EZRA: Ephemeral Zero-Knowledge Relay Archive

Liam J. Goss

Western Governors University

College of Information Technology

Aprile 18, 2025

Originally submitted April 18, 2025. Revised November 29, 2025.

Abstract

EZRA (Ephemeral Zero-Knowledge Relay Archive) is a secure file-sharing platform developed to address risks associated with metadata exposure, persistent storage, and insider threats. Designed for both anonymous and enterprise use, EZRA supports cryptographically verifiable access through zero-knowledge proofs and provides strong encryption via AES-256-GCM. Public mode enforces strict metadata minimization and client-side encryption, while enterprise mode introduces configurable role-based access control, optional logging, and compliance-aligned key management. Developed using an Agile methodology, the system underwent thorough testing using OWASP-informed attack vectors and passed with over 94% code coverage. EZRA's architecture prioritizes zero trust, privacy-by-design, and modular configurability, resulting in a field-ready solution tailored to secure file exchange in high-risk and regulated environments.

Keywords: secure file sharing, zero-knowledge proof, AES-256-GCM, metadata suppression, ZK-SNARK, privacy by design, cryptographic access control

EZRA: Ephemeral Zero-Knowledge Relay Archive

A: Adopted Policies

The EZRA platform was developed to address the need for secure, anonymous, and ephemeral file sharing across high-risk environments such as journalism, defense, and corporate whistleblowing. As part of the implementation, core security policies were established to ensure alignment with privacy-by-design principles, zero-trust architecture, and minimal data exposure.

Public Mode Policies

In the public deployment mode, policies prioritize user anonymity, metadata suppression, and ephemeral storage. EZRA adopts a strict no-logging policy, ensuring that no IP addresses, user agents, or access timestamps are retained. In addition, uploaded files are encrypted immediately by the client's browser using AES-256-GCM and stored with modification timestamps forcibly reset (timestomping) to a fixed epoch (Dec 31, 1969 11:59:59 PM). This timestomping is done at the application level. Due to file system format and operating system constraints, the creation time ("ctime") and birth timestamps cannot be altered without kernel-level operations (which are beyond EZRA's scope). Nonetheless, EZRA ensures no actionable metadata is retained beyond the encrypted file content. Files are deleted automatically based on expiration timers or after a single access, depending on user configuration.

Enterprise Mode Policies

EZRA implemented configurable policies for enterprise clients that extend functionality to support internal compliance and access accountability. These include:

- **Optional Metadata Logging:** Organizations may enable audit logging that tracks employee actions without exposing sensitive file contents or filenames. Logs are sanitized before storage.

- **Role-Based Access Control (RBAC):** A basic RBAC policy allows system administrators to define user roles and assign access scopes using a static configuration file. These policies enforce access boundaries and support organizational data governance models.
- **File Type Restrictions:** Enterprise deployments may enable MIME-type validation and define allow/disallow lists to mitigate the risk of malicious uploads or file spoofing.
- **TLS Enforcement:** All enterprise deployments must enforce HTTPS (via Let's Encrypt, a self-signed certificate, etc.) to protect data in transit.

These policies balance maintaining cryptographic integrity and meeting operational or legal obligations common in corporate/regulated environments.

A.1: Cybersecurity Decision-Making:

EZRA enhances cybersecurity decision-making by shifting trust from the human-defined access controls to cryptographically verifiable proofs. EZRA enables public and enterprise users to share files without relying on usernames, passwords, or traditional authentication mechanisms. In public mode, access is granted exclusively through valid zero-knowledge proofs (ZKPs), allowing individuals to demonstrate authorization without revealing identity or contextual metadata. This reduces the risk of phishing, credential compromise, and insider abuse, as no long-term access tokens or accounts are needed.

Cybersecurity decision-making is improved in enterprise mode using static role-based access control, configurable audit logging, and policy-driven file expiration mechanisms. Administrators can define access roles, expiration policies, and logging preferences at a granular level via configuration files, enabling proactive data governance. All access decisions are cryptographically bound to a proof-of-access model, reducing ambiguity and eliminating reliance on perimeter-based trust models.

EZRA enables stakeholders to make data access decisions based on verifiable conditions rather than assumed trust by enforcing ephemeral file lifecycles, non-interactive proof validation, and metadata minimization. This aligns with zero-trust principles and promotes defensible, auditable security practices across public and enterprise use cases.

B: Cybersecurity Assurance Criteria

Promotes Automation in Cybersecurity

EZRA promotes automation in cybersecurity by removing the need for manual identity checks, access control enforcement, and post-access logging in its public deployment mode. Using non-interactive zero-knowledge proofs automates access validation: users must submit a valid cryptographic proof before processing any file retrieval. This proof is verified without human intervention and without revealing any identifying information, streamlining access control and minimizing administrative overhead. In enterprise mode, automated enforcement of expiration policies, proof validation, and role-based access checks allows for scalable deployment with minimal operational complexity. This automated verification process aligns with modern cybersecurity principles, where reducing human dependency in access control decisions is essential to lowering insider threat risk and misconfiguration errors (Scarfone et al., 2020).

Improves and Modernizes Security

EZRA modernizes security through the integration of advanced cryptographic primitives and privacy-preserving protocols. Most notably, it employs:

- Zero-Knowledge Proofs (ZKPs) using a Rank-1 Constraint System (R1CS) circuit model.
- Poseidon Hash functions optimized for ZKP-native operations.
- AES-256-GCM for authenticated encryption of all file content.

By replacing traditional identity-based access models with ZKP-based verification, EZRA eliminates the need for storing passwords, using session tokens, or implementing a centralized user databases, significantly reducing the attack surface. In public mode, the encryption key is generated in-browser client-side and embedded directly within the secret used to create the access proof, ensuring that the EZRA server never has access to the decryption key at any point. This enforces true end-to-end confidentiality and enables the platform to uphold zero-knowledge principles even in the absence of user accounts or trusted intermediaries.

In contrast, enterprise mode uses a server-side encryption model where a new random key is generated for each uploaded file. These file-specific keys are then encrypted using a master enterprise encryption key defined in the deployment configuration. This approach gives enterprise clients greater control over key management and aligns with internal compliance or auditability requirements while ensuring strong confidentiality and data isolation between users.

Additionally, metadata suppression strategies such as timestomping and log disabling remove common channels for surveillance, inference attacks, and forensic recovery in public use cases. These improvements reflect core security objectives outlined by NIST and privacy by design guidelines in GDPR (Privacy by design, 2024; Dworkin, 2007). In enterprise mode, the solution supports policy-based access, TLS-encrypted transmission (also featured in public mode), and optional audit logging, all aligning with standard best practices in high assurance environments.

Implements Industry-Standard Security Tools and Infrastructure

EZRA is built entirely on open-source, industry-validated technologies, ensuring transparency, flexibility, and community-vetted security. These include:

- AES-256-GCM for encryption as recommended by NIST SP 800-38D (Dworkin, 2007)
- TLS (HTTPS) for all transport security using Let's Encrypt or self-signed certificates
- Circom + SnarkJS for ZK-SNARK (Zero-Knowledge Succinct Non-interactive Arguments of Knowledge) proof generation and verification, conforming to the ZKProof Community Reference (ZKProof, 2022)

- Python + Flask for the API backend, supported by virtual environments and compatibility with Linux server infrastructure

EZRA's compliance with ZKProof v0.3 standards ensures soundness, completeness, and non-interactive proof supports, while its use of established cryptographic algorithms ensures alignment with industry security expectations where feasible. Although the Poseidon hash function is not FIPS-certified like SHA-256, it is a widely adopted ZK-native hash function optimized explicitly for efficiency in zero-knowledge proof systems. EZRA uses Poseidon within the Circom + SnarkJS framework, using the Groth16 proving system on the bn254 elliptic curve, a configuration widely supported by the ZKProof and Ethereum research communities (ZKProof, 2022). This setup was recently benchmarked alongside other ZK-friendly hash functions, where Poseidon and its successor Poseidon2 demonstrated high efficiency in constraint count and proof generation performance within Circom-based circuits (Guo et al., 2024). These findings reinforce its suitability for low-latency proof systems like EZRA, where minimizing circuit complexity is critical for usability and speed.

C: Data Collection and Implementation Elements

Collects Digital Evidence (for Analysis or Forensics)

In enterprise mode, EZRA includes a configurable logging system that allows administrators to collect limited but meaningful digital evidence for internal analysis, security monitoring, and forensic readiness. The system captures (configurable) events such as file uploads, access attempts, proof validation outcomes, role-based authorization checks, and deletion actions. Importantly, file contents are never logged (but filenames can be optionally logged), preserving confidentiality while providing operational visibility.

These logs are structured in JSON format for compatibility with centralized log management systems and can be used in incident response scenarios to reconstruct user behavior or validate access policy enforcement. In line with privacy-aware design, log entries are timestamped, sanitized, and scoped to exclude any sensitive user or file identifiers, unless explicitly configured otherwise by the organization. Public mode, in contrast, intentionally disables all logging to prevent the accumulation of metadata or traceable artifacts, consistent with its threat model focused on anonymity and zero-knowledge interaction.

Implements Confidentiality, Integrity, and Availability (CIA)

EZRA was designed from the ground up to enforce the three core pillars of information security. Confidentiality is enforced through AES-256-GCM encryption, ensuring that all uploaded files are protected at rest with authenticated encryption. In public mode, end-to-end confidentiality is guaranteed by client-side key generation, while enterprise mode enables policy-driven control over encryption key lifecycle. TLS encryption is enforced in both modes to ensure secure transmission.

Integrity is ensured through the authentication tag provided by AES-GCM, which prevents undetected file tampering (Dworkin, 2007). Additionally, RBAC and zero-knowledge proof validation mechanisms ensure that only authorized users can retrieve files, with proof verification acting as an integrity gate for access control.

Availability is supported by a lightweight and stateless architecture that relies on minimal server dependencies. Files are stored temporarily with automated deletion policies, reducing the risk of long-term data exposure while ensuring timely availability within the defined access window. In enterprise deployments, configuration options allow administrators to set access windows, enforce role usage limits, and manage proof validation behavior in a controlled, fault-tolerant way. These mechanisms collectively ensure that EZRA satisfies foundational cybersecurity principles while adapting its behavior based on the security expectations of the environment in which it is deployed.

D: Cybersecurity Investigations and Mitigations

EZRA implements both preventative and responsive mechanisms to mitigate cybersecurity incidents within its deployment environments. While its public mode prioritizes anonymity and metadata suppression (intentionally disabling persistent logging to protect at-risk users), its enterprise mode enables configurable controls to support incident detection, investigation, and response.

Incident Investigation in Enterprise Mode

In enterprise deployments, EZRA supports the investigation of security events by generating structured logs that capture the following:

- Upload and download attempts (optionally including timestamps, user-roles, user-emails, IP address, and user agent)
- Proof validation outcomes (success/failure)
- Rejection of malformed proofs or unauthorized access attempts
- Expiration or deletion triggers
- File extension spoofing attempts and whitelist/blocklist events

The logs are stored in a machine-readable format (JSON) and can be ingested by security information and event management (SIEM) systems or forensic tools. Although file contents and sensitive metadata are excluded to preserve confidentiality, the available telemetry allows enterprise security teams to trace actions back to specific roles or time windows, supporting both proactive alerting and post-incident root cause analysis.

Mitigation Capabilities Across Modes

EZRA incorporates multiple controls to prevent common file-sharing threats before they escalate into incidents:

- **Zero-Knowledge Proof Verification:** Prevents unauthorized access by requiring valid, cryptographically sound proofs. Attempts to reuse, spoof, or guess proofs are rejected at the protocol level.
- **Filename and Content Sanitization:** Protects against injection attacks, such as command or newline injection through filenames or headers. All uploaded files undergo strict MIME-type detection and content-based validation in enterprise mode.
- **Timestomping and Log Suppression:** Reduces traceability in public mode to prevent adversaries from performing correlation attacks or forensic recovery after access.
- **Brute Force and Enumeration Protections:** The download route does not reveal file existence unless a valid proof is provided, mitigating oracle-based attacks (Packetlabs, 2024).
- **Expiration and Auto Deletion:** Limits exposure time and reduces the attack window for stolen or intercepted secrets.

In addition, the system was tested using OWASP-informed test cases and the Burp Suite Community Edition to simulate malformed upload attempts, proof manipulation, and metadata extraction strategies. Any vulnerabilities identified during testing were addressed, and the mitigations were verified through re-testing. By incorporating proactive defenses and supporting forensic investigation in enterprise contexts, EZRA enables organizations to detect, analyze, and respond to cybersecurity threats while preserving user privacy in public deployments.

E: Cybersecurity Plans and Procedures Developed

The cybersecurity posture of EZRA was developed using a defense-in-depth strategy that combines cryptographic best practices, zero-trust design principles, and configurable operational controls. These plans and procedures were structured to support both anonymous public usage and enterprise deployments with enhanced audit and policy features.

Cryptographic Procedures

All files are encrypted using AES-256-GCM, a NIST-approved algorithm that supports authenticated encryption to ensure confidentiality and integrity (Dworkin, 2007). In public mode, keys are generated client-side and never shared with the server, enforcing strict end-to-end encryption. In enterprise mode, keys are server-generated per file container, and the corresponding key and nonce are then encrypted using a master enterprise key, offering compliance-focused key control.

Zero-Knowledge Proofs enforce access without requiring identity disclosure. All proof circuits were implemented using the Circom framework and follow the Rank-1 Constraint System model per the ZKProof Community Reference v0.3 (ZKProof, 2022). The solution uses Groth16 proofs on the bn254 curve, and access verification occurs server-side before any file retrieval is permitted.

Operational Procedures

- **Public Mode:** No logging, IP retention, or timestamps are stored. Files expire after a user-defined time or earlier after the first download. Metadata is obfuscated and destroyed (e.g., via timestomping). Data is encrypted in the browser before transit using AES-256-GCM, then encrypted in transit via TLS, and finally, at rest, it remains encrypted. File decryption only occurs in the recipient's/downloader's browser after transit.
- **Enterprise Mode:** Allows RBAC enforcement, optional logging, TLS enforcement, and file type restrictions. Logs can be toggled to include proof verification attempts, access actions, and role-based policy decisions while avoiding file-content exposure.

- **Penetration Testing and Threat Modeling:** Procedures included test cases for file-spoofing, brute-force proof attempts, malformed uploads, and oracle-based inference attacks. Mitigations were implemented and retested for verification.

These procedures are defined in both the source code and deployment documentation (GitHub README), and this document allows organizations to tailor them based on compliance and operational requirements.

E.1 Alignment to Cybersecurity Initiatives and Regulatory Compliance

EZRA aligns with multiple key cybersecurity initiatives and regulatory principles:

- **NIST Guidelines:** By using AES-256-GCM and TLS for secure file encryption and transmissions, EZRA aligns with standards from NIST SP 800-38D and SP 800-52 Rev. 2 on cryptographic best practices (Dworkin, 2007; McKay & Cooper, 2019).
- **Zero Trust Architecture:** EZRA removes implicit trust by requiring all access to be validated via cryptographic proof, reflecting the zero-trust principles outlined in NIST SP 800-207 (Rose et al., 2020).
- **Privacy by Design (GDPR):** In public mode, EZRA meets the privacy-by-design standard of the General Data Protection Regulation (GDPR) by defaulting to data minimization, disabling logging, and enforcing ephemeral storage (Privacy by Design, 2024).
- **ZKProof Interoperability:** The ZKP implementation conforms to the ZKProof v.03 standard, ensuring security guarantees of soundness, completeness, and zero-knowledge for public and enterprise use cases (ZKProof, 2022).
- **OWASP Secure File Transfer Guidelines:** File uploads and downloads are protected from spoofing, injection, and enumeration attacks under OWASP Web Security Testing Guide best practices (OWASP, 2023).

While EZRA does not directly integrate with IAM frameworks like LDAP or Okta, it provides foundational compliance support through configurable roles, key management, logging, and secure protocol. These controls align with cybersecurity frameworks like CIS Controls v8.1 and ISO/IEC 27001 principles related to access control, cryptographic protection, and auditability (Center for Internet Security, 2021; International Organization for Standardization, 2022).

E.2 Applications, Tools, Guides Developed:

The following supporting documentation and tooling were developed as part of the EZRA implementation:

- **enterprise_config.json:** A sample configuration file defining RBAC roles, logging preferences, and TLS enforcement options. This allows organizations to tailor their deployment to specific policies.
- **User & Admin Deployment Guide (Markdown):** Explains how to install dependencies, configure modes, upload/download files, and validate ZKPs using the CLI, GUI, Web, or manual testing.
- **System Architecture Diagram:** A visual artifact (see **H: Original Artifact**) showing file flow, proof generation/validation, encryption layers for enterprise mode
- **Testing Utilities:** Internal scripts to simulate proof validation, malformed uploads, expiration handling, and metadata suppression. These files were used for both automated and manual security testing.
- **Installation Script:** A setup script for creating the Python virtual environment, installing dependencies, and initializing a development instance of EZRA, including ZKP verification key generation.

These resources ensure that technical and non-technical users can install, configure, and safely operate the platform with minimal friction.

F: Post-Implementation Environment

Following implementation, EZRA was deployed in a private development environment and validated it across both public and enterprise deployment modes. The post-implementation environment includes new application logic, role-based access control, audit configuration capabilities, and a fully functional Zero-Knowledge Proof access system for secure file handling. A system architecture diagram (see **H: Original Artifact**) illustrates the core components: the frontend client, Flask-based API server, AES-256-GCM encryption layer, Circom/SnarkJS-based proof system, and temporary file vault. Each mode (public and enterprise) supports tailored security behaviors, such as encryption key placement, logging scope, and access controls.

F.1: Improved Security Posture and Organizational Efficiency

EZRA significantly enhances an organization's security posture by enforcing:

- Zero-trust access via cryptographic proofs instead of traditional authentication
- End-to-end encryption of file content with no server-side key visibility in public mode
- Role isolation and least privilege through basic RBAC in enterprise mode
- Ephemeral file storage to reduce long-term breach impact

By eliminating account-based identity in public mode and allowing organizations to configure controls via JSON-based policy files in enterprise mode, the solution reduces administrative overhead and improves operational efficiency. Staff are not required to manage users, rotate passwords, or maintain long-lived logs unless necessary for internal compliance.

F.2: New Data and Business Process Impact

In enterprise deployments, EZRA introduces structured audit logs that can be configured to capture:

- Upload/download events
- Proof validation success or failure

- Role-based access evaluations
- File expiration and deletion actions
- File type bypass attempts
- Number and names of files uploaded

These logs are sanitized and timestamped, making them suitable for integration with SIEM tools or internal audit platforms. Business processes such as incident response, file sharing, and access monitoring are streamlined by allowing lightweight deployment with automatic policy enforcement. Public mode deliberately avoids collecting any new data, which is consistent with its privacy-focused use case. This contrast will enable organizations to offer the platform to both anonymous users in sensitive fields (e.g., journalism, activism) and regulated enterprise clients.

F.3: Summative Evaluation and Test Results

A summative evaluation plan included penetration testing and simulated attacks targeting upload validation, proof-based access control, and metadata exposure across both public and enterprise deployment modes. Testing methodologies were informed by OWASP guidance and designed to verify the platform's ability to reject unauthorized access and handle malformed or malicious input safely.

Tools and Techniques Used

- Burp Suite Community Edition for active HTTP/s fuzzing and input injection
- OWASP Web Security Testing Guide for structured attack case coverage
- Custom malformed proof generators to test ZKP verification endpoints
- File spoofing simulations, including renamed executables and invalid MIME headers
- Oracle inference attacks, testing where file presence could be confirmed without submitting a valid proof.

Results

- 100% of unauthorized download attempts were rejected

- 100% of invalid, replayed, or tampered ZKPs were blocked
- Proofless access attempts could not infer file existence (oracle mitigation successful)
- Timestomping behavior removed actionable access/modify metadata in public mode
- Enterprise mode logging captured structured events without leaking sensitive data

Test Coverage

- Unit tests were written for encryption, expiration logic, proof validation, URL endpoints, and role-based access behavior
- Integration tests validated upload/download flows and proof checks across both deployment modes
- All functionality and security tests passed
- Test coverage exceeded 94% of the total codebase, verified using Python's "coverage" tool

Corrections and Retests

- Removed pre-proof file existence checks, which previously allowed basic oracle behavior
- Implemented MIME-type scanning for enterprise uploads to guard against spoofed file types caused by renaming extensions
- Sanitized and scoped log outputs to prevent newline injection and data leakage.

All mitigations were confirmed via repeat testing and reviewed for regression following these corrections. The final build meets all summative evaluation criteria and reflects a hardened, field-ready implementation.

F.4: Post Implementation Risks and Mitigation

Risk	Likelihood	Impact	Mitigation
Metadata exposure in enterprise logs	Medium	High	Default sanitization; logging is opt-in
Proof reuse or tampering	Low	High	Proofs are tied to secrets and are tamper-checked
Encryption misconfiguration	Medium	High	Static test cases & encryption unit tests
Incomplete file deletion	Low	Medium	Expiry logic tested and confirmed within 120 seconds
Client loss of secret	High	Low	Recovery is not possible by design; user education is recommended

NOTE: These risks are documented in deployment materials and mitigated via code-level safeguards and administrative guidelines

F.5: Stakeholder Needs Alignment

EZRA was developed in response to security, privacy, and operational concerns raised by six key stakeholder groups identified below. Each system component – from cryptographic protocol choice to deployment configuration – was guided by the stakeholders' specific needs, influence, and risk posture.

F.5.1: Software Engineers (Internal)

Software engineers required a technically feasible modular solution supporting public and enterprise deployments. EZRA meets this need with a Python-based backend (Flask) and modular architecture that cleanly separates file encryption, proof generation, and access verification layers. The system offers extensibility and manageable complexity by implementing Circom + SnarkJS for proof logic and maintaining clear interfaces between components. Using static configuration files for RBAC and logging also reduces the need to maintain dynamic databases, minimizing operational burden. Engineers can quickly deploy and extend the system using the included documentation and “enterprise_config.json” template.

F.5.2: Security Architects (Internal)

Security architects were responsible for selecting cryptographic libraries and designing a zero-trust access model that minimizes exposure and mitigates insider threats. EZRA directly addresses these concerns by employing:

- AES-256-GCM for strong encryption with integrity guarantees
- ZK-SNARKs using GROTH16 on the bn254 curve to enforce non-interactive, anonymous access control
- Poseidon hash functions for optimized ZKP performance within R1CS constraints
- Strict timestomping and log suppression in public mode to preserve anonymity

The solution reflects zero-trust architecture principles outlined in NIST SP 800-207 (Rose et al., 2020) and is hardened against insider enumeration or privilege escalation.

F.5.3: Compliance Officers (Internal)

Compliance officers emphasized the need for standards-aligned encryption practices, auditability in enterprise environments, and data minimization policies. EZRA addresses these by:

- Providing a detailed audit trail in enterprise mode, without logging file content (and optionally filenames)
- Using FIPS-approved AES-256-GCM and TLS for secure handling as per NIST SP 800-38D and SP 800-52r2
- Supporting GDPR-style privacy-by-design defaults in public mode, with all logging and tracking disabled

- Including technical documentation for encryption behavior, access validation, and metadata handling, aiding internal reviews or regulatory disclosures

These features help organizations meet internal and external regulatory mandates such as ISO/IEC 27001 and CIS Controls V8.1.

F.5.4 Penetration Testers (External, Third-Party Security Firm)

Penetration testers require a threat model, access to test environments, and clearly defined verification boundaries. EZRA delivered this through:

- A structured threat model covering spoofing, enumeration, and oracle-style inference attacks
- Early access to testing scripts and documentation
- A responsive update cycle where identified vulnerabilities (e.g., oracle-style file presence tests) were mitigated and retested

Final test results confirmed 0% unauthorized access success and successful mitigation of all high-priority threats, supporting secure deployment.

F.5.5 Enterprise Security Teams (External, Corporate/Government Clients)

These teams needed a configurable and compliant system with logging, RBAC, and encryption controls suitable for internal deployment. EZRA provides:

- A deployment-mode toggle between public and enterprise, allowing tailored security policies
- RBAC enforcement through static configuration files, enabling simple role-to-permission mapping
- Logging hooks via JSON output for integration with SIEM platforms, scoped to actions (not file content)

- Configurable file expiration, TLS enforcement, and per-file-container key wrapping for compliance

EZRA makes it easy for these teams to enforce policies that align with organizational security objectives and integration constraints.

F.5.6 Public Users (External, Anonymous End Users)

Public users such as whistleblowers, journalists, activists, and privacy-conscious individuals influenced the system's focus on privacy, anonymity, and trust minimization. EZRA fulfills these expectations by:

- Requiring no user accounts or identifiers
- Performing client-side encryption, ensuring the server never holds decryption keys
- Enforcing zero logging, no IP address/user agent retention, and ephemeral file lifecycles
- Enabling ZKP-based access control with no metadata leakage

This mode positions EZRA as a unique, secure, anonymous communication and data transfer tool in high-risk or surveillance-heavy contexts.

Across all roles – technical, operational, compliance, and end-user – EZRA delivered features and safeguards aligned with stakeholder expectations and threat models. The combination of privacy-by-design defaults, verifiable cryptographic access, and flexible deployment configurations ensures that EZRA is adaptable to both anonymous use and enterprise security infrastructure.

G: Post-Implementation Maintenance Plan

Following deployment, the EZRA platform includes a structured post-implementation maintenance plan to ensure long-term security, functionality, and configurability in public and enterprise environments. This plan addresses patching, key management, log rotation, proof updates, and configuration oversight, with procedures designed to minimize administrative overhead while supporting evolving compliance and threat requirements.

G.1: Cryptographic Updates and Circuit Maintenance

EZRA uses Circom-based zero-knowledge proof circuits compiled for Groth16 on the bn254 curve. While these circuits are currently optimized and stable (Guo et al., 2024), cryptographic best practices dictate periodic reassessment. Maintenance includes:

- Recompiling circuits with updated trusted setups or proof systems as needed
- Monitoring for deprecations in the bn254 curve or vulnerabilities in the Groth16 proving system
- Periodic revalidation of Poseidon hash compatibility and performance within the circuit framework

Any changes to cryptographic parameters or Poseidon hash constants will require re-generating proofs and verifying backward compatibility with existing files where applicable.

G.2: Encryption Key Management

In enterprise mode, the master encryption key used to wrap file-container-specific keys must be securely stored and rotated per the organization's key lifecycle policy. Maintenance responsibilities include:

- Secure backup and offline storage of the master key
- Periodic rotation and archival of expired keys
- Updating encryption configuration without compromising access to historical files

Public mode does not require server-side key storage; all encryption is client-side and ephemeral.

G.3: Configuration Management

EZRA supports policy-driven configuration via structured JSON files. As business or compliance needs evolve, system administrators may need to update:

- RBAC role definitions and access scopes
- Logging verbosity and retention durations
- File expiration defaults and deletion behavior
- TLS certificate paths and enforcement requirements

Changes should be tested in a staging environment before deployment to avoid misconfigurations.

G.4: Logging and Audit Rotation

Audit logs may accumulate over time in enterprise mode, depending on the retention policy and access/usage frequency. Maintenance procedures include:

- Log rotation schedules and automated archival
- Secure storage of logs using cryptographic checksums or signature mechanisms
- Regular review for suspicious activity or usage anomalies

Public mode requires no log maintenance, as it disables persistent logging entirely.

G.5: Software and Dependency Updates

EZRA's backend is written in Python using Flask, with cryptographic logic implemented via Circom and SnarkJS. The maintenance plan includes:

- Periodic updates to Python packages (using pinned versions and virtual environments)
- Monitoring upstream libraries for security advisories (e.g., Flask, cryptography, SnarkJS)
- Rebuilding the environment when core libraries or cryptographic dependencies change.

Package versions and environment variables should be documented and version-controlled using requirements files and deployment scripts.

G.6: Security Review and Revalidation

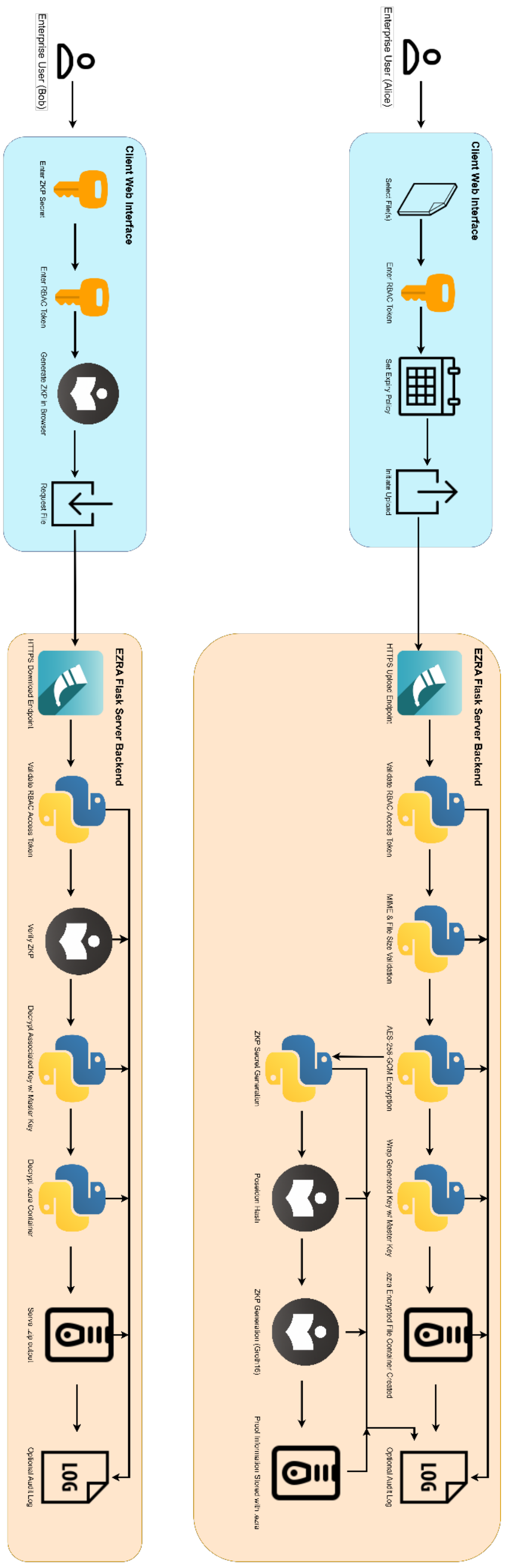
A biannual security review is recommended to include:

- Re-running penetration tests using current OWASP methodologies
- Verifying no regressions in metadata suppression, file deletion behavior, or proof validation
- Reviewing log formats and cryptographic parameters for continued alignment with standards such as NIST SP 800-38D and ZKProof v0.3

Reviews should be documented and used to inform change management decisions.

This maintenance plan ensures that EZRA remains secure, adaptable, and operationally viable long after initial deployment, supporting ongoing privacy and compliance needs for both anonymous and enterprise-focused file-sharing use cases.

H: EZRA Enterprise Mode - Secure Upload and Download Flow (Architecture Diagram)



The previous diagram illustrates EZRA's enterprise mode's full cryptographic and procedural flow, from file upload to authorized retrieval. It visualizes how client devices submit files to the server, which applies MIME validation, AES-256-GCM encryption, and master key wrapping. Zero-knowledge proofs are generated via the Groth16 protocol on the bn254 curve using Poseidon hash commitments. The server stores encrypted .ezra containers and logs sanitized events per enterprise policy. Upon receiving a secret, the recipient client submits a proof of access, which is verified before file decryption and delivery. This artifact demonstrates EZRA’s alignment with zero-trust architecture and showcases its privacy-preserving, verifiable access design.

References

Center for Internet Security. (2021). *CIS Controls v8: The Center for Internet Security Critical Security Controls for Effective Cyber Defense*. https://www.cisecurity.org/controls/v8-1

Dworkin, M. (2007b, November 28). *Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC*. NIST Computer Security Resource Center. https://csrc.nist.gov/pubs/sp/800/38/d/final

Guo, H., Feng, Y., Wu, C., Li, Z., & Xu, J. (2024). *Benchmarking ZK-friendly hash functions and SNARK proving systems for EVM-compatible blockchains*. arXiv. https://arxiv.org/abs/2409.01976

International Organization for Standardization. (2022). *ISO/IEC 27001:2022 - Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. https://www.iso.org/standard/82875.html

McKay, K., & Cooper, D. (2019, August 29). *Guidelines for the selection, configuration, and use of Transport Layer Security (TLS) implementations*. NIST Computer Security Resource Center. https://csrc.nist.gov/pubs/sp/800/52/r2/final

OWASP Foundation. (2023). *OWASP Web Security Testing Guide (WSTG) v4.2*. https://owasp.org/www-project-web-security-testing-guide/

Privacy by design. General Data Protection Regulation (GDPR). (2024, July 25). https://gdpr-info.eu/issues/privacy-by-design/

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020, August 11). *Zero trust architecture*. NIST Computer Security Resource Center. https://csrc.nist.gov/pubs/sp/800/207/final

Scarfone, K., Scholl, M., & Murugiah, M. (2020a, August). *ITL BULLETIN AUGUST 2020 - Security Considerations for Exchanging Files Over the Internet*. NIST Computer Security Resource Center. https://csrc.nist.gov/files/pubs/shared/itlb/itlbul2020-08.pdf

What are oracle attacks?. Packetlabs. (2024, December 27). https://www.packetlabs.net/posts/what-are-oracle-attacks/

ZKProof Community. (2022). *ZKProof community reference* (Version 0.3). ZKProof. https://docs.zkproof.org/reference