PHANTOMVOX: A Review of Secure File Sharing Platforms
A Review of Cryptographic Techniques, Access Control, and Audit Compliance for Modern Secure File Exchange

Hidhesh A[1], Kandha Prasanth S[2], Sindhura Selvam[3], Mrs. Sasikala[4]
[1, 2, 3] Dept of Cyber security
[4] Associate Professor, Dept of Cyber security
[1, 2, 3, 4] Dhanalakshmi Srinivasan University Samayapuram, Tiruchirappalli – 621112, Tamil Nadu, India

Abstract- *Secure Digital File Sharing Has Become A Fundamental Requirement For Individuals, Enterprises, And Regulated Industries In An Era Characterized By Pervasive Data Breaches And Expanding Regulatory Obligations. Traditional File-Sharing Solutions—Consumer Cloud Drives, Email Attachments, And Ftp Servers—Consistently Fail To Provide End-To-End Encryption, Granular Access Control, Cryptographic File Integrity Verification, And Tamper-Evident Audit Trails. This Review Paper Analyses The Academic And Standards Literature Underpinning Modern Secure File Sharing And Uses That Analysis To Contextualise The Design Of Phantomvox, A Proposed Full-Stack Web Application That Integrates Aes-256-Gcm Encryption At Rest, Tls 1.3 In Transit, Role-Based Access Control (Rbac), Multi-Factor Authentication Using Totp, Sha-256 Integrity Verification, And An Append-Only Hash-Chained Audit Log Into A Unified Platform. Six Literature Sources Spanning Symmetric Encryption, Transport Security, Authentication, Access Control, And Compliance Logging Are Critically Reviewed, And Five Persistent Research Gaps Are Identified: Absence Of Unified End-To-End Encryption In Accessible Tools, Weak Access Control Defaults, Insufficient Auditability, Lack Of File Integrity Enforcement, And Poor Usability Of Secure Sharing Workflows. The Phantomvox System Directly Addresses Each Gap Through Its Modular Architecture, Built On Node.Js, Express.Js, React, And Postgresql, And Is Demonstrated To Satisfy All Functional, Security, And Performance Requirements Through Comprehensive Testing.*

Keywords: Secure File Sharing; Aes-256-Gcm; Tls 1.3; Role-Based Access Control; Multi-Factor Authentication; Sha-256; Audit Logging; Jwt; Node.Js; Express.Js; Postgresql; Gdpr; Hipaa; Rbac; Cryptography.

I. INTRODUCTION

The exchange of files across networks has become a central activity for individuals, enterprises, government bodies, and academic institutions. Sensitive documents, financial statements, intellectual property, medical records, legal contracts, and personal photographs are routinely transmitted across the internet using a variety of platforms. While the convenience of cloud-based file sharing has revolutionised collaboration and productivity, it has simultaneously introduced an expansive attack surface for adversaries who target valuable or confidential data. High-profile data breaches over the past decade have repeatedly demonstrated that even the largest service providers can suffer compromises that expose user data, often because the data is stored in plaintext or protected only by server-side encryption that the provider itself can reverse.

Traditional platforms such as Google Drive, Dropbox, Microsoft OneDrive, email attachments, and FTP servers offer broad accessibility and ease of use, but they do not consistently provide end-to-end encryption, fine-grained access controls, expiring share links, or tamper-proof audit trails. The shared-link model used by many consumer services produces URLs that, once leaked, grant indefinite access to anyone who possesses them. Similarly, traditional email attachments transit a number of intermediary servers and can be retained for extended periods in inboxes, backups, and spam filters where the original sender exercises no control over their lifetime.

To address these shortcomings, this paper presents the design and implementation of PhantomVox—a Secure File Sharing Platform that leverages modern cryptographic techniques and access-control mechanisms while remaining

accessible to non-technical users through a clean, intuitive web interface. The platform enables users to register, authenticate using a combination of password and time-based one-time password, upload files encrypted with AES-256-GCM before being written to disk, share files with carefully scoped permissions, and view a complete, immutable history of every action performed against the system.

This paper is structured as follows. Section II provides background on secure file sharing systems. Section III presents the literature review of eight significant research contributions. Section IV analyses strengths and limitations. Section V enumerates research gaps. Section VI introduces the proposed PhantomVox system. Section VII describes the system architecture. Section VIII discusses impact and feasibility. Sections IX through XI address advantages, future scope, and conclusions.

II. OVERVIEW OF SECURE FILE SHARING SYSTEMS

A. Traditional File Sharing Tools

Traditional file sharing has historically relied upon consumer cloud drives, email attachments, and FTP/SFTP servers. Services such as Google Drive, Dropbox, and Microsoft OneDrive offer polished interfaces and generous free tiers but operate on a trust model that assumes the service provider is a trusted custodian of user data. Files are typically encrypted on disk using keys managed by the provider, meaning provider employees, government agencies issuing valid legal requests, or any attacker who compromises the provider's infrastructure can access plaintext content. FTP servers provide reasonable performance for large files but lack any user-friendly sharing model, access control beyond filesystem permissions, or detailed logging.

B. Enterprise Content Management Systems

Enterprise platforms such as SharePoint, Citrix ShareFile, and Egnyte offer sophisticated access controls, integration with directory services, and richer compliance features. They are, however, expensive, complex to deploy, and require dedicated administrators. Their user interfaces have historically lagged behind consumer products, which often results in users circumventing them by emailing copies of documents to themselves or to colleagues. The gap between enterprise security and consumer usability remains a fundamental challenge that the proposed platform attempts to bridge.

C. Modern Cryptographic File Sharing

The third generation of file sharing platforms incorporates end-to-end encryption, authenticated encryption modes, and granular access control to protect data throughout its entire lifecycle. Modern platforms can apply AES-256-GCM transparently to every file at upload time, enforce TLS 1.3 for all data in transit, and generate tamper-evident audit logs that satisfy regulatory requirements. The integration of multi-factor authentication eliminates reliance on passwords alone, while RBAC ensures that each user accesses only the resources they are explicitly authorised to access.

D. Node.js in Security Applications

Node.js and the Express.js web framework have emerged as a mature, performant substrate for security-sensitive web applications. The Node.js crypto module provides hardware-accelerated AES-256-GCM and SHA-256 implementations through native AES-NI instructions on supported CPUs, making symmetric encryption inexpensive enough to apply transparently to every file. The availability of well-maintained libraries for JWT authentication, TOTP-based MFA, bcrypt password hashing, and Sequelize ORM for PostgreSQL enables rapid assembly of a comprehensive security stack without custom cryptographic implementation.

III. LITERATURE REVIEW

This section presents a systematic analysis of eight significant research contributions spanning symmetric encryption, transport security, authentication standards, access control models, audit compliance, and open-source secure sharing tools.

A. AES-256-GCM for Data at Rest (NIST SP 800-38D, 2007)

NIST Special Publication 800-38D defines the Galois/Counter Mode (GCM) of operation for the Advanced Encryption Standard, establishing it as the recommended authenticated encryption standard for data at rest [1]. GCM combines confidentiality with built-in integrity through an authentication tag, and NIST recommends 256-bit keys with unique 96-bit initialization vectors per encryption operation. The publication provides the foundational specification for the encryption module in PhantomVox, which generates a fresh DEK and IV per file and stores the wrapped DEK, IV, and authentication tag in the database alongside file metadata. The limitation of this specification is that it does not address key management, key rotation, or integration with web application authentication flows—gaps that PhantomVox addresses through its three-tier key management architecture.

B. TLS 1.3 for Data in Transit (IETF RFC 8446, 2018)

RFC 8446 defines TLS version 1.3, the current best practice for protecting data in transit [3]. Compared to TLS 1.2, it removes obsolete cryptographic primitives, simplifies the handshake, mandates forward-secret key exchange, and reduces handshake latency. Industry analyses recommend disabling all earlier versions of TLS and SSL in production deployments. PhantomVox enforces TLS 1.3 at the Nginx reverse proxy layer and rejects connections that attempt to negotiate older protocol versions. The RFC does not address application-layer authentication or access control, which are separately addressed in the PhantomVox design.

C. TOTP-Based Multi-Factor Authentication (IETF RFC 6238, 2011)

RFC 6238 defines the Time-Based One-Time Password algorithm that underlies virtually every authenticator application in common use [5]. TOTP generates six-digit codes from a shared secret and the current Unix timestamp, making it resistant to replay attacks while remaining compatible with widely available client applications. PhantomVox implements TOTP using the speakeasy library, enforces MFA for all users by default, and rejects TOTP code re-use within the same 30-second window. The RFC does not address session management or token revocation, which are handled through PhantomVox's JWT-based access-token and refresh-token architecture.

D. JSON Web Tokens for Session Management (IETF RFC 7519, 2015)

RFC 7519 defines the JSON Web Token format for representing claims securely between parties [4]. Best practices include short-lived access tokens with refresh tokens stored as HTTP-only cookies, validation of issuer and audience claims, and refresh-token rotation on every use to detect token theft. PhantomVox implements fifteen-minute access tokens, seven-day refresh tokens with per-use rotation, and immediate family invalidation on detected theft. The RFC does not address integration with MFA flows or RBAC enforcement, which are separately implemented in PhantomVox's authentication middleware.

E. Role-Based Access Control (Sandhu et al., 1996)

The seminal paper by Sandhu and colleagues formalised role-based access control and remains the canonical reference [7]. RBAC associates permissions with roles and users with roles, simplifying administration in environments where many users share similar access patterns. PhantomVox implements three principal roles—Administrator, Editor, and Viewer—with per-file and per-folder permission scoping enforced by Express middleware on every API request. The paper predates web application architectures and does not address share-link-based access or TOTP integration, which are novel contributions of the PhantomVox design.

F. Tamper-Evident Audit Logging (OWASP ASVS v4.0.3, 2021)

The OWASP Application Security Verification Standard provides comprehensive guidance on audit logging requirements for security-sensitive applications, including the recommendation for tamper-evident, append-only log stores that record all security-relevant events [8]. PhantomVox implements a hash-chained audit log table in PostgreSQL where each entry incorporates the hash of the previous entry, with daily checkpoint exports to immutable cold storage. OWASP ASVS provides requirements but not implementation guidance, leaving the specific architecture of the hash chain and checkpoint mechanism to the implementer.

G. Open-Source Secure Sharing Landscape (Cloud Security Alliance, 2017)

The Cloud Security Alliance's Security Guidance for Critical Areas of Focus in Cloud Computing surveys the state of cloud file sharing security, identifying end-to-end encryption, access control, and audit logging as the three most critical dimensions [18]. Several open-source projects address overlapping subsets of these requirements: Cryptpad provides end-to-end encrypted collaborative documents but is optimised for real-time editing rather than file storage; Send (formerly Firefox Send) provided ephemeral encrypted file sharing but was discontinued by Mozilla; Nextcloud offers a full-featured self-hosted cloud platform with optional end-to-end encryption, though its encryption module has historically been complex to configure correctly. PhantomVox draws lessons from each of these projects and focuses specifically on the file-sharing use case to make stronger security defaults possible.

H. Applied Cryptography (Schneier, 1996; Boneh& Shoup, 2020)

The foundational texts on applied cryptography—Schneier's Applied Cryptography and Boneh and Shoup's graduate course—provide the theoretical underpinning for every cryptographic decision in the PhantomVox design [16, 17]. Key management is identified as the most common source of cryptographic failure in real deployments, motivating PhantomVox's three-tier key architecture separating the master key, data encryption keys, and session

keys. The texts confirm that AES-GCM's security guarantees hold only when IV uniqueness is maintained across the lifetime of a single key, which PhantomVox enforces through cryptographically secure random IV generation per file.

IV. ANALYSIS OF EXISTING SYSTEMS

A. Comparative Summary

Table 1 presents a structured comparative analysis of the reviewed works and existing file-sharing solutions across the dimensions most critical to a unified secure file-sharing platform.

System / Study	E2E Enc.	RBAC	Audit	MFA
Google Drive / Dropbox	No	Basic	Limited	Optional
Email Attachments	No	No	No	Optional
Enterprise CMS (SharePoint)	Partial	Yes	Yes	Optional
Cryptpad	Yes	No	No	No
Firefox Send	Yes	No	No	No
Nextcloud	Partial	Yes	Partial	Optional
PhantomVox (Proposed)	Yes	Yes	Yes	Enforced

Table 1: Comparative Analysis of Existing Secure File-Sharing Systems

B. Strengths of Existing Literature

The reviewed body of work demonstrates several commendable advances. The maturation of symmetric encryption has produced AES-256-GCM as a universally supported, hardware-accelerated standard that eliminates the performance barrier to universal file encryption. The RBAC model has provided organizations with a scalable administrative framework for managing access to shared resources. The growing literature on hash-chained audit logging confirms that tamper-evident records are achievable without purpose-built hardware or blockchain infrastructure. The TOTP standard has democratised multi-factor authentication by enabling compatibility with a wide range of authenticator applications.

C. Limitations of Existing Literature

Despite these advances, five persistent limitations define the research gap. Fragmented tool ecosystems mean that no reviewed platform integrates AES-256-GCM encryption, TLS 1.3 enforcement, TOTP-based MFA, granular RBAC, expiring share links, and hash-chained audit logging in a single cohesive system. The absence of universal end-to-end encryption in accessible tools leaves the majority of user files vulnerable to provider-side access. Static access control defaults across consumer platforms result in share links that persist indefinitely. Insufficient auditability in consumer tools makes regulatory compliance difficult. Poor usability of secure workflows drives users toward less secure alternatives.

V. RESEARCH GAPS IDENTIFIED

A. Absence of Unified End-to-End Encryption in Accessible Tools

Every reviewed accessible file-sharing platform either foregoes end-to-end encryption entirely or implements it as an optional, complex add-on. A user investigating a potential data breach must trust the provider's assurance that data is encrypted, without any technical mechanism to verify this claim. No widely adopted open-source platform applies AES-256-GCM encryption transparently to every file as a non-negotiable default. The operational cost of this gap is measurable in the volume of data exposed in provider-side breaches that would have been rendered useless by client-independent encryption.

B. Weak Access Control Defaults

Share links generated by consumer platforms are typically unauthenticated, unlimited in download count, and persistent until manually revoked—defaults that make accidental disclosure far more likely than necessary. No reviewed consumer platform enforces expiry, download limits, and optional password protection on all share links by default. The availability of mature JWT and RBAC libraries makes fine-grained, expiry-enforced access control technically feasible without significant implementation overhead, yet it remains absent from the default configuration of widely deployed tools.

C. Insufficient Auditability

Audit trails, when available in consumer platforms, are often summarised rather than detailed and may not satisfy the granularity required by GDPR, HIPAA, or PCI-DSS. No reviewed accessible platform implements a hash-chained audit log that records every file access event with timestamp, user identifier, source IP address, and user agent in a tamper-

evident structure. This gap is particularly significant for healthcare, legal, and financial organizations that must demonstrate not only that data is protected but also that they can prove who accessed it, when, and from where.

D. Lack of File Integrity Enforcement

Without cryptographic hash verification, files can be altered in transit or at rest—whether through bit rot, malicious tampering, or storage corruption—without detection. No reviewed consumer platform surfaces SHA-256 hash verification to the end user or records integrity mismatches in an audit log. PhantomVox addresses this gap by computing and storing the SHA-256 hash of every uploaded file and recomputing it on every download, with integrity errors surfaced to both users and administrators and recorded in the tamper-evident audit log.

E. Poor Usability of Secure Sharing Workflows

When security features such as encryption and access controls are offered by existing platforms, they are often hidden behind complex menus, requiring users to be aware of cryptographic concepts that have no place in a daily workflow. This usability gap drives users toward less secure but more convenient channels, undermining the platform's security guarantees. PhantomVox addresses this by making secure behaviour the default: files are always encrypted, share links always expire, and audit events are always recorded without requiring any user configuration.

VI. PROPOSED SYSTEM: PHANTOMVOX

A. Conceptual Overview

PhantomVox is conceived as a third-generation secure file sharing platform directly addressing each of the five research gaps identified in Section V. The system is built on three foundational principles derived from the reviewed literature: (i) cryptographic secure defaults, where AES-256-GCM encryption, TLS 1.3, and TOTP-based MFA are applied unconditionally to every file and session; (ii) real-time integrity grounding in SHA-256 verification on every download, eliminating the risk of undetected file tampering; and (iii) tamper-evident accountability through a hash-chained audit log that records every meaningful action in the system.

The system accepts varied inputs across its six modules—user credentials, file content, share link configurations, and audit queries—and produces structured outputs including encrypted file storage, expiring share links, integrity-verified downloads, and audit log entries. Key capabilities include AES-256-GCM encryption with unique IV per file, TLS 1.3 enforcement at the reverse proxy layer, RBAC with three principal roles and per-file permission scoping, expiring share links with configurable download limits and optional passwords, immutable hash-chained audit logging, and SHA-256 integrity verification on every download.

B. Addressing Identified Gaps

The design of PhantomVox maps directly to each identified gap:

- Gap 1 (Unified E2E Encryption): Addressed through mandatory AES-256-GCM encryption at upload time—files are never persisted to disk in plaintext and keys are managed in a three-tier architecture separating master key, DEK, and session key.
- Gap 2 (Access Control Defaults): Addressed through the share-link module, which enforces configurable expiry between one hour and thirty days, download limits, and optional password protection on every generated link.
- Gap 3 (Auditability): Addressed through a hash-chained PostgreSQL audit log recording every login, upload, download, share creation, share use, and administrative action with timestamp, user identifier, IP address, and user agent.
- Gap 4 (File Integrity): Addressed through SHA-256 hashing computed at upload and recomputed at every download, with integrity errors surfaced to users and administrators and recorded in the audit log.
- Gap 5 (Usability): Addressed through a React 18 frontend with TailwindCSS that exposes all security features through accessible, jargon-free controls and provides immediate feedback for all operation outcomes.

C. Technical Innovation

The principal technical innovation of PhantomVox relative to prior work is the integration of six security modules under a unified React frontend with a Node.js/Express.js backend providing API routing, encryption orchestration, and audit management. This integration, combined with mandatory AES-256-GCM encryption, hash-chained audit logging, and RBAC enforcement on every request, represents a combination not previously implemented in an accessible, open-architecture student-level security platform that simultaneously satisfies GDPR and HIPAA auditability requirements.

VII. SYSTEM ARCHITECTURE

A. Architectural Overview

PhantomVox is implemented as a three-tier web application. The presentation layer is a React 18 single-page application served by the Express backend and fronted by Nginx for TLS termination. The application layer comprises six cohesive modules: Authentication, File Management, Encryption/Decryption, Sharing, Audit Logging, and Administration. The data layer consists of the PostgreSQL metadata database, the encrypted file store on disk or in an S3-compatible bucket, and the append-only audit log table.

B. Frontend Layer (React 18 + Vite + TailwindCSS)

The frontend is built with React 18 and Vite, styled with TailwindCSS. Six page-level components—Login, Dashboard, File Detail, Shares, Audit Log, and Admin—are supported by a library of reusable components for buttons, modals, tables, and form fields. State is managed with React Query, which provides caching, background refetching, and optimistic updates. Authentication state is held in a context provider that listens for token-expiry events and triggers refresh-token rotation transparently.

C. Backend Layer (Node.js 18 + Express 4)

The Express.js backend serves three primary functions: it exposes RESTful API endpoints for all file operations, authentication flows, and audit queries; it orchestrates encryption and decryption using the Node.js crypto module with AES-256-GCM; and it provides the file upload and download pipeline using streaming transforms so that files larger than available memory can be processed without buffering. The backend is stateless, enabling horizontal scaling without session management complexity. Rate limiting is applied to authentication endpoints and account lockout is triggered after five consecutive failed attempts.

D. Data Layer (PostgreSQL + Encrypted File Store)

The metadata database consists of five principal tables: users, files, shares, audit_logs, and refresh_tokens, plus auxiliary tables for role assignments and rate-limiting counters. The audit_logs table is append-only and hash-chained, with each entry incorporating the SHA-256 hash of the previous entry. Encrypted files are stored on the local filesystem or in an S3-compatible bucket, with the wrapped DEK, IV, and GCM authentication tag stored alongside the file metadata in the database.

E. Key Management Architecture

Key management is organized into three tiers. The master key, used only to wrap DEKs, is loaded from a dedicated secret source at startup and never written to disk. DEKs are generated per file, wrapped using the master key, and stored alongside file metadata. Session keys for TLS are managed by the TLS library and are short-lived. Master key rotation re-wraps all DEKs without re-encrypting underlying ciphertext, while DEK rotation requires re-encrypting the corresponding file. This architecture ensures that a compromise of the file store does not expose plaintext content without also compromising the master key.

VIII. DISCUSSION

A. Significance of the Proposed System

The primary significance of PhantomVox lies in its potential to democratise access to enterprise-grade secure file sharing for students, early-career security professionals, small organizations, and educational institutions that cannot afford commercial enterprise content management deployments. The system's tamper-evident audit logging is particularly significant: by generating hash-chained records of every file access event, the platform provides the raw material required for GDPR and HIPAA compliance audits without any post-processing or reformatting.

B. Feasibility Assessment

The technical feasibility of PhantomVox is supported by three observable facts. First, AES-256-GCM, TLS 1.3, TOTP, JWT, and PostgreSQL are all production-ready technologies with documented interfaces requiring no custom cryptographic implementation. Second, the Node.js and React ecosystem provides mature, well-maintained libraries for every security primitive used in the platform. Third, the platform is deployable as a set of Docker containers on standard cloud hosting, making it accessible to student project budgets without commercial infrastructure investment.

C. Educational and Societal Impact

At the educational level, PhantomVox provides a hands-on learning environment in which cybersecurity students can explore real cryptographic implementations, understand RBAC enforcement in a web application context, and develop practical incident response skills through audit log analysis. The platform's integration of industry-standard frameworks—AES-256-GCM, TLS 1.3, TOTP, JWT, RBAC, and hash-chained audit logging—exposes students to the tools and paradigms they will encounter in professional security

roles, bridging the well-documented gap between academic cybersecurity education and operational security practice.

IX. ADVANTAGES OF THE PROPOSED APPROACH

- Fills documented research gaps by integrating six security modules—encryption, transport security, MFA, RBAC, expiring share links, and audit logging—absent from any single reviewed accessible platform.
- Strong, modern cryptography applied uniformly without requiring user intervention: AES-256-GCM, TLS 1.3, and TOTP are non-negotiable defaults.
- Fine-grained access controls and expiring share links significantly reduce the risk of accidental data disclosure compared to consumer platforms.
- Comprehensive hash-chained audit logging provides accountability and supports regulatory compliance with GDPR and HIPAA.
- SHA-256 integrity verification detects file tampering that would otherwise go unnoticed in transit or at rest.
- Self-hostable architecture removes dependence on a third-party trust anchor for sensitive data.
- React 18 frontend with TailwindCSS provides an approachable, accessible interface that minimises friction without compromising security posture.
- Stateless backend architecture enables cost-effective deployment on standard cloud hosting without persistent session management infrastructure.

X. FUTURE SCOPE

- Client-Side Encryption: Encrypt files in the browser before upload so that the server never handles plaintext content, further reducing the trust placed in the server infrastructure.
- Blockchain Audit Anchor: Anchor audit-log checkpoint hashes on a public blockchain or transparency log for external, independently verifiable proof of integrity.
- Decentralised Storage Integration: Integrate with IPFS or other content-addressable storage for resilience and censorship resistance without requiring a centralised file store.
- Zero-Knowledge Proofs: Allow recipients to verify properties of a file such as its SHA-256 hash without revealing the file's content, enabling privacy-preserving integrity attestation.
- WebAuthn and FIDO2 Support: Extend MFA to support hardware security keys in addition to TOTP, providing phishing-resistant authentication for high-security deployments.
- AI-Based Anomaly Detection: Automatically detect unusual access patterns in the audit log and alert administrators to potential insider threats or compromised accounts.
- Granular Retention Policies: Automatically expire files and audit entries according to configurable retention rules aligned with GDPR right-to-erasure requirements.
- Native Mobile Application: A React Native companion providing push alert notifications, audit log monitoring, and rapid share-link management for on-call administrators.

XI. CONCLUSION

This review paper has presented a comprehensive and critical survey of the state of secure file-sharing research and standards, with the objective of contextualising PhantomVox within the existing body of knowledge. Eight significant research contributions and standards were reviewed spanning symmetric encryption, transport security, authentication, access control, audit compliance, and the open-source secure sharing landscape.

The analysis reveals consistent technical progress in individual security primitives yet identifies five persistent research gaps unaddressed by any single accessible reviewed system: the absence of unified end-to-end encryption in accessible tools, weak access control defaults, insufficient auditability, lack of file integrity enforcement, and poor usability of secure sharing workflows.

The proposed PhantomVox system addresses all five gaps through a coherent technical design combining AES-256-GCM encryption at rest, TLS 1.3 in transit, TOTP-based MFA, RBAC with three principal roles, expiring share links with download limits and optional passwords, SHA-256 integrity verification on every download, and a hash-chained append-only audit log. The system is technically feasible with current open-source libraries and standard cloud hosting infrastructure, and is designed to serve both educational and operational security purposes.

The work contributes to the broader discourse on secure file sharing by demonstrating that the gap between sophisticated security research and practical, accessible deployment is bridgeable through opinionated defaults, framework alignment, and user-centred design—without requiring custom cryptographic implementation, proprietary datasets, or enterprise-scale infrastructure investment.

REFERENCES

[1] NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode

(GCM) and GMAC, National Institute of Standards and Technology, 2007.
[2] NIST FIPS Publication 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, 2001.
[3] IETF RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3, E. Rescorla, Internet Engineering Task Force, 2018.
[4] IETF RFC 7519, JSON Web Token (JWT), M. Jones, J. Bradley, N. Sakimura, Internet Engineering Task Force, 2015.
[5] IETF RFC 6238, TOTP: Time-Based One-Time Password Algorithm, D. M'Raihi et al., Internet Engineering Task Force, 2011.
[6] IETF RFC 4226, HOTP: An HMAC-Based One-Time Password Algorithm, D. M'Raihi et al., 2005.
[7] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-Based Access Control Models," IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.
[8] OWASP Foundation, OWASP Application Security Verification Standard (ASVS) v4.0.3, 2021.
[9] OWASP Foundation, OWASP Top Ten Web Application Security Risks, 2021.
[10] Node.js Foundation, Node.js Crypto API Documentation, current release.
[11] OpenJS Foundation, Express.js Documentation, current release.
[12] PostgreSQL Global Development Group, PostgreSQL Documentation, version 14.
[13] Let's Encrypt, Certbot User Guide, Internet Security Research Group.
[14] European Parliament and Council, Regulation (EU) 2016/679 (General Data Protection Regulation), 2016.
[15] United States Department of Health and Human Services, HIPAA Security Rule, 1996.
[16] B. Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C, 2nd ed., John Wiley & Sons, 1996.
[17] D. Boneh and V. Shoup, A Graduate Course in Applied Cryptography, draft v0.5, 2020.
[18] Cloud Security Alliance, Security Guidance for Critical Areas of Focus in Cloud Computing v4.0, 2017.