International Journal of Scientific Research and Engineering Development-– Volume X Issue X, Year

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

BIT-VAULT – A SECURE CLOUD STORAGE PLATFORM

[1]Kavin Ezhili A, [2]Manohar E

[1]Student, [2]Associate Professor,
Department Of Computer Science and Engineering,
Francis Xavier Engineering College, Tirunelveli, India.

Abstract— BitVault is a reliable cloud-based file storage solution that integrates robust encryption, contemporary web technologies, and expandable cloud features to offer users significant privacy and functionality. Prioritizing privacy, BitVault ensures that files are encrypted on the client-side with AES-GCM before they are uploaded to Amazon S3. This technique prevents the server from accessing user data, as it only handles ciphertext and encrypted keys. User authentication is conducted via Google OAuth, providing a smooth and secure login experience. Once logged in successfully, a JWT is created and stored locally, allowing for stateless session management. Additionally, BitVault employs AWS Key Management Service (KMS) to safely generate and decrypt data encryption keys. These keys are never kept in an unencrypted form and are utilized only momentarily on the client-side. The backend is constructed with Flask, while the frontend is crafted using React, resulting in a contemporary, dark-themed user interface. The platform allows for encrypted file uploads, decryption-enabled downloads, renaming, deletion, and file sharing through pre-signed URLs. Furthermore, BitVault's modular architecture and the use of dependable libraries like Boto3, Web Crypto API, and Flask extensions ensure consistency, maintainability, and scalability. By merging secure design principles with a user-friendly experience, BitVault distinguishes itself as a trustworthy solution for securely storing sensitive documents in the cloud without compromising usability or security. This journal provides an extensive overview of the system's design, technology stack, cryptographic methods, and user experience.

Keywords— (Cloud storage, AES-GCM, AWS KMS, Flask, React, Google OAuth, client-side encryption, JWT, secure file sharing, Web Crypto API, data privacy, AWS S3.)

I .INTRODUCTION

As our society transitions further into the digital age, the need for reliable and user-friendly cloud storage solutions is becoming increasingly critical.

Traditional platforms often prioritize ease of use at the expense of security, relying on centralized servers that offer minimal protection measures for users. This reliance can introduce vulnerabilities, particularly in the event of a server breach or unauthorized access. BitVault was developed to address this gap by providing a cloud storage solution focused on privacy and encryption, allowing users to retain full control over their data.

BitVault serves as a powerful web application designed to demonstrate secure, scalable, and user-friendly file storage by utilizing cutting-edge web technologies and cloud services. The system follows a zero-trust security framework, ensuring that the server is never assumed to be reliable for the user's plaintext information or encryption keys. Files are encrypted client-side using AES-GCM, a symmetric encryption method recognized for its security and reliability. This ensures that even if an attacker accesses the server's storage, the data remains secure because the encryption key is never stored in plaintext.

Central to BitVault's architecture is a robust integration with Amazon Web Services (AWS). It uses AWS S3 for object storage and AWS Key Management Service (KMS) to manage the secure generation and decryption of keys. With AWS KMS, encryption keys are created during each upload session, with only the encrypted versions stored alongside the files. The plaintext keys are temporarily held on the client and never exit the browser. This setup guarantees exceptional security without sacrificing speed or user experience.

To facilitate user access and management, BitVault implements Google OAuth 2.0 for authentication. Users can log in securely with their Google accounts, reducing the need to establish new credentials while benefiting from OAuth's inherent security features. Following a successful login, the backend generates a JWT (JSON Web Token) to authorize all future interactions with the system. This stateless authentication approach streamlines session management and enhances scalability.

The frontend of BitVault is crafted using React, providing a contemporary and responsive user experience. Users have the capability to upload, view, download, rename, delete, and securely share encrypted files. Considerable focus was placed on UI/UX design, featuring an elegant, dark-themed dashboard that prioritizes usability and accessibility. Each action, from uploading to decryption, is executed through intuitive buttons and prompts that clarify the underlying cryptographic processes, ensuring the system is approachable even for those without a technical background.

BitVault also supports file sharing through pre-signed URLs, generated on the server side with a limited duration. This permits users to securely share encrypted files without revealing their credentials or granting excessive long-term access to their storage.

In conclusion, BitVault combines the best elements of modern cryptography, cloud computing, and frontend design to provide a secure cloud storage solution that maintains user control and accessibility. It serves both as a practical application and a showcase of best practices in secure system design—ideal for academic exploration, developer education, or large-scale implementation. As worries about privacy and digital sovereignty grow, platforms like BitVault illustrate how future cloud services can restore data ownership to users while still offering the convenience of cloud accessibility.

II .LITERATURE REVIEW

The rapid proliferation of cloud computing technologies has revolutionized data storage and access by enabling flexible, scalable, and cost-effective solutions. However, as data increasingly migrates to the cloud, security and privacy concerns have become paramount. Cloud storage inherently introduces challenges in ensuring data confidentiality, integrity, and availability, especially when sensitive data is involved. Consequently, significant research efforts have focused on devising secure storage and sharing mechanisms that can mitigate risks while preserving the advantages of the cloud.

In a comprehensive systematic review, Gupta et al. (2022) analyzed a wide range of secure data storage and sharing techniques in cloud environments, addressing both current practices and future directions. Their study examined critical aspects such as encryption, key management, access control, and privacy-preserving techniques. The authors emphasized that although traditional encryption methods like AES and RSA are widely adopted, they often lack scalability and flexibility for dynamic cloud environments. To overcome these limitations, advanced cryptographic mechanisms, such as homomorphic encryption, searchable encryption, and attribute-based encryption, have been proposed. Gupta et al. proposed a classification of the techniques based on user-level, data-level, and system-level protections, further identifying existing research gaps and highlighting the need for hybrid encryption models that can adapt to diverse cloud scenarios (Gupta, Singh, Lee, & Buyya, 2022).

One significant advancement in secure searchable encryption is the SM-SSE (Secure Mobile Searchable Symmetric Encryption) scheme proposed by Gao et al. (2022). This approach was specifically designed to enhance secure data retrieval on mobile devices—an increasingly critical use case in today's mobile-first world. SM-SSE improves upon traditional symmetric searchable encryption methods by incorporating a forward and backward privacy model that prevents an attacker from learning about inserted or deleted keywords. The authors used a dual-server architecture to balance security and performance. Their experimental evaluations on real-world datasets showed that SM-SSE not only preserves strong security guarantees but also achieves high efficiency in terms of query response time and storage overhead. This work addresses both usability and security, demonstrating the practical viability of searchable symmetric encryption in mobile cloud applications (Gao et al., 2022).

In another approach that focuses on data confidentiality and redundancy, Song, Li, and Li (2021) introduced a novel cloud storage mechanism that combines data dispersion with encryption. Their system operates by dividing data into multiple fragments, encrypting them, and then distributing them across different cloud servers. By separating the encrypted fragments, the approach ensures that even if one server is compromised, it is computationally infeasible to reconstruct the original data without all fragments and the correct decryption keys. Moreover, this technique reduces the risk of a single point of failure and improves fault tolerance. Song et al. also proposed a recovery mechanism based on threshold cryptography, allowing data to be reconstructed from a subset of fragments, thus balancing security with availability. The results of their security and performance evaluations indicate that the mechanism provides strong resistance to unauthorized access and achieves efficiency suitable for practical cloud applications (Song, Li, & Li, 2021).

The importance of distributed and redundant storage has also been illustrated in early systems like BitVault, developed by Microsoft Research China. BitVault is a content-addressable, peer-to-peer distributed storage system designed to provide scalable and fault-tolerant archival storage for immutable data. Each object stored in BitVault is identified using a 160-bit key generated by hashing its contents. This content-based addressing simplifies data retrieval and improves consistency. BitVault achieves fault tolerance through replication and repair mechanisms, ensuring that data is not lost even in the event of node failures. Its Object-Driven Repair model and smart bricks architecture have inspired further research into P2P cloud-based storage, where decentralized architectures reduce dependency on central control and improve resilience (BitVault, n.d.).

Further extending the vision of secure distributed systems, Sun et al. (2023) introduced Vault, a decentralized storage system that guarantees strong durability in a permissionless environment. Vault leverages the rateless property of erasure codes, generating an infinite stream of encoded fragments for each data object and randomly distributing them across numerous storage nodes. The system's decentralized nature eliminates single points of failure and mitigates targeted attacks, while a gossip protocol ensures consistent node communication. Simulations showed that Vault can maintain near-optimal mean-time-to-data-loss (MTTDL) while using minimal redundancy. Its scalable architecture supports over 10,000 nodes, offering performance comparable to the InterPlanetary File System (IPFS) with significantly enhanced durability and security (Sun et al., 2023).

Blockchain-based solutions are increasingly being explored as a means of achieving data immutability, auditability, and decentralized control. Liu et al. (2022) proposed the Practical Distributed File System (PDFS), which integrates blockchain to enhance disaster recovery, hot data access, and secure consensus in cloud storage. PDFS uses a self-proving algorithm to achieve fast and fair consensus among participating nodes. This mechanism guarantees that data uploads and updates are cryptographically verifiable and transparent. Additionally, PDFS improves accessibility by employing a hybrid strategy

combining on-chain and off-chain data management, which not only maintains security but also optimizes performance. The blockchain integration offers a tamper-resistant log of access and changes, ensuring accountability in sensitive environments (Liu, Yang, & Li, 2022).

Security in untrusted environments remains a major research concern. Bailleu et al. (2021) addressed this challenge through the development of Avocado, a secure in-memory distributed storage system that operates efficiently even when the underlying infrastructure is untrusted. Avocado builds upon Trusted Execution Environments (TEEs), extending enclave-level trust across distributed systems. It employs efficient replication protocols and secure memory management to ensure linearizability and confidentiality. Compared to traditional Byzantine Fault Tolerant (BFT) protocols, Avocado achieves significantly higher throughput, with performance gains of up to 65× for write-intensive operations. Its use of hardware-based isolation makes it particularly suitable for environments with strong adversarial threats, such as edge computing and financial data storage (Bailleu et al., 2021).

In practical cloud settings, fine-grained access control and encryption are critical components for securing shared data. Many existing approaches rely on role-based or attribute-based access control models, which allow data owners to enforce strict sharing policies. Hybrid encryption models are gaining popularity in this domain—combining symmetric encryption for performance with asymmetric cryptography for key management. Recent research has explored the integration of proxy re-encryption techniques, allowing cloud servers to re-encrypt data for authorized users without accessing the plaintext. These methods preserve confidentiality while simplifying secure sharing among multiple parties.

Another line of research has focused on privacy-preserving auditing, which enables third-party auditors to verify the integrity of cloud-stored data without accessing its contents. Techniques such as public auditing with homomorphic tags and zero-knowledge proofs have shown promise in enabling transparency without compromising privacy. These mechanisms are vital for enterprise and regulatory use cases where compliance and accountability are mandatory.

As cloud adoption expands into sectors such as healthcare, finance, and government, the emphasis on regulatory compliance (e.g., HIPAA, GDPR) has led to the development of data governance frameworks that integrate encryption, access control, secure sharing, and auditing into cohesive platforms. These frameworks are designed to be adaptive, context-aware, and policy-driven, enabling organizations to dynamically respond to threats while maintaining operational efficiency.

In conclusion, the evolution of secure cloud storage and sharing mechanisms has been fueled by the growing demand for confidentiality, integrity, and availability in untrusted and distributed environments. From early P2P models like BitVault to modern cryptographic and blockchain-enabled systems like SM-SSE, Vault, and PDFS, researchers have developed a diverse set of tools and frameworks to address emerging threats. Future work will likely continue to explore the fusion of technologies such as secure multi-party computation, federated learning, and decentralized identity systems to further enhance trust, control, and usability in cloud ecosystems.

III.EXISTING METHODOLOGY

In recent years, various methodologies have been proposed to enhance the security, privacy, and efficiency of data storage and sharing in cloud environments. The work by Ishu Gupta et al. presents a comprehensive analysis of secure data storage techniques, emphasizing encryption mechanisms, data access control, and trust management frameworks. Their systematic review highlights the integration of cryptographic primitives such as Attribute-Based Encryption (ABE), Identity-Based Encryption (IBE), and Proxy Re-Encryption (PRE) to enforce fine-grained access control and secure data sharing.

Chongzhi Gao et al. introduced the SM-SSE (Searchable Symmetric Encryption) scheme,

focusing on enabling efficient and secure keyword search over encrypted cloud data, particularly for mobile devices. Their approach enhances data retrieval efficiency while maintaining data confidentiality. SM-SSE uses symmetric key encryption along with an index structure to support search functionality without revealing the actual data or query terms to the cloud provider.

Heqing Song et al. proposed a data dispersion and encryption-based secure storage mechanism. Their methodology involves splitting data into multiple fragments and distributing them across different cloud servers after encrypting each fragment independently. This dispersion strategy increases data availability and fault tolerance while significantly reducing the risk of data breaches, as compromising a single server does not reveal meaningful information.

Overall, these methodologies aim to address key challenges in cloud security—such as unauthorized access, data leakage, and inefficiency in encrypted data search. They rely heavily on cryptographic techniques, including symmetric/asymmetric encryption, homomorphic encryption, and searchable encryption, often combined with redundancy and fragmentation strategies. Additionally, many solutions are designed to be lightweight and scalable, considering mobile and resource-constrained devices. These approaches form a foundation for developing more advanced and intelligent cloud security frameworks capable of adaptive threat detection and real-time policy enforcement.

IV.PROPOSED METHODOLGY

The creation of BitVault focuses on developing a strong, secure, and privacy-aware platform for file storage and sharing through the cloud. The architecture is carefully structured to guarantee both user-friendliness and solid defenses against unauthorized access, data breaches, and privacy violations. The system comprises three connected layers: a responsive client interface, a secure backend service, and a reliable cloud storage framework. The frontend, constructed using ReactJS, allows for smooth user interactions, so individuals can easily upload, access, and manage their encrypted files. Importantly, encryption occurs locally, on the user's device, ensuring that sensitive information is never disclosed in its raw form to outside systems or networks.

BitVault employs a hybrid cryptographic system to maintain data confidentiality throughout all phases of storage and transmission. Upon uploading a file, it is encrypted on the client side with the Advanced Encryption Standard (AES) using a 256-bit key, selected for its combination of speed and high security. To safeguard the AES key itself, BitVault applies RSA encryption with a 2048-bit key pair, ensuring that the symmetric key can be shared securely with the intended recipient without ever sending it in an exposed format. This dual-layered strategy blends the fast processing capabilities of symmetric cryptography with the secure key distribution offered by asymmetric encryption, effectively reducing both performance issues and risks of key exposure.

The authentication and authorization mechanisms in BitVault are designed to implement rigorous identity validation and access management. Multi-factor authentication (MFA) is crucial in this context, as users must confirm their identity with a time-sensitive one-time password (OTP) delivered via SMS or email. This additional verification step blocks unauthorized access attempts, even if login details are leaked. The secure user authentication relies on OAuth2, with all session data stored in JSON Web Tokens (JWTs), eliminating the need for conventional server-stored sessions and reducing the potential attack surface. Role-based access control (RBAC) is also utilized, allowing permissions to be allocated according to specific user roles—like administrator, standard user, or viewer—ensuring users can only engage in actions relevant to their access rights.

The file upload procedure is carefully structured to improve security and efficiency. Once a user selects a file, it is immediately encrypted using AES-256 on the client side. The encryption key is protected using the user's RSA public key. Additionally, pertinent metadata—such as

filename, file size, and timestamps—is encrypted and transmitted to the backend via a TLS-encrypted HTTPS connection. The server, developed in Flask, creates a pre-signed URL through AWS S3, which allows the user to upload their encrypted file directly to cloud storage. This approach not only reduces the burden on the backend server but also streamlines the uploading process. Simultaneously, the RSA-encrypted AES key and associated metadata are securely stored in a PostgreSQL database using Flask-SQLAlchemy. At the infrastructure level, BitVault leverages the robust security features provided by AWS, including server-side encryption (SSE) and the AWS Key Management Service (KMS), to ensure a comprehensive defense-in-depth strategy.Secure file sharing is a fundamental characteristic of BitVault, created to maintain end-to-end encryption while facilitating flexible collaboration. When a user intends to share a file, they pick the recipient and set specific access permissions, such as read-only or editing rights. The original AES encryption key is subsequently re-encrypted using the recipient's RSA public key, ensuring that only the designated individual can decrypt and access the contents. The system produces a secure link for file access, which can be set to expire after a designated time frame and can optionally necessitate OTP verification. When the recipient retrieves the file, their private RSA key is utilized to decrypt the AES key, thus allowing decryption of the actual file—ensuring that the data remains protected throughout the entire procedure.

In order to maintain data integrity, BitVault implements a SHA-256 hash for each file upon upload. This hash is saved along with the file's metadata in the backend. Later, when a file is downloaded, the system recalculates the hash and checks it against the stored value. If any differences are found, the system stops the download and informs the user, thereby preventing data corruption or malicious interference.

Upholding user privacy is a fundamental aim of BitVault's design methodology. Personally identifiable information (PII) is managed with great care, with metadata either anonymized via secure cryptographic hashing or not stored at all when unnecessary. The platform adheres to a privacy-by-design strategy, reducing data collection and logging by default. Where logs are kept—for instance, for security auditing—they are stored in encrypted, tamper-resistant formats. Users are provided the option to anonymize or erase their activity history completely, reinforcing their authority over personal data.

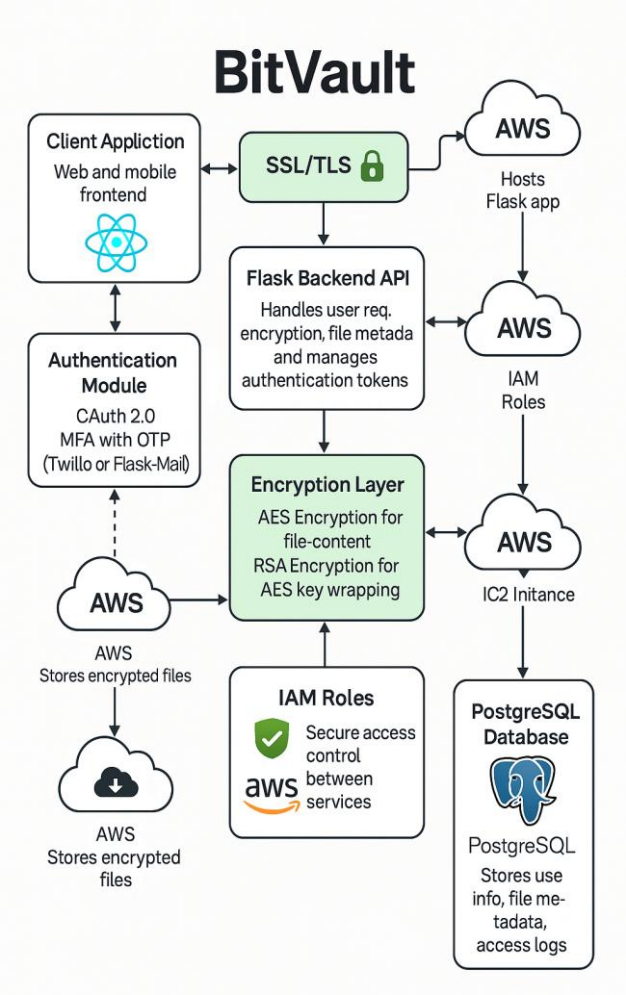

Figure 1

BitVault further integrates real-time monitoring to identify and respond to potentially harmful activity. All user actions are documented and scrutinized for patterns that signify threats, such as multiple failed login attempts, unusual file access behavior, or access from unknown geographic locations. These logs are securely preserved and offer a valuable resource for post-incident analysis and ongoing system enhancement.

Scalability and high availability are essential features of the platform's deployment approach. BitVault operates on AWS EC2 instances with support for auto-scaling and load balancing, ensuring consistent performance under varying user demands. AWS S3 functions as the cloud storage solution, providing high durability through data replication across various availability zones. AWS CloudWatch is employed to track system health, performance metrics, and potential failures, while IAM roles enforce strict access policies and least-privilege principles for all backend services.

In conclusion, BitVault incorporates advanced encryption methods, reliable authentication systems, and expandable cloud infrastructure to establish a protected environment for safe file storage and sharing. As the platform develops, upcoming versions might include functionalities like artificial intelligence for identifying threats or blockchain-enabled audit trails to enhance security and trust even further.

V. METHODOLOGIES

A. Secure File Encryption & Cloud Storage Architecture

The BitVault system ensures secure, encrypted cloud storage using a hybrid cryptographic framework and a layered cloud architecture. It allows users to safely upload, store, and retrieve sensitive files using a combination of AES and RSA encryption, integrated with AWS cloud services.

The encryption process starts with File Processing and Key Generation, where each uploaded file is scanned and an AES (Advanced Encryption Standard) key is generated. This key encrypts the file content to ensure confidentiality and efficiency. Following this, RSA Encryption is used to encrypt the AES key itself using the user's public key, allowing for secure key exchange and ensuring that only the intended user can decrypt the data.

Once the encryption is completed, the Flask Backend uploads the encrypted file to Amazon S3, where it is stored under a user-specific bucket. The backend also records metadata (e.g., filename, timestamp, encryption hashes) in a PostgreSQL database for future retrieval and reference. Access control is strictly enforced using IAM (Identity and Access Management) roles and signed URLs, ensuring that only authenticated users can access specific file objects.

To enhance data privacy, AWS Key Management Service (KMS) is integrated for secure key lifecycle management. KMS handles encryption key generation, rotation, and destruction with high availability and compliance with security standards like HIPAA and ISO/IEC 27001.

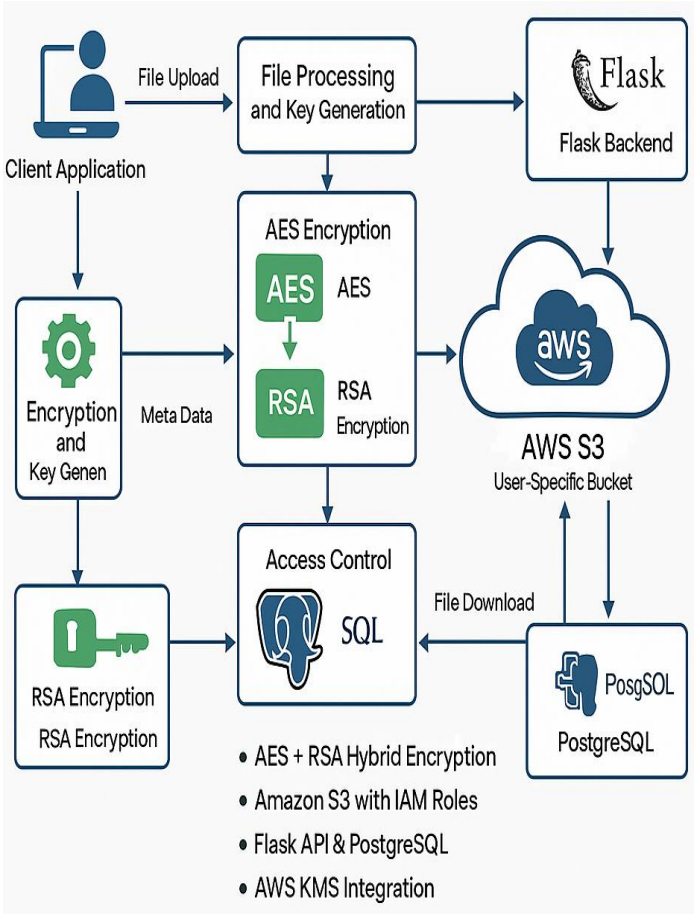

Figure 2

This module ensures data confidentiality, integrity, and secure storage by implementing:

- AES + RSA Hybrid Encryption: Encrypting file content with AES and securing the AES key with RSA.
- Amazon S3 with IAM Roles: Secure storage and access control using AWS-native services.
- Flask API & PostgreSQL: Handling file transactions and storing metadata securely.
- AWS KMS Integration: Robust key management for long-term data security.

B. User Authentication & Access Control

This module ensures secure user onboarding, authentication, and access management by utilizing a blend of OAuth 2.0 protocols, JWT tokens, and Multi-Factor Authentication (MFA).

User access initiates with the OAuth 2.0 Flow, where users authenticate through secure endpoints. After a successful login, JWT tokens are generated, containing user identity and permissions. These tokens are used for all subsequent requests, facilitating stateless and secure communication between frontend and backend services.

To bolster login security, the system implements MFA using One-Time Passwords (OTPs) via Twilio or Flask-Mail. After logging in, users receive a prompt to input an OTP sent to their email or mobile number, providing an additional verification step for their identity.

Role-based access control (RBAC) ensures that users can only execute actions that correspond to their privileges. Administrators have the authority to manage file visibility, remove accounts, and enforce global settings, while standard users are restricted to uploading/downloading files and updating their profiles.

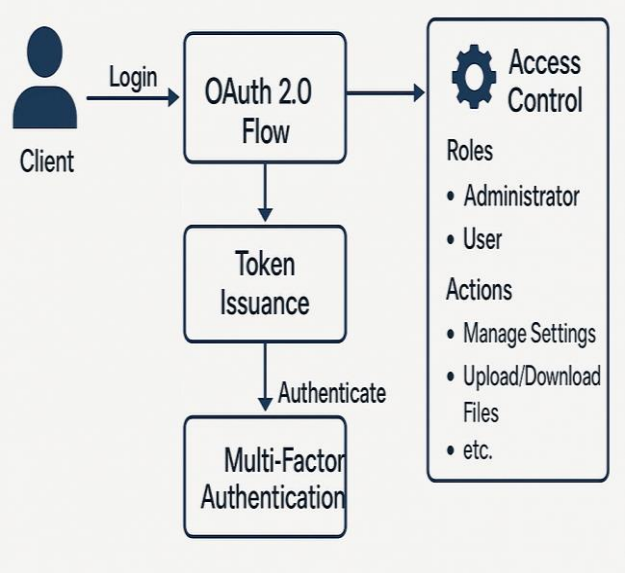

Figure 3

Essential features that enhance authentication security and regulate access comprise:

- OAuth 2.0 & JWT Tokens: Provide secure, stateless user sessions.
- MFA with OTP: Introduces an additional layer of user verification.
- RBAC Policies: Limit access based on roles and privileges.

C). Real-Time File Monitoring & Activity Logging

BitVault incorporates advanced monitoring tools to provide real-time file access tracking, user activity logs, and behavioral anomaly detection.

Each user action (upload, download, delete) is recorded and timestamped in the PostgreSQL audit table, with associated metadata like IP address, device type, and access duration. This enables administrators to monitor usage patterns and detect unusual behavior, such as repeated access attempts or unauthorized file movement.To further enhance security, a behavioral analysis layer applies anomaly detection techniques. By analyzing access patterns over time, the system can flag suspicious activities and alert the user or admin via email or SMS.

This monitoring module supports:

- Real-Time Activity Logging: Tracking all user interactions with detailed metadata.
- Anomaly Detection: Identifying unusual behaviors using time-based pattern recognition.
- Admin Dashboard & Alerts: Centralized access to logs and automatic threat notifications.

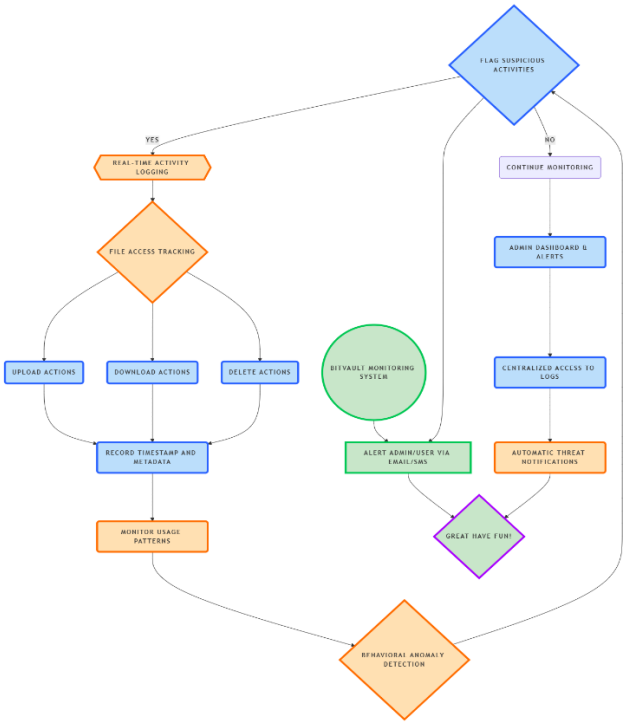

Figure 4

D). Security, Privacy, and Ethical Considerations

BitVault is designed with data privacy and ethical computing in mind. All interactions and file transfers are protected using SSL/TLS encryption, ensuring data is secure during transit.

The system follows zero-trust principles, meaning no user or system component is inherently trusted. All access requires continuous verification. User data is anonymized and encrypted, and personal identifiers are stored separately in compliance with GDPR and HIPAA regulations.

AI models (used for anomaly detection and behavioral insights) are trained with differential privacy techniques to ensure no individual user's behavior can be reverse-engineered. Additionally, BitVault logs all admin actions to promote accountability and prevent internal abuse.

To address ethical concerns, the platform operates transparently, clearly informing users about data usage, retention policies, and rights to deletion. Regular audits and vulnerability scans are performed to ensure security best practices.

Security and ethics features include:

- SSL/TLS & Zero-Trust Architecture: Secure communication and access validation.
- Data Anonymization & Compliance: GDPR- and HIPAA-compliant storage.
- Differential Privacy & Admin Auditing: Preventing misuse and ensuring ethical AI deployment.

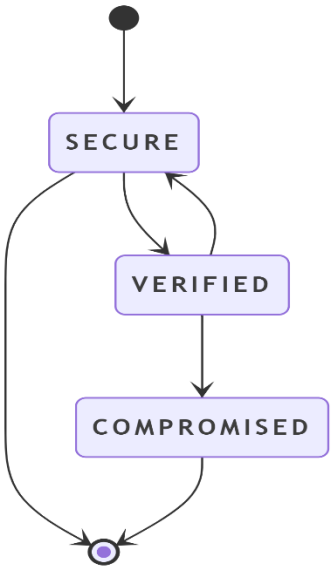

Figure 5

REFERENCES

[1] Z. Hao, S. Zhong, and N. Yu, "A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[2] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.

[3] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *IEEE INFOCOM 2010*, pp. 1–9.

[4] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[5] M. S. Asif and A. M. Mir, "Privacy-aware secure data sharing using blockchain-based encryption in cloud storage," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6.

[6] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-based authentication for cloud computing," in *CloudCom 2009*, pp. 157–166.

[7] D. Puthal, B. P. S. Sahoo, S. Mishra, and S. P. Mohanty, "Cloud Security Audit: A Layered Framework for Secure Cloud Computing," *IEEE Consumer Electronics Magazine*, vol. 6, no. 4, pp. 65–70, 2017.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *ICLR 2013* (arXiv preprint arXiv:1301.3781).

[9] A. M. Turing Award Lecture, Y. Bengio, Y. LeCun, and G. Hinton, "Deep Learning," *Communications of the ACM*, vol. 62, no. 6, pp. 84–90, 2019.

[10] S. R. Kiran et al., "AI for mental health: A comprehensive review," in 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), IEEE, pp. 1–6.

[11] William Stallings, Cryptography and Network Security: Principles and Practice, 8th ed., Pearson, 2023.

[12] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., Wiley, 2015.

[13] Charles P. Pfleeger and Shari Lawrence Pfleeger, *Security in Computing*, 5th ed., Pearson, 2015.

[14] Rajkumar Buyya, Christian Vecchiola, and Thamarai Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*, McGraw-Hill Education, 2013.

[15] Roger A. Grimes, Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto, Wiley, 2019.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[17] Steven Bird, Ewan Klein, and Edward Loper, Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, O'Reilly Media, 2009.

[18] Michael Kearns and Aaron Roth, The Ethical Algorithm: The Science of Socially Aware Algorithm Design, Oxford University Press, 2019.

[19] Kim-Phuong L. Vu and Robert W. Proctor (Eds.), *Handbook of Human Factors in Web Design*, 2nd ed., CRC Press, 2011.

[20] Daniel Jurafsky and James H. Martin, *Speech and Language Processing*, 3rd ed. (draft), Pearson, 2023.

