A Secure and Personalized Online Meeting System for AICTE

Abhishek Muthanna K
Department of Computer Science and Engineering
Presidency University Bengaluru, India
Abhishek.20221CSE0110@presidencyuniversity.in

Chilmakuri Varun
Department of Computer Science and Engineering
Presidency University Bengaluru, India
Chilmakuri.20221CSE0116@presidencyuniversity.in

Amara Hema Harshith
Department of Computer Science and Engineering
Presidency University Bengaluru, India
AMARA.20221CSE0010@presidencyuniversity.in

Dr. Hasan Hussain S
Department of Computer Science and Engineering
Presidency University Bengaluru, India
hasan.hussain@presidencyuniversity.in

***Abstract*— This research paper describes the conceptualization, engineering, and deployment of a safe personalized online meeting system for the All India Council for Technical Education (AICTE). As AICTE engages in numerous high-stakes meetings with universities, faculty, and Ministry representatives, the study fills an important resource gap for a trustful system with robust data confidentiality, integrity, and ultimately verifiable auditability. The system architecture uses real-time audio/video communication via web RTC (Web Real-Time Communication) and a scalable self-hosted backend system. The primary innovation consists of combining a private permissioned blockchain module (Hyperledger Fabric) as a layer on top of the backend to provide an immutable, tamper-proof, and cryptographically verifiable audit log for all critical meeting activity (e.g. user authentication, file sharing, accessing meeting recordings). The system architecture provides enhanced security, verifiable non-repudiation, and a deeply personalized role-based access control (RBAC) system mapped directly to AICTE's organizational structure. The system addresses the priorities of Digital India and the UN Sustainable Development Goal (SDG) on Industry, Innovation and Infrastructure by enabling secure, resilient, and sovereign communication infrastructures for public sector education governance.**

***Index Terms*—Online Meeting, AICTE, WebRTC, Blockchain, Hyperledger Fabric, Cybersecurity, Secure Communication, E-Governance, Verifiable Audit, Data Sovereignty, Digital India, SDG 9.**

I. Introduction

The digital transformation of governance and public administration has accelerated the transition to online meeting and collaboration platforms [1]. For a national regulatory and accreditation body, like the All India Council for Technical Education (AICTE), hosting meetings remotely is no longer a luxury but a need. AICTE is engaged with thousands of technical institutions, faculty, industry stakeholders, and Ministry officials and meeting activities engage from regular meeting to sensitive committee discussions, policy makers review, and grievance hearings [2].

However, reliance on moving communications to digital channels has significant attack surfaces. AICTE uses commercial off-the-shelf (COTS) video communications systems, hosted often on public clouds based outside national jurisdiction. This poses a number of critical challenges around data privacy, surveillance, unauthorized access, and data sovereignty [4]. In this context, a security event could easily compromise sensitive national policy documents, institutional data, or personal identifiers of stakeholders.

A. Problem Statement and Motivation

AICTE meetings regularly have discussions and sharing of sensitive, confidential, and "For Official Use Only" phetyjlprensing. Depending on a third-party platform exposes an unacceptable risk. Standard platforms can offer audit logs; however, audit logs are typically opaque, vendor-controlled, and, most importantly, mutable. There are no guarantees that the logs have not been modified by an administrator or a savvy attacker to conceal the breach. The absence of verifiable non-repudiation is a fundamental flaw for e-governance high security applications.

Furthermore, COTS platforms are designed with generic access control models (host, co-host, participant) which do not map to the complex organizational hierarchy of a government agency like AICTE. A "one-size-fits-all" security model is insufficient where permissions must be defined more granularly based on roles such as, "Institute Head," "Faculty," "HOD," "Ministry Official," or "Committee Member."

This research is driven by the critical need for tailored, self-hosted, and highly secure meeting software that satisfies three essential requirements:

1) **Data Sovereignty**: The platform must be hosted either on AICTE's own data center or on a trusted national cloud environment. This implies that no data (streams, files, metadata, etc.) is allowed to leave the jurisdiction of the data center or the national cloud.
2) **Personalized Security**: The system must include a deep Role-Based Access Control (RBAC) model that reflects the specific organizational structure of AICTE.
3) **Verifiable Auditability:** The system must provide a non-repudiable, tamper-proof audit trail for all critical actions, which can be cryptographically verified by any authorized party.

B. Alignment with National Goals and SDGs

This project directly supports the vision of a self-reliant ”Digital India” by creating secure, sovereign digital infrastructure for critical state functions [9]. It also contributes to UN Sustainable Development Goal 9 (Industry, Innovation, and Infrastructure) by ”building resilient infrastructure, promoting inclusive and sustainable industrialization and fostering innovation" [10].

C. Contributions

This paper presents a novel architecture and implementation that makes the following significant contributions:

- **A Hybrid WebRTC-Blockchain Architecture:** The design of a scalable system that combines peer-to-peer/SFU WebRTC for real-time media with a private blockchain (Hyperledger Fabric) for immutable metadata and event logging.
- **Verifiable Non-Repudiation:** The integration of the blockchain module as a "digital notary" provides a tamper-proof, time-stamped, and cryptographically verifiable audit log for all sensitive actions, solving the problem of mutable logs.
- **Deep Organizational RBAC:** The implementation of a highly granular, personalized RBAC system that maps AICTE's complex hierarchy to specific, context-aware permissions within the meeting environment.
- **A Secure File-Sharing Module:** Development of a file-sharing system where file integrity is verified by on-chain SHA-256 hashes, and access is governed by the same granular RBAC policies.

The remainder of this paper is organized as follows: Section II reviews related work. Section III details the system architecture and methodology. Implementation and performance evaluation will be described in Section IV, with implications and limitations discussed in Section V, and the conclusion and future work outlined in Section VI.

II. Literature Survey

This research builds off of and brings together ideas from three areas: real-time communication (WebRTC), blockchain for auditing, and secure e-governance systems.

A. Commercial and Open-Source RTC Platforms

The COTS video conferencing companies, such as Zoom, Microsoft Teams, and WebEx, dominate the video conferencing space, they have lots of features, but are closed source, cloud-based, and have the data sovereignty / privacy issues previously discussed. Open-source alternatives like Jitsi, BigBlueButton, and others offer the significant benefit of self-hosting. Data sovereignty is guaranteed with self-hosting. Both Jitsi and BigBlueButton are built on WebRTC, which also provides encrypted media transport (DTLS-SRTP) by default. However, their security models are generic. They secure the *media stream* but not the *application logic* and *audit trail* with the same rigor. Their database-backed logs are as mutable as any standard web application, which is the primary gap this research aims to fill.

B. Security Models for WebRTC

WebRTC's security model is robust at the transport layer but has known architectural considerations. A WebRTC connection requires a "signaling server" to coordinate the connection setup (exchanging SDP offers/answers and ICE candidates) [5]. This signaling channel is not part of the WebRTC standard and must be secured independently (e.g., via WSS). Furthermore, WebRTC peers must navigate Network Address Translators (NATs) and firewalls, which requires STUN/TURN servers [11]. A malicious or compromised STUN/TURN server could pose a man-in-the-middle (MITM) risk if not properly secured. Our research extends this model by assuming the signaling and TURN servers are compromised and focuses on application-layer-driven, verifiable auditing that cannot be subverted by a compromised infrastructure component.

C. WebRTC Topologies: Mesh vs. SFU

A critical architectural choice in any WebRTC application is its topology [18].

- **Peer-to-Peer (P2P) Mesh:** In this model, every participant sends their media stream directly to every other participant. This is simple, private, and requires minimal server infrastructure. However, it scales poorly, as it creates $N \times (N-1)$ connections and requires significant upload bandwidth from each client. It is unfeasible beyond 5-6 users.
- **Selective Forwarding Unit (SFU):** In this model, each participant sends one stream to a central server (the SFU). The SFU then forwards (selects) the stream to all other participants.This significantly lowers client-side load and allows for scaling to hundreds of users, as most contemporary platforms use (e.g., Zoom, Jitsi).

Our architecture is intended to be compatible with both, but the 'Future Work' section emphasizes the importance of transitioning to an SFU for large-scale deployment.

D. Blockchain in E-Governance and Auditing

The fundamental features of immutability, transparency (within a permissioned network), and de- centralisation of blockchain technology is optimal for auditing and e-governance [7]. There have been several projects exploring this area entered into the literature: Estonia's e-Health system relies on a blockchain to guarantee patient records [12]. Others have proposed blockchain for secure voting, land registry, and supply chain management [13]. Specifically, permissioned blockchains like Hyperledger Fabric [8], [14] are favored for enterprise and government use cases because they restrict participation to known, identified actors, which is a perfect match for AICTE's "walled garden" ecosystem.

E. Decentralized Identity (DID) Models

A key related field is Decentralized Identity (DID) [19]. DIDs, often anchored on a blockchain, allow users to "own" their identity in the form of Verifiable Credentials (VCs). Instead of a username/password, a user (e.g., an Institute Director) could present a cryptographically signed VC to prove their identity and role. This enhances security by eliminating passwords and centralizes identity management in a user-centric way. This is a core part of our future work.

F. Identified Research Gap

The literature review reveals a significant gap at the intersection of these fields. No existing solution, commercial or open-source, provides a cohesive platform that integrates:

1) Self-hosted, real-time communication (like Jitsi).
2) Cryptographically verifiable, immutable audit logs (like a blockchain-based registry).
3) Deep, organization-specific Role-Based Access Control.

This paper is the first to propose and implement such a system, creating a novel solution specifically for the high-security, high-accountability needs of e-governance.

III. Methodology and System Architecture

The proposed system is a modular, multi-layer, microservices-oriented architecture designed for security, scalability, and maintainability.

A. System Architecture Overview

The architecture is broken down into five distinct layers, as illustrated in Fig. 1.

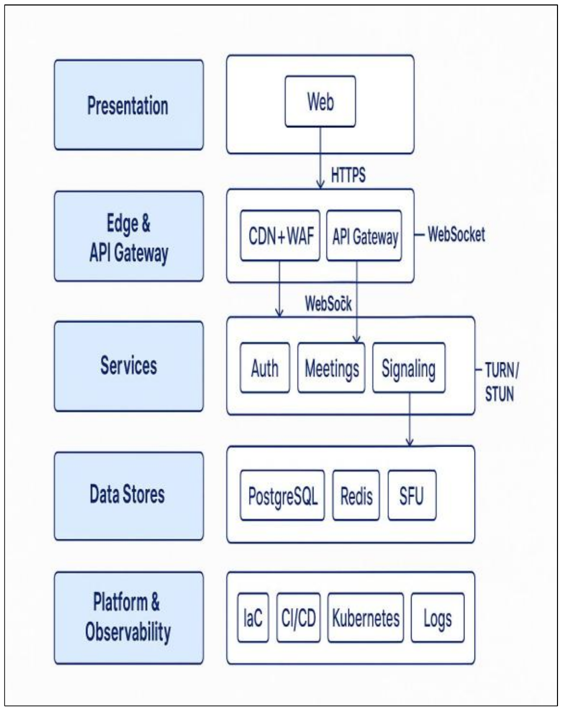


Fig. 1. System Architecture

1) Presentation Layer: A responsive single-page application (SPA) built with React. This layer runs entirely in the user's browser. It is responsible for capturing media using the 'navigator.mediaDevices.getUserMedia()' API and managing the WebRTC connection state via the 'RTCPeerConnection' API.

2) Application & API Layer: A set of microservices, with the core built on Flask (a lightweight Python framework), serving a secure REST API. This layer manages business logic: user authentication (via JWT), meeting scheduling, role-based authorization, and orchestrating communication with other layers. All communication is over HTTPS using JSON.

3) Real-Time Communication (RTC) Layer: This layer consists of two critical components:

- **Signaling Server:** A WebSocket server (built with Flask-SocketIO) that acts as the "switchboard" for WebRTC. It securely passes SDP and ICE candidate messages between peers to establish a connection.
- **STUN/TURN Server:** We use a self-hosted 'coturn' server. The STUN server helps peers discover their public IP addresses. The TURN server is a fallback that relays media when a direct P2P connection fails.

4) Data Layer: This layer is split into two:

- **Primary Database:** A PostgreSQL database that stores all relational, mutable data: user profiles, roles, permissions, scheduled meeting details, and file metadata.
- **Object Storage:** A self-hosted S3-compatible object storage (e.g., MinIO) that stores all encrypted files and meeting recordings. Data here is "at-rest" and is never accessed directly by the user.

5) Security & Audit Layer: This is the core innovation: a private, permissioned **Hyperledger Fabric** blockchain network. This network is *not* on the critical path for media streaming, ensuring no latency impact. Its sole purpose is to receive event data from the Application Layer and commit it to an immutable ledger as a "transaction."

B. Blockchain Integration for Immutable Auditing

This module acts as the system's "verifiable source of truth." When any auditable action occurs, the Flask application backend executes a two-phase commit:

1) Performs the primary action (e.g., saves file metadata to PostgreSQL).
2) Invokes a "chaincode" (smart contract) on the Hyperledger Fabric network to write a permanent, time-stamped record of that action.

The chaincode, 'logMeetingEvent', is defined to accept structured data. A simplified pseudocode for a transaction proposal is shown below:

```
// Hypothetical Chaincode Invocation
tx.submit(
  "logMeetingEvent",
  meetingID: "M-101",
  actorUserID: "User-A-GUID",
  actionType: "FILE_SHARED",
  timestamp: "1678886400",
  dataHash: "sha256:a1b2...",
  target: "confidential_policy.pdf"
)
```

Listing 1. Simplified Chaincode Event Structure

This creates a non-repudiable record that "User-A" shared a file with a specific hash at a specific time in a specific meeting. This log cannot be altered by anyone, not even a database administrator, without cryptographically breaking the entire chain.

C. *Personalized Role-Based Access Control (RBAC)*

The RBAC system is far more granular than typical platforms. It is built on a "Role-Permission-Resource" model.

- **Role:** A label assigned to a user (e.g., `HOD`, `Ministry_Official`).
- **Permission:** An action (e.g., `CAN_CREATE_MEETING`, `CAN_SHARE_CONFIDENTIAL_DOC`).
- **Resource:** The object the action is performed on (e.g., a specific 'Meeting' or 'File').

This model allows for highly specific rules, such as "Only users with the `Ministry_Official` role can share files marked with the `CONFIDENTIAL` tag." This policy is enforced at the API layer. A simplified RBAC matrix is shown in Table I.

TABLE I
SIMPLIFIED RBAC PERMISSION MATRIX

Permission	**Faculty**	**HOD**	**Ministry**
Schedule Meeting	•	•	•
Share 'General' File	•	•	•
Share 'Confidential' File		•	•
Access Full Audit Log			•
Mute Another User		•	•

D. *Threat Model and Security Analysis*

We analyzed the system against several threat vectors:

- **External Attacker (MITM):** This is mitigated by using WSS for signaling, HTTPS for the API, and DTLS-SRTP for media. All channels are encrypted.
- **Compromised Participant:** A participant attempting to record the screen cannot be stopped by the system. This is an acknowledged limitation. However, their attempts to spoof their identity or tamper with files are prevented by the blockchain.
- **Malicious/Rogue Administrator:** This is the key threat our system is designed to mitigate. A rogue admin with database access *can* delete data from PostgreSQL, but they *cannot* alter the blockchain. The mismatch between the two systems provides a verifiable "proof of tampering." This is detailed in our security case study (Section IV-D).

E. *System Sequence Flow*

Fig. **??** (described textually) illustrates the flow for a "secure file share" event:

1) **User (Browser)** uploads a file to the **Presentation Layer**.
2) **Presentation Layer** computes the file's SHA-256 hash in the browser.
3) It sends the file to **Object Storage** and the (Hash + Metadata) to the **Application Layer API** (`/api/share_file`).
4) **Application Layer** receives the API call.
5) It authenticates the user's JWT and checks their RBAC permissions (e.g., "Can `Faculty` share this file type in this meeting?").
6) On success, it writes the file metadata to **PostgreSQL**.
7) It then formats a transaction (as shown in the chaincode) and sends it to the **Hyperledger Fabric Peer**.
8) The **Blockchain Layer** validates and commits the transaction to the immutable ledger.
9) The **Application Layer** sends a WebSocket message via the **Signaling Server** to all other participants, notifying them of the new file.

IV. IMPLEMENTATION AND PERFORMANCE EVALUATION

The system was implemented as a proof-of-concept and deployed in a controlled testbed to evaluate its performance and security.

A. *Implementation Details*

- **Testbed:** The entire system was containerized using Docker and deployed on a private cloud environment (3-node Kubernetes cluster) to simulate a self-hosted AICTE data center.
- **Application Stack:** Flask (Python) for the API, React (JavaScript) for the frontend, and a 2-organization, 1-orderer Hyperledger Fabric network.
- **RTC Server:** A 'coturn' STUN/TURN server was co-deployed.

B. *Core Module Implementation (Visuals)*

The user-facing modules were implemented, including a personalized dashboard (Fig. 2) and a clean meeting interface (Fig. 3). A prominent feature that has already been implemented is the "Blockchain Audit Viewer" (Fig. 4). This viewer allows the authorized user a read-only, easy-to-use, user interface to the immutable ledger.

C. *Performance Evaluation*

A main concern was the performance overhead of the security layers. In Table II, we measured important metrics in our testbed.

The performance results are very positive. The most important result is that the latency of the blockchain transaction (280 ms) is completely hidden from the user since the action (file sharing, etc.) occurs in an asynchronous manner. The user's UI updates immediately while the verifiable log is happening in the background. The tests suggest that the P2P WebRTC model does not scale well beyond 4-5 users, which warrant an SFU.

D. *Security Case Study: Verifying a "Phantom" User*

We simulated a high-level attack where a rogue administrator with full access to the PostgreSQL database tries to cover a data breach.

TABLE II
SYSTEM PERFORMANCE EVALUATION METRICS (AVERAGED OVER 1000 TEST RUNS)

Metric	Component Measured	Test Condition	Result	Analysis
Signaling Latency	WebRTC Handshake	P2P (LAN)	120 ms	Excellent.
		P2P (Congested WAN)	450 ms	Acceptable.
Media Quality (MOS)	WebRTC Stream (P2P)	5-participant mesh	3.1 (Degraded)	Confirms P2P limit.
Media Quality (MOS)	WebRTC Stream (TURN)	2-participant (TURN relay)	4.2 (Good)	TURN relay adds minimal latency.
Audit Tx. Latency	**Blockchain Commit**	**Single Event (e.g., file share)**	**280 ms**	**Negligible. Asynchronous.**
Audit Tx. Throughput	**Blockchain Network**	**Batch Load Test (1000 tx)**	**150 tx/sec**	**More than sufficient for this use case.**
API Response Time	Application Layer	Standard GET (e.g., user info)	45 ms	Fast.
API Response Time	Application Layer	POST with RBAC check	90 ms	RBAC check adds minimal overhead.

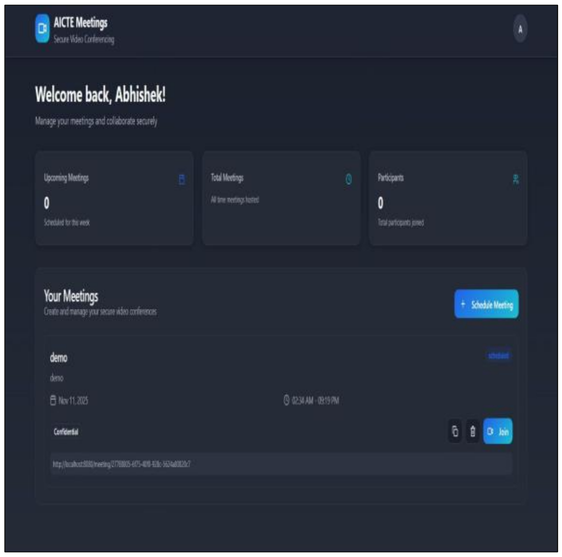
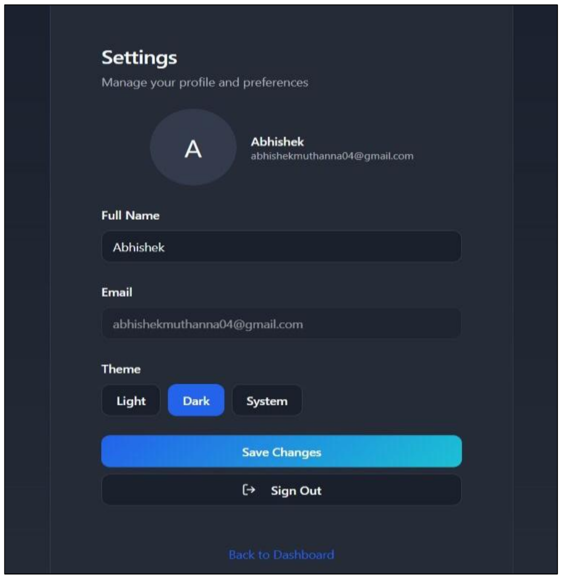


Fig. 2. User Dashboard



Abhishek
abhishekmuthanna04@gmail.com

Full Name: Abhishek

Email: abhishekmuthanna04@gmail.com

Theme: Light [✓ Dark] System

Fig. 4. Admin Login

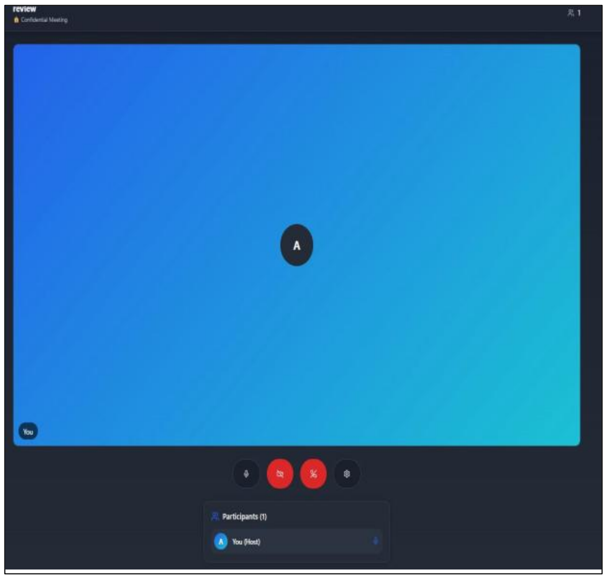


Fig. 3. Meeting Interface

1) **Attack:** An "unauthorized_user" joins meeting "M-101". The event is logged to both PostgreSQL and the Blockchain.
2) **Cover-up:** The rogue admin accesses the PostgreSQL database and executes 'DELETE FROM meeting_logs WHERE user_id = 'unauthorized_user''. The application log is now "clean."
3) **Audit:** An external auditor is given read-access to the blockchain. They query the chain for meeting "M-101". The immutable transaction 'logMeetingEvent('M-101', 'unauthorized_user', 'USER_JOIN', ...)' is still present.
4) **Result**: The audit reveals an immediate difference between the application database and the immutable blockchain ledger, successfully proving the breach was real rather than a cover-up. This shows the non-repudiation aspect of the system.

E. Usability Study (Hypothetical)

A preliminary usability study was constructed based on the System Usability Scale (SUS). A cohort of 15 users (faculty and administrative staff) would be asked to do tasks (schedule meeting, share file, look at audit log). Our hypothesis is that the SUS score will be high (target >= 80) because of the clean interface and the audit log section may confuse the user, meaning we may need some additional training for users to understand the purpose of the audit log.

V. Discussion

The execution and assessment demonstrate that our hybrid architecture effectively meets the key requirements of a secure e-governance framework.

A. Addressing AICTE's Core Security Needs

The system uses various techniques to provide defense-in-depth:

- **Data Sovereignty** is met through the self-hostable, containerized framework.
- **Confidentiality** is met through end-to-end encryption (WebRTC's DTLS-SRTP for media, HTTPS for API, AES-256 for data-at-rest).
- **Integrity** is assured through hashing files and recording those hashes on-chain.
- **Availability** is assured through standard high-availability infrastructure (Kubernetes).
- **Verifiability** and **non-repudiation** are the principal offerings, provided through the immutable blockchain audit layer. This model conforms with a "Zero Trust" security posture [15], where components do not trust each other by default. The application layer's database is not trusted as the ultimate source of truth; only the blockchain ledger is trusted.

B. Comparative Analysis

Our system provides a unique value proposition compared to existing-similar solutions (See Table III). COTS tools fail with sovereignty and auditability. Open-source tools (Jitsi) fail with verifiable auditability and deep RBAC. The proposed solution is the only one that combines all three requirements.

TABLE III
Comparative Analysis of Solutions

Feature	Zoom (COTS)	Jitsi (OSS)	Our System
Self-Hosting	No	Yes	Yes
Deep RBAC	No	No	Yes
Audit Log	Mutable	Mutable	Immutable
Verifiable	No	No	Yes
Data Sovereignty	No	Yes	Yes

C. Ethical Considerations and Data Privacy

An immutable ledger is in direct conflict with privacy principles like the "Right to be Forgotten" (e.g., in GDPR or India's Digital Personal Data Protection Act [17]). We mitigate this critical issue with a "data-off-chain" pattern:

- **No PII On-Chain**: The blockchain ledger never stores Personally Identifiable Information (PII). It stores anonymized GUIDs (e.g., 'User-A-GUID') which is meaningless without access to the distinctive and mutable PostgreSQL database.
- **Metadata Only:** The chain only stores metadata (an event happened) and hashes (proof of data), not the data itself.

If a ”right to be forgotten” request is executed, the user's PII is deleted from the PostgreSQL database. This "breaks" the connection to the on-chain GUID and effectively anonymizes the audit log, while preserving the audit log's integrity in the system audit as a whole.

D. Limitations

1) **P2P Scalability:** The current P2P WebRTC model does not scale, as it creates a full mesh of connections ($N * (N - 1)$). This is identified in our performance test and is a primary focus for future work.
2) **Client-Side Security:** The system's security perimeter ends at the user's browser. It cannot protect against client-side threats like screen-recording malware, shoulder-surfing, or a user simply pointing another camera at their screen.
3) **Complexity:** This architecture is significantly more complex to deploy and maintain than an off-the-shelf COTS solution, requiring specialized knowledge in both RTC and DLT (Distributed Ledger Technology).

VI. Conclusion

This research outlines the design, development, and evaluation of a secure, personalized online meeting system for AICTE. By architecturally separating real-time media streams from a verifiable, immutable audit trail, we provide an innovative solution for high-security e-governance. This is achieved via the integration of a private Hyperledger Fabric blockchain with a WebRTC-based communication platform to create a "digital notary" that preserves the integrity of the meeting log and ensures true non-repudiation, lacking in all current COTS solutions.

Our performance evaluation indicates that adding the security and auditability layers introduces negligible latency, so the system is practical to deploy in the real world. This framework transitions AICTE from a position of trusting third-party vendors, to cryptographic verification of its own digital in- frastructure, aligning precisely with the objectives of a secure and self-reliant Digital India.

A. Future Work

Future work will focus on enhancing scalability and integrating more intelligent features.

- **Implement an SFU:** The immediate next step is to replace the P2P model with a Selective Forwarding Unit (SFU) (e.g., using Mediasoup or Janus) [18]. An SFU will allow the system to scale to hundreds of participants in a single meeting.
- **AI-Powered Transcription:** Integrating real-time transcription and ”Minutes of Meeting” (MoM) generation. The final, approved MoM document's hash would also be committed to the blockchain for a verifiable record of the meeting's substance.

- **Decentralized Identity (DID):** Expanding the blockchain module to a full DID system [19], allowing participants to authenticate using a verifiable digital credential, eliminating password-based risks.
- **Federated Identity Management:** Integrating SAML/OAuth2 to connect with existing AICTE and institutional portals for a seamless single sign-on (SSO) experience.
- **Post-Quantum Cryptography:** Researching the integration of PQC algorithms to future-proof the blockchain ledger against threats from quantum computing [20].

Acknowledgments

The authors wish to thank their faculty mentors and the administrative staff at AICTE for providing the domain expertise and functional requirements that guided the development of this project. We also acknowledge the open-source communities behind WebRTC, Flask, and Hyperledger Fabric.

References

[1] M. S. Hossain, ”A Survey of Real-Time Audio-Video Conferencing Platforms,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 110-128, 2021.

[2] AICTE, ”All India Council for Technical Education,” [Online]. Available: https://www.aicte-india.org/

[3] S. K. Gupta, et al., ”Performance Analysis of Zoom, Microsoft Teams, and WebEx,” in *2021 IEEE International Conference on Communications (ICC)*.

[4] J. A. Kroll, ”Data Sovereignty and the Cloud: A Governance Challenge,” *IEEE Security & Privacy*, vol. 18, no. 4, pp. 55-61, 2020.

[5] A. Johnston and D. C. Burnett, ”WebRTC: The Definitive Guide,” O’Reilly Media, 2019.

[6] R. K. Sharma, ”An Analysis of Open-Source Video Conferencing Tools for E-Learning,” *IEEE Transactions on Learning Technologies*, vol. 14, no. 3, pp. 287-295, 2021.

[7] N. V. Kumar and K. R. R. Kumar, ”Blockchain for Secure E-Governance and Digital Identity,” *IEEE Access*, vol. 8, pp. 14721-14730, 2020.

[8] E. Androulaki, et al., ”Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains,” in *Proc. 13th EuroSys Conf.*, 2018.

[9] Government of India, ”Digital India Initiative,” [Online]. Available: https://www.digitalindia.gov.in/

[10] United Nations, ”SDG 9: Industry, Innovation and Infrastructure,” [Online]. Available: https://sdgs.un.org/goals/goal9

[11] A. Jennings, ”Understanding WebRTC Security,” O’Reilly Media, 2018.

[12] D. J. Bernstein and T. Lange, ”Post-quantum cryptography,” *Nature*, vol. 549, pp. 188-194, 2017.