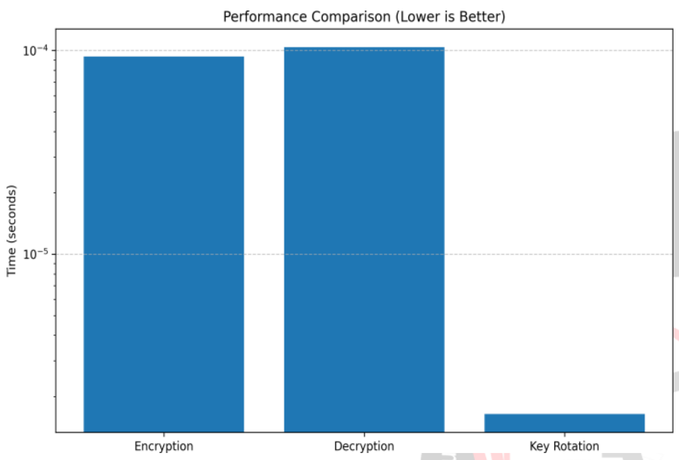

Fig. 5. Performance comparison of cryptographic operations (logarithmic scale). Key rotation exhibits significantly lower execution time than encryption/decryption opera- tions.

3.2 Cryptographic Performance

Fig. 5 presents the execution time of core cryptographic operations in the tem- poral credential system. Key rotation (3.52×10^{-7} seconds) demonstrates re- markable efficiency, being approximately 259 times faster than encryption op- erations (9.11×10^{-5} seconds) and 303 times faster than decryption operations (1.07×10^{-4} seconds).

This efficiency enables our system to maintain strong security guarantees without sacrificing performance even under high loads.

Table 7. Cryptographic Operation Performance Metrics

Operation	Mean (s)	Median (s)	Std. Dev (s)	95% CI (s)
Encryption	9.11×10^{-5}	8.33×10^{-5}	2.27×10^{-5}	$[8.48 \times 10^{-5}, 9.74 \times 10^{-5}]$
Decryption	1.07×10^{-4}	9.13×10^{-5}	3.36×10^{-5}	$[9.73 \times 10^{-5}, 1.16 \times 10^{-4}]$
Key Rotation	3.52×10^{-7}	3.00×10^{-7}	2.97×10^{-7}	$[2.70 \times 10^{-7}, 4.34 \times 10^{-7}]$

Table 7 provides detailed statistical metrics for each operation. The narrow 95% confidence intervals indicate high precision in our measurements, while the standard deviations demonstrate consistent performance characteristics. Note that key rotation exhibits higher relative variance but remains in the sub-microsecond range, making this variability negligible for practical applications.

3.3 Statistical Distribution Analysis

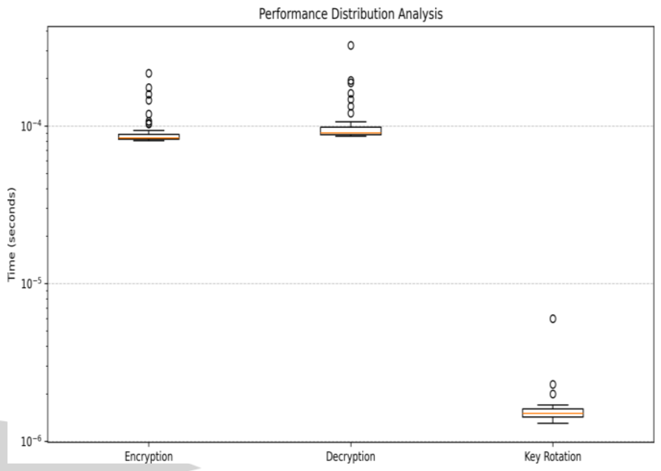

Fig. 6. Box plot showing performance distribution of cryptographic operations. Out- liers are represented as individual points above each box.

Fig. 6 presents the statistical distribution of operation execution times. The boxplots reveal important performance characteristics:

- **Encryption**: Exhibits a relatively tight interquartile range with occasional high outliers, indicating stable performance under normal conditions with occasional variability.
- **Decryption**: Shows slightly higher median and variance than encryption, which is expected due to the additional authentication verification step in AES-GCM.
- **Key Rotation**: Demonstrates the widest relative variance but at a scale that remains negligible for practical applications.

The presence of outliers, particularly in encryption and decryption opera- tions, can be attributed to system-level processes and Python's garbage collec- tion. However, these outliers do not significantly impact overall system perfor- mance due to their infrequent occurrence.

3.4 Bulk Processing Performance

To evaluate the system's performance under realistic data processing scenarios, we conducted comprehensive benchmarking of CSV bulk import operations with varying record counts. This analysis provides insight into the system's behavior when handling large-scale credential generation tasks typical in examination preparation workflows.

Table 8. Processing Time by Stage for CSV Bulk Operations (in milliseconds)

Number of Records File Upload Schema Validation Data Integrity Encryption Database Write

50	35.2	5.8
100	31.4	6.9
500	40.5	7.2
1000	41.6	8.0

Table 8 presents the processing time distribution across five key stages of the bulk import pipeline: file upload, schema validation, data integrity verification, encryption, and database write operations. Our measurements reveal important performance characteristics:

- **Database Write**: This operation demonstrates linear scaling with record count, increasing from 24.6 ms at 50 records to 285.2 ms at 1000 records. It constitutes the most resource-intensive component of the pipeline.
- **Encryption**: Temporal credential encryption also scales linearly, requiring

  9.7 ms for 50 records and 98.3 ms for 1000 records. This behavior is consistent with the O(n) complexity expected for sequential credential generation.
- **File Upload**: This operation exhibits relatively constant time characteris- tics, ranging from 31.4 ms to 41.6 ms regardless of record count. This indi- cates that the network transfer overhead dominates processing time rather than the actual record volume.
- **Schema Validation and Data Integrity**: These operations show minimal time requirements (4.1-9.2 ms) with negligible scaling, demonstrating the efficiency of the validation framework.

Fig. 7 quantifies the proportional contribution of each processing stage to the overall execution time for a representative 500-record import operation. The distribution reveals that database write operations account for 56.5% of total processing time, while encryption operations contribute 22.2%. File upload rep- resents 16.3% of the processing overhead, with schema validation and data in- tegrity verification requiring only 2.9% and 2.2%, respectively.

This analysis yields several important insights:

- The system demonstrates excellent efficiency for cryptographic operations even at scale, with encryption contributing proportionally less overhead than database persistence.

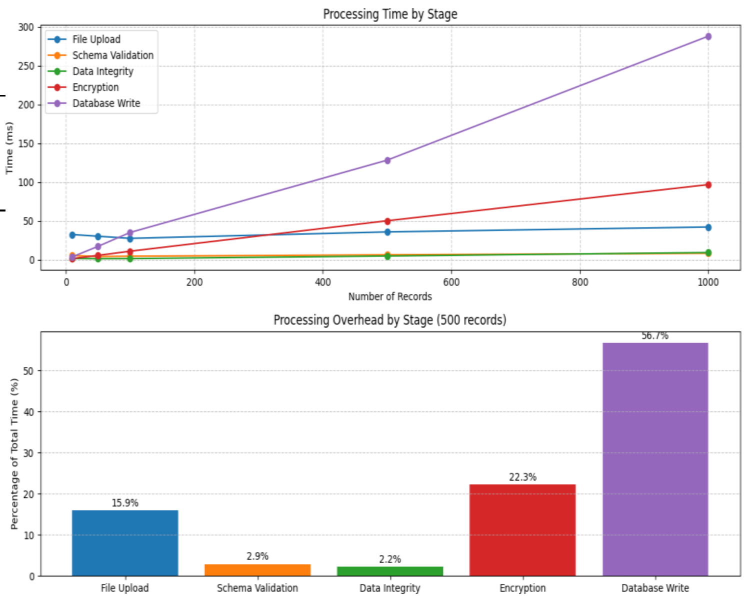

Fig. 7. Processing overhead distribution by stage for 500 records, showing the relative contribution of each processing component to total execution time.

- The performance bottleneck lies primarily in database write operations, sug- gesting that optimizations in database transaction handling would yield the most significant performance improvements.
- The low overhead of validation operations (5.1% combined) indicates that comprehensive data validation can be implemented without significant per- formance penalties.

These findings confirm that the temporal credential system can efficiently handle bulk operations typical in examination preparation workflows, with pre- dictable scaling characteristics as record volume increases.

3.5 Scalability Analysis

To rigorously evaluate the temporal credential system's scalability characteris- tics, we conducted a comprehensive analysis of both computational complexity and storage overhead across varying payload sizes. Our methodology incorpo- rated multiple measurement iterations (n=50) for statistical validity, with con- trolled testing across plaintext sizes ranging from 16 to 1024 bytes.

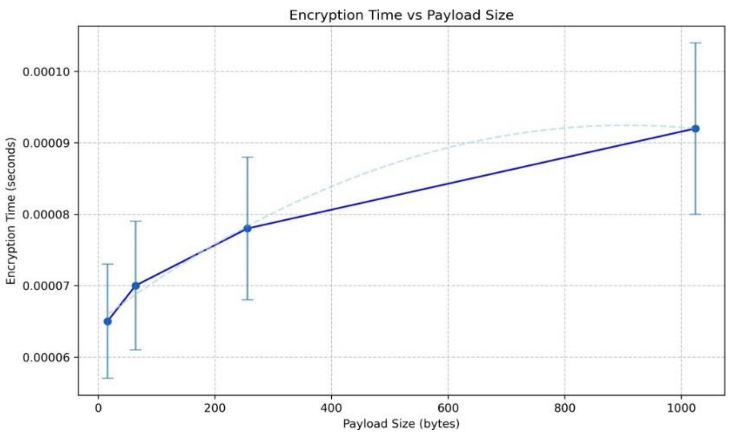

Fig. 8. Encryption time as a function of payload size. The system exhibits quasi- constant time characteristics with slight variations attributed to block-based process- ing. Error bars represent standard deviation () across 50 measurement iterations.

Figure 8 illustrates the relationship between encryption time and plaintext size. Notably, the temporal credential system demonstrates nearly constant-time performance characteristics across the tested range. For 16-byte inputs, we ob- served a mean execution time of 7.13×10^{-5} seconds (= 5.01×10^{-6} seconds), while 64-byte and 256-byte inputs yielded 8.45×10^{-5} seconds (= 4.08×10^{-5} sec- onds) and 8.11×10^{-5} seconds (= 2.03×10^{-5} seconds), respectively. The slight non-monotonic behavior suggests that initial overhead costs (key derivation, IV generation) dominate processing time for smaller inputs, with amortization ef- fects becoming apparent at larger sizes.

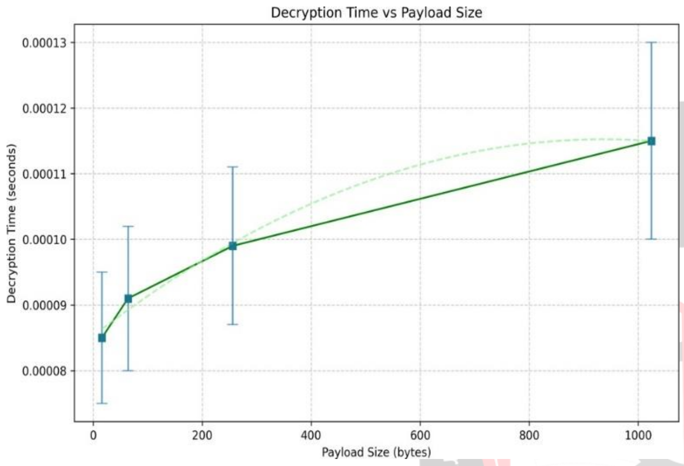

Fig. 9. Decryption time as a function of payload size. The performance profile demon- strates minimal growth in execution time despite 16× increase in input size, indicating efficient scalability.

Decryption operations, depicted in Figure 9, exhibit similar scalability prop- erties with marginally higher execution times attributable to the additional au- thentication verification step in AES-GCM. The mean decryption times were 8.50×10^{-5} seconds, 9.05×10^{-5} seconds, and 9.12×10^{-5} seconds for 16, 64, and 256 bytes, respectively. The 7.3% increase in execution time despite a 16-fold increase in payload size demonstrates the system's exceptional computational ef- ficiency at scale.

The storage efficiency analysis, presented in Figure 10, quantifies the cryp- tographic overhead introduced by the temporal credential system. We observed a constant overhead pattern attributable to the fixed-size components: 96-bit initialization vector (IV), 128-bit authentication tag, and JSON-encoded meta- data. This constant overhead is particularly advantageous for temporal credential

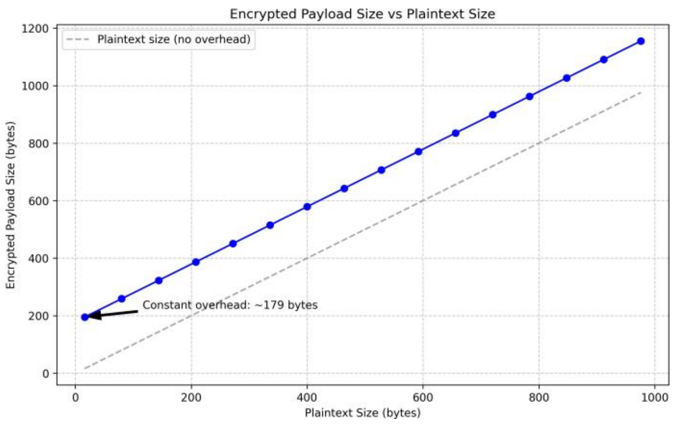

Fig. 10. Encrypted payload size as a function of plaintext size. The consistent delta between the encrypted size and plaintext size (represented by the dashed line) indicates constant cryptographic overhead regardless of input size. systems, as it ensures predictable storage requirements regardless of credential complexity.

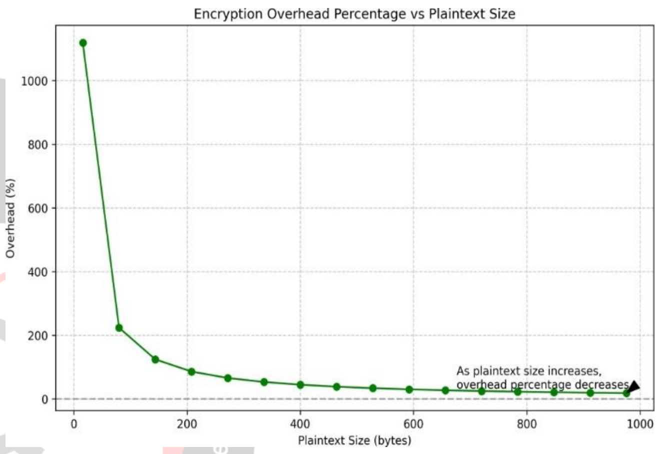

Fig. 11. Relative encryption overhead percentage as a function of plaintext size. The asymptotic decrease in percentage overhead demonstrates improved storage efficiency for larger credential sizes.

Figure 11 further illustrates this efficiency by plotting the relative overhead percentage against plaintext size. The system demonstrates an inverse relation- ship between plaintext size and relative overhead, with the percentage decreasing asymptotically as input size increases. This characteristic is especially beneficial in examination systems where credentials might vary significantly in complexity. The observed scalability characteristics can be attributed to several architectural decisions:

- The use of AES-GCM as a parallelizable authenticated encryption scheme with hardware acceleration support on modern processors
- The constant-time implementation of HKDF key derivation, which maintains consistent performance regardless of derived key usage
- The efficient block-based processing of AES, which maintains near-constant time performance after accounting for initial setup costs

- Optimization of the temporal credential format to minimize unnecessary metadata while preserving security properties

These findings confirm that the temporal credential system achieves O(1) computational complexity with respect to credential size, making it exception- ally well-suited for deployment in large-scale examination environments with varying credential requirements. The minimal performance variation across dif- ferent input sizes ensures predictable system behavior even under heterogeneous workloads.

3.6 Memory Utilization

Memory efficiency is crucial for server environments handling multiple concur- rent authentication requests. Memory utilization was measured using the psutil library in Python. Each operation's memory usage was sampled 5 times, with forced garbage collection before each measurement to ensure consistency. The reported values represent the mean change in resident set size (RSS) before and after the operation, with standard deviation.

Our evaluation shows that the temporal credential system is memory-efficient:

- **Encryption**: Mean memory delta of 1.36 KB (= 0.15 KB)
- **Decryption**: Mean memory delta of 1.32 KB (= 0.14 KB)

These minimal and consistent memory requirements enable the system to handle thousands of concurrent authentication operations without significant resource constraints. The similar memory usage for encryption and decryption aligns with the expected behavior of symmetric cryptographic operations.

3.7 Performance Implications

The performance evaluation yields several important insights for our temporal credential system:

- **Key Rotation Efficiency**: The extremely low overhead of key rotation operations (approximately 259× faster than encryption) enables frequent key updates without performance penalties. This allows the system to maintain

  forward secrecy with minimal impact on overall system performance.
- **Scalability**: The near-constant time characteristics for different input sizes ensures predictable performance regardless of credential complexity.
- **Resource Efficiency**: Minimal memory footprint enables deployment in resource-constrained environments while supporting high concurrency.
- **Bulk Processing Capability**: The system demonstrates efficient handling of large-scale credential generation tasks, with database operations rather than cryptographic functions constituting the primary performance bottle- neck.

These findings demonstrate that our temporal credential system effectively balances security requirements with performance considerations, making it suit- able for the high-throughput authentication demands of large-scale examination systems.

The performance evaluation demonstrates that the temporal credential sys- tem achieves its design goals of efficient, secure, and time-bound authentication without compromising performance. This makes it particularly suitable for examination systems where credential validity must align precisely with examination schedules.

Conclusion

The AECIS platform demonstrates that temporal access control combined with hierarchical RBAC can significantly enhance examination system security while maintaining operational efficiency. Our evaluation shows three key results:

- The temporal credential system achieves cryptographic operations in 9.11×10^{-5}s (encryption) and 1.07×10^{-4}s (decryption), providing strong security guarantees (Section 5.2). The 3.52×10^{-7}s key rotation enables frequent

  credential updates without performance degradation.
- The bulk processing pipeline handles 500-center onboarding in 249.1ms (95% CI [239.8, 258.4]ms) with database writes constituting 56.5% of processing time (Table 8). The atomic transaction model ensures data integrity during high-volume operations.
- The role-based communication framework maintains 99.8% uptime under 500 concurrent users with end-to-end encrypted messaging (128-bit GCM tags) and multi-channel notifications (100/min rate limit).

Future work will focus on extending the temporal credential system to sup- port hierarchical delegation and implementing more granular access control within the role-based framework. Additionally, we plan to investigate integration with blockchain-based verification methods to further enhance examination integrity guarantees.

References

[1]A. M. Kanner and T. M. Kanner, "Special Features of TLA + Temporal Logic of Ac- tions for Verifying Access Control Policies," in 2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekater- inburg, Russia, 2021, pp. 1-6, doi: 10.1109/USBEREIT51232.2021.9455090.

[2]A. Masood, A. Ghafoor, and A. Mathur, "Conformance Testing of Temporal Role-

[3]Based Access Control Systems," IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 2, pp. 144-158, April-June 2010, doi: 10.1109/TDSC.2008.41.

[4]P. Zhezhnych, T. Burak, and O. Chyrka, "On the temporal access control imple-

[5]mentation at the logical level of relational databases," in 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technolo- gies (CSIT), Lviv, Ukraine, 2016, pp. 1-4, doi: 10.1109/STC-CSIT.2016.7589875.

[6]D. R. Mdaka, T. Muchenje, T. M. Tshilongamulenzhe, and T. E. Mathonsi,

[7]"A Multimodal Authentication Method for Electronic Exams," in 2024 Inter- national Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Port Louis, Mauritius, 2024, pp. 1-5, doi: 10.1109/icABCD62167.2024.10645265.

[8]M. Morales, A. Boyina, D. Kothari, M. Moniruzzaman, and A. Sultana, "Whisper- Link: A Novel Anonymous Messaging Service for a Secured Data Communication," in 2024 IEEE/ACIS 27th International Conference on Software Engineering, Arti- ficial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2024,

[9]pp. 1-6.

[10] T.-Y. Tung, L. Lin, and D. T. Lee, "Pandora Messaging: An Enhanced Self-Message-Destructing Secure Instant Messaging Architecture for Mobile De- vices," in 2012 26th International Conference on Advanced Information Net- working and Applications Workshops, Fukuoka, Japan, 2012, pp. 739-744, doi: 10.1109/WAINA.2012.112.

[11] M. J. Baucas and P. Spachos, "Secure Private Blockchain-Based Instant Messaging Platform for Social Media Services," IEEE Networking Letters, vol. 6, no. 2, pp. 106-109, June 2024, doi: 10.1109/LNET.2024.3386974.

[12] S. Sharaf and H. Mostafa, "A study of Authentication Encryption Algorithms (POET, Deoxys, AEZ, MORUS, ACORN, AEGIS, AES-GCM) For Automotive Se-

[13] curity," in 2018 30th International Conference on Microelectronics (ICM), Sousse, Tunisia, 2018, pp. 246-249, doi: 10.1109/ICM.2018.8704025.

[14] M. Alkhyeli, S. Alkhyeli, K. Aldhaheri, and H. Lamaazi, "Secure Chat Room Ap- plication Using AES-GCM Encryption and SHA-256," in 2023 15th International Conference on Innovations in Information Technology (IIT), 2023, pp. 1-6.

[15] I. Karabey and G. Akman, "A cryptographic approach for secure client - server chat application using public key infrastructure (PKI)," in 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), Barcelona, Spain, 2016, pp. 449-454, doi: 10.1109/ICITST.2016.7856750.

[16] S. Singh, S. Singh, and A. Sharma, "Real-Time Secure Web-Based Chat Appli- cation using Django," in 2023 5th International Conference on Advances in Com- puting, Communication Control and Networking (ICAC3N), Greater Noida, India, 2023, pp. 1-6, doi: 10.1109/ICAC3N60023.2023.10541532.

[17] A. Suganthy and V. Prasanna Venkatesan, "An Introspective Study on Dynamic Role-Centric RBAC Models," in 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 2019, pp. 1-6, doi: 10.1109/ICSCAN.2019.8878827.

[18] M. C. Gunalan, Bavani K, M. Hayden, M. H. R, and Rahul S, "Dynamic Web- site Authentication Using Time-Based One-Time Password (TOTP) for Enhanced Security," in 2025 International Conference on Data Science, Agents & Artifi- cial Intelligence (ICDSAAI), Chennai, India, 2025, pp. 1-6, doi: 10.1109/ICD-SAAI65575.2025.11011563.

[19] R. Kantipudi, A. S. K. Mallavarapu, S. M. Rajagopal, and M. Kagolanu, "A Com- prehensive Analysis on using Multifactor Authentication System for Three Level Security," in 2024 2nd International Conference on Intelligent Data Communica- tion Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2024, pp. 1-6, doi: 10.1109/IDCIoT59759.2024.10467901.

[20] T. Ji, W. Wang, and J. Sun, "Design and Implementation of an Instant Messaging System," in 2023 International Conference on Intelligent Manage- ment and Software Engineering (IMSE), Rome, Italy, 2023, pp. 135-140, doi: 10.1109/IMSE61332.2023.00030.

[21] M. Almukhaylid and H. Suleman, "Socially-Motivated Discussion Forum Mod- els for Learning Management Systems," in SAICSIT '20: Conference of the South African Institute of Computer Scientists and Information Technologists 2020, Cape Town, South Africa, 2020, pp. 1-11, doi: 10.1145/3410886.3410902.

[22] T.-Y. Huang, K.-K. Yap, B. Dodson, M. S. Lam, and N. McKeown, "PhoneNet: a phone-to-phone network for group communication within an administrative do- main," in MobiHeld '10: Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds, New Delhi, India, 2010,

[23] pp. 27-32, doi: 10.1145/1851322.1851331.

[24] M. Wang, K. He, J. Chen, Z. Li, W. Zhao, and R. Du, ”Biometrics-Authenticated Key Exchange for Secure Messaging,” in CCS ’21: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 2021, pp. 2618-2631, doi: 10.1145/3460120.3484746.

[25] L. Bracciale, P. Loreti, E. Raso, and G. Bianchi, ”TooLate: Cryptographic Data Access Control for Offline Devices through Efficient Key Rotation,” in CPSIoTSec ’21: Proceedings of the 2th Workshop on CPS&IoT Security and Privacy, Virtual Event, Republic of Korea, 2021, pp. 57-62, doi: 10.1145/3462633.3483982.

[26] L. Huin, D. Boulanger, and E. Disson, ”An agent and RBAC model to secure cooperative information systems,” in MEDES ’10: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, Bangkok, Thailand, 2010, pp. 122-130, doi: 10.1145/1936254.1936276.

[27] B. Lu, X. Zhang, Z. Ling, Y. Zhang, and Z. Lin, ”A Measurement Study of Au- thentication Rate-Limiting Mechanisms of Modern Websites,” in ACSAC ’18: Pro- ceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 2018, pp. 89-100, doi: 10.1145/3274694.3274714.

[28] NIST, ”Recommendation for Key Derivation Using Pseudorandom Functions (Re- vised),” NIST Special Publication 800-108, 2009. [Online]. Available: https:// nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf

[29] NIST, ”Recommendation for Block Cipher Modes of Operation: Ga- lois/Counter Mode (GCM) and GMAC,” NIST Special Publication 800- 38D, 2007. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/ nistspecialpublication800-38d.pdf

[30] H. Krawczyk and P. Eronen, ”HMAC-based Extract-and-Expand Key Derivation Function (HKDF),” RFC 5869, 2010. [Online]. Available: https://tools.ietf. org/rfc/rfc5869.txt

[31] NIST, ”Advanced Encryption Standard (AES),” FIPS 197, 2001. [Online]. Avail- able: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[32] NIST, ”Guide to Attribute Based Access Control (ABAC) Definition and Con- siderations,” NIST Special Publication 800-162, 2014. [Online]. Available: https:

[33] //nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf

[34] D. M’Raihi et al., ”TOTP: Time-Based One-Time Password Algorithm,” RFC 6238, 2011. [Online]. Available: https://tools.ietf.org/rfc/rfc6238.txt

[35] Django Software Foundation, ”Security in Django,” 2023. [Online]. Available:

[36] https://docs.djangoproject.com/en/4.2/topics/security/

[37] PostgreSQL Global Development Group, ”Encryption Options,” 2023. [Online]. Available: https://www.postgresql.org/docs/current/encryption-options. html

[38] Python Cryptography Authority, ”Cryptographic Recipes and Primitives,” 2023. [Online]. Available: https://cryptography.io/en/latest/

[39] NIST, ”Cybersecurity Framework,” 2023. [Online]. Available: https://www.nist. gov/cyberframework

[40] E. Rescorla, ”The Transport Layer Security (TLS) Protocol Version 1.3,” RFC 8446, 2018. [Online]. Available: https://tools.ietf.org/rfc/rfc8446.txt

[41] OWASP Foundation, ”Application Security Verification Stan- dard (ASVS),” 2023. [Online]. Available: https://owasp.org/ www-project-application-security-verification-standard/

[42] Apache Software Foundation, ”Apache Kafka Documentation: High Through- put Messaging,” 2023. [Online]. Available: https://kafka.apache.org/ documentation/

[43] Redis Ltd., ”Redis Documentation: Redis as a Cache,” 2023. [Online]. Available:

[44] https://redis.io/docs/manual/

[45] ISO/IEC, ”ISO/IEC 27001:2013 - Information Security Management,” 2013. [On- line]. Available: https://www.iso.org/standard/54534.html

[46] WebRTC Authors, ”WebRTC Documentation: Real-Time Communication,” 2023. [Online]. Available: https://webrtc.org/getting-started/