AMERICAN UNIVERSITY OF BEIRUT

Project 68

A Permissioned System for Documenting Chain of Custody Using Hot and Cold Blockchain & Jump Servers

by

Rami Eid, CCE
Omar Kaaki, CCE
Jad Awad, CCE
Mostafa Jammoul, CCE

Advisor: Dr.Hussein Bakri

A final year project
submitted in partial fulfillment of the requirements
for the degree of Bachelor of Engineering
to the Department of Electrical and Computer Engineering
of the Maroun Semaan Faculty of Engineering and Architecture (MSFEA)
at the American University of Beirut
Beirut, Lebanon

Acknowledgments

We would like to express our sincere gratitude to Dr. Hussein Bakri for his invaluable guidance, constructive feedback, and consistent support throughout the preparation and ongoing development of this project.

This proposed system institutes a legally cognizable chain-of-custody framework for digital evidence, addressing admissibility, authenticity, and integrity requirements across key jurisdictions. By enforcing forensic soundness, continuous custody, and tamper-evident logging at each handling stage, it permits digital exhibits to be treated ipso facto as reliable before the trier of fact, preserving their probative value against challenges to alteration or provenance.

Lebanon — Law No. 81 of 2018 (Electronic Transactions), Arts. 122–123. Evidence must remain unaltered and every transfer must be recorded; the proposed system issues tamper-evident, timestamped custody records for each seizure, retention, and transfer event.

United States — Fed. R. Evid. 901(a) and 902. Evidence must be authenticated, typically through provenance and a documented chain-of-custody; the proposed system assigns unique identifiers and maintains an immutable audit trail to substantiate authenticity and integrity under these rules.

International — CoE e-Evidence Guidelines (2018); CoE Cybercrime Convention (2001) Art. 21; ICC/UN practice. Electronic evidence must be authentic, intact, and supported by uninterrupted custody; the proposed system employs cryptographic hashes, controlled transfers, and auditable preservation procedures to maintain integrity and traceability across borders.

Executive Summary

Digital evidence has become central to modern criminal investigations. Since traditional chain of custody systems offer only limited security guarantees, there is an urgent need for tamper proof documentation that ensures integrity, availability, and legal admissibility. Blockchain technology offers the potential to help forensic investigators with immutable, auditable, and cryptographically secured evidence handling from crime scene to courtroom.

The project will design a permissioned blockchain based chain of custody system to advance evidence management workflows and security guarantees; provide integrity, availability, and confidentiality; ensure adherence to the law and operational efficiency. The general problem from the reviewed literature seems to be a lack of scalable and privacy oriented blockchain systems for digital forensics. Existing technologies do not address hot/cold chain separation, TEE-based confidentiality, or court oriented workflows and are not integrated with off chain storage solutions. There are two major limitations to this research. Technically, the system requires SGX capable hardware, sufficient computational resources, and infrastructure for mTLS, which limit deployment flexibility and testing scope. Ethically, the system is subject to strict standards regarding the admissibility of evidence and privacy regulations, Lebanese Law No. 81/2018, U.S. Federal Rules of Evidence 901/902, and CoE guidelines, and needs to be assured through cryptographic guarantees, role based access control, and audit trails.

The proposed system addresses these challenges by incorporating four components:

- **Dual-Chain Architecture:** Hyperledger Fabric networks segregate active investigations (known as the hot chain) from archived cases, known as the cold chain, to achieve scalability.

- **JumpServer Gateway:** Secure access point with multi factor authentication and role-based access control for investigators, auditors, and court users.

- **TEE-based Private Chaincode:** Intel SGX enclaves executing Fabric Private Chaincode for performing confidential verification operations and sensitive data protection.

- **Private IPFS Cluster:** Off chain storage for large forensic artifacts, maintaining cryptographic hash links on chain for integrity verification.

We have conducted a comprehensive literature review, designed the dual chain architecture with appropriate organizations and consensus mechanisms, and developed operational prototypes for each component. The hot and cold Fabric networks were instantiated with LabOrg, CourtOrg, and OrdererOrg using RAFT ordering services, and certificate authority infrastructure became operational. JumpServer gateway was deployed with MFA and mTLS authentication, and public chaincode for custody operations was implemented and tested. The private IPFS cluster was configured and integrated with blockchain metadata layers. All these confirm the technical feasibility of the project and set a strong foundation for the next semester in terms of full integration, optimization, security validation, and deployment testing.

Contents

List of Figures

List of Tables

Glossary

Blockchain	A distributed ledger technology that records transactions across multiple nodes in an immutable, cryptographically-secured chain of blocks.
CA (Certificate Authority)	A trusted entity that issues digital certificates to verify the identity of users and devices in a network.
Chaincode	Smart contract code deployed on Hyperledger Fabric blockchain that executes business logic and manages ledger state.
Chain of Custody (COC)	A documented process that tracks the collection, transfer, and storage of evidence from crime scene to courtroom.
Cold Chain	A blockchain channel dedicated to storing archived or closed investigation cases for long-term immutable storage.
Cryptocurrency	A digital currency secured by cryptography that operates on blockchain technology without central authority control.
DFIR	Digital Forensics and Incident Response - the practice of investigating and responding to cybersecurity incidents and digital crimes.
Digital Evidence	Electronic data stored or transmitted in digital form that can be used as evidence in legal proceedings.
Enclave	A hardware-isolated secure memory region (like Intel SGX) that protects code and data from unauthorized access.
Ethereum	A public blockchain platform featuring smart contracts that enable decentralized applications and programmable transactions.

GUID	Globally Unique Identifier - a 128-bit number used to uniquely identify information while preserving anonymity.
Hot Chain	A blockchain channel handling active investigations with frequent transactions and real-time evidence updates.
Hyperledger Fabric	A permissioned blockchain framework designed for enterprise use with modular architecture and private transactions.
IPFS	InterPlanetary File System - a decentralized, peer-to-peer storage network that uses content-based addressing via cryptographic hashes.
Jump Server	A hardened gateway server (bastion host) that provides secure, controlled access to internal network resources.
MFA (Multi-Factor Authentication)	A security mechanism requiring two or more independent verification methods to authenticate user identity.
MITM (Man-in-the-Middle)	A cyberattack where an attacker intercepts and potentially alters communication between two parties without their knowledge.
MSP (Membership Service Provider)	A Hyperledger Fabric component that manages identities and enforces access control through certificate validation.
mTLS (Mutual TLS)	A security protocol requiring both client and server to authenticate each other using digital certificates.
Oracle	A third-party service that connects blockchain smart contracts with external real-world data sources.
Peer	A network node in blockchain that maintains a copy of the ledger and validates transactions.
PKI (Public Key Infrastructure)	A framework for managing digital certificates and public-key encryption to secure communications.
RBAC (Role-Based Access Control)	A security model that restricts system access based on predefined user roles and permissions.

SGX (Software Guard Extensions)	Intel's hardware technology that creates isolated execution environments (enclaves) for protecting sensitive code and data.
Smart Contract	Self-executing code deployed on blockchain that automatically enforces agreement terms when conditions are met.
TEE (Trusted Execution Environment)	A secure, isolated hardware environment that protects code execution and data from external tampering.
TrustZone	ARM's hardware-based security technology that creates isolated secure execution environments on mobile processors.

Note: Throughout this document, glossary terms are hyperlinked. Click on any highlighted term to return to this glossary for its definition.

Chapter 1

Introduction

Digital evidence is the core basis of investigations, litigation, and legal compliance. They are formed from the vast amount generated through smartphones, laptops, servers, and IoT devices. In order for this evidence to be legally admissible in court or any other formal process, its data should adhere to a secure and well documented COC. Traditional systems, paper based and centralized digital COC, face vulnerabilities that cannot be overlooked, being susceptible to loss and damage and having a single point of failure, respectively. This project proposes the utilization of a blockchain-based COC system to address these vulnerabilities. By depending on a distributed ledger, each custody event is kept track of through the use of cryptographic security throughout the decentralized network. The system is composed of two tiers: hot blockchain for active investigations and data; and a cold blockchain for long term immutable storage. In addition, we make use of IPFS to store digital evidence securely across many nodes, while the blockchain keeps cryptographic links. As an outcome of this dual tier system, we achieve authenticity, scalability, and reliability of evidence management.

1.1 Motivation

Digital forensics in our day and age has become an essential component in aiding law enforcement, where the criminal cases involving them have exceeded 90% [1]. In the year 2024 solely, FBI reported 859,532 cybercrime complaints leading to $16.6 billion in losses [2]. This underscores the need for a tamper proof management system to ensure COC authenticity for court admissibility.

Problems with Traditional Methods. Current COC systems reveal vulnerabilities that cannot go unnoticed. Centralized databases create single points of failure which can be taken advantage of as evidently shown by the 2022 DEA portal breach [3]. Moreover, paper-based systems are insecure and unreliable due to the risk of loss, damage, and unauthorized alteration. Thus, there is a need for cryptographic protection that rids of these gaps.

The blockchain's architecture is inherently built on being secure, it achieves 5 critical cybersecurity features: integrity (cryptographic hash chains detect tampering), availability (distributed storage eliminates single points of failure), confidentiality (permissioned access control), authentication (digital signatures verify handler identity), and non-repudiation (immutable ledger entries).

Previous research and implementation using a standard blockchain architecture failed due to unbounded ledger growth (storing massive volumes on data on chain). This project proposes a new innovative two-tier architecture: a hot blockchain for active cases and a cold blockchain for archived cases. Large evidence files reside in the IPFS while cryptographic hashes and custody metadata reside on-chain [4].

This research aims to support and be in the service of groups involved in DFIR, them being Law enforcement, defense attorneys, judges, forensic auditors, and expert witnesses. Each of these stakeholders relies on secure, verifiable, and tamper-evident evidence records, in which our system achieves through decentralized storage, role-based access control, and immutable audit trails.

1.2 Desired Needs

Stakeholders require a secure and tamper-proof environment to manage digital evidence. Traditional methods lack multiple security aspects, which may not satisfy the clients' desires. Thus, this project addresses those obstacles by introducing an integrated security system composed of various components, each designed to tackle a specific security concern. The project provides a secure gateway, the jump server, that acts as a single trusted entry point in which the user authenticates themselves using a certificate issued by the server's CA after a multi-factor authentication (MFA) login. This utilization of certificates and MFA over passwords eradicates the risk of password cracking, ensuring the protection of the users. Moreover, prioritizing a gateway instead of direct access reduces the attack surface by isolating the main storage components from the rest of the system. Furthermore, this project implements a strict role-based access control system (RBAC) that is validated by the admin. This RBAC component differentiates users according to their supported roles: Court, Auditor, and Investigator. After authentication, the gateway offers a user-friendly interface based on the authenticated role, which limits the user's capabilities to only what their role permits, preventing any unauthorized access. Users can access their respective dashboard to either upload digital evidence securely to the IPFS, view the uploaded evidence, or review blockchain entries; however, all these actions are strictly governed by their role, as only an investigator can add evidence, while auditors and court users can only view. Once an upload is complete, the jump server automatically logs the uploaded evidence and hashes its content to ensure that evidence is protected and verifiable. All interactions are logged to prevent users from denying actions and occur over an encrypted channel to protect their confidentiality, with optional support for anonymity through GUID-based identifiers that conceal the investigator's real identity.

1.3 Business Aspect

The digital forensics market reached $12.94 billion in 2025 and is projected to exceed $22.81 billion by 2030 [5]. Despite this growth, current chain of custody systems have critical flaws that compromise evidence integrity and court admissibility, two of the most important aspects of our implementation. Existing solutions generally fall into three categories, each with fundamental weaknesses that our project scope addresses.

Blockchain-based systems, while theoretically tamper-proof, suffer from practical limitations in some aspects. Ethereum-based implementations provide for anonymity and transactions are public for everyone on the network, but "it is nearly impossible to infer who were involved in these transactions" [6], making them unsuitable for investigations and digital forensics.

Centralized commercial platforms dominate the market but create single points of failure and vendor lock-in, while paper-based systems cannot scale to modern digital evidence volumes.

To the best of our knowledge, no current solution simultaneously addresses court admissibility, operational scalability (¿1000 TPS), and multi-stakeholder access without centralized trust dependencies. The European Union's €7 million LOCARD project concluded after three years that "the technology has to evolve to overcome current technological limitations, such as interoperability, scalability, and privacy " [7].

Our two-tier Hyperledger Fabric architecture with IPFS storage and JumpServer access control directly addresses these gaps by combining blockchain integrity with operational efficiency, role-based access control with cryptographic verification, and distributed storage with court-admissible audit trails.

Chapter 2

Technical Background

2.1 Introduction

Digital forensic is a wide field composed of various branches that focus on evidence acquisition, analysis, and preservation. With the evolution of technology, digital evidence has taken shape in many forms, ranging from simple documents to terabytes of hard drives. Additionally, chain-of-custody documents require a safe, tamper-proof storage in order to maintain the integrity if the collected evidence in court. For this reason, blockchain technology has emerged as a promising solution. However, existing blockchain implementations for chain-of-custody have faced critical challenges in different aspects. Therefore, this chapter will discuss existing approaches and identify gaps in their designs and literatures, while focusing on this project's design choices. Figure 2.1 below will guide the reader through different paths: Path A focuses on scalability and hot and cold implementations, path B emphasizes the Jumpserver mechanism, while path C explores the chain-code of the blockchain. All paths will lead to a gap analysis section, where our our solution's benefits and advantages over the existing papers will be explained.

2.2 Blockchain Based Chain-of-Custody

Digital forensics has began evolving from the early 1990's, where paper-based COC implementations were used as a primary method for tracking evidence and verifying the work of investigators [8]. However, such methods vulnerable to external damage and suffer from storage issues, such as the systematic process of storing, tracking, and moving evidence [9], in addition to the lack of integrity of the printed chain-of-custody documents [10]. By 2009, cryptocurrencies, such as

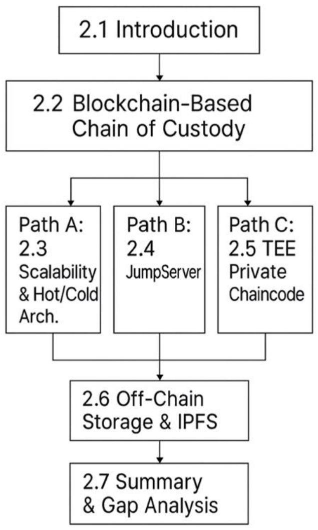

2.1 Introduction
2.2 Blockchain-Based Chain of Custody
Path A: 2.3 Scalability & Hot/Cold Arch.
Path B: 2.4 JumpServer
Path C: 2.5 TEE Private Chaincode
2.6 Off-Chain Storage & IPFS
2.7 Summary & Gap Analysis

Figure 2.1: A block diagram of chapter 2 sections showing different reading paths, each with a different focus.

Bitcoin, have become a booming topic in the market [11], which lead to the utilization of blockchains in multiple industries. This introduction of blockchains has inspired forensic companies to explore blockchains for chain-of-custody transactions [12]. Therefore, the following subsections will examine existing blockchain-based COC systems and identify any potential limitation.

2.2.1 Early Ethereum Based Approaches

Back in 2019, Bonomi et al. [13] have proposed a blockchain implementation for COC, making it one of the earliest proposals for such an idea. Their system uses a permissioned Ethereum network where evidence custody events are recorded as transactions on a dematerliazed ledger, which is a public blockchain platform. A smart contract component is triggered every time a custody transfer occurs to validate the records based on pre-defined rules which are responsible for logging the events. The authors implement this COC system using an emulator (EVM) that runs on a virtual resource that is known as "gas". Each EVM instruction consumes gas, which allows the authors to measure different aspects, such as gas consumption and latency. Their results demonstrate that permissioned Ethereum can handle moderate forensic workloads (2-5 seconds per transaction) and manageable storage growth (hundreds of MB per thousands of evidence).

Despite their contributions to the DFIR field, their COC implementation exhibits a couple of weak points:

1. Off-Chain Identity verification: users are authenticated by an external party, not by the blockchain. In case of compromise, this external entity can become a single point of failure, which potentially may grant unwanted access or deny authorized users from using the blockchain.

2. Public blockchain dependencies: despite being permissioned, this system still relies on Ethereum's public chain architecture. In other words, even in controlled and closed deployments, the system still uses the same transaction model and execution environment as the public Ethereum network, where anyone can participate. These unnecessary public-chain additions can make the solution less efficiently introducing overhead, which conflicts the requirements of blockchain COC technologies.

On the other hand, Lone and Mir [14] presented another Ethereum based COC system. Their proposed design maintains integrity, authenticity, and audibility of digital evidence as it moved through different stages of a forensic investigation. This system also uses a smart contract to record events such as evidence acquiring and transferring. The author describes a workflow where the first responder takes a cryptographic hash of digital evidence, along with its metadata, and records it securely on the blockchain through a smart contract. During the succeeding transfers, the smart contract automatically logs the user's address, current state of evidence, permission level, and timestamp.

Despite satisfying the primary requirements of the forensics process, this design still suffers from a couple limitations:

1. Off-Chain Identity verification: similar to the previous paper, this design handles user verification, encryption, and access permissions outside the blockchain.

2. Large evidence storage: this paper also doesn't address the off-chain storage of large digital artifacts, such as RAM dumps and CCTV footage, but only focuses on the handling of chain-of-custody records.

3. No implementation: this paper focuses heavily on the theoretical aspect of an Ethereum based blockchain, with no physical implementation.

2.2.2 Transition to Hyperledger Based Approaches

After their paper on the Ethereum platform, Lone and Mir [15] decided to experiment with the Hyperledger technology. In 2019, an alternative to Ethereum was proposed due to its recognized limitations. The shift to a permissioned blockchain has introduced several improvements, making the Hyperledger technology more advantageous. Instead of relying on external identity providers, this system uses blockchain identities to map user identities to their respective participant role (investigators, judges, Prosecutors, etc...) and stores this information in an "Identity Registry" within the blockchain itself. This eliminates the previous vulnerability of external single points of failure, in addition to adding an RBAC system managed by the Hyperledger Composer, a framework that facilitates building blockchain applications. Moreover, identity management operations, such as assigning new roles and users, are carried out by admin peers owned by what is called "consortium organizations", which are organizations that join together to collaborate on a shared project. This decentralizes the process of identity authority across multiple organizations rather than entrusting a single external provider. Additionally, this implementation utilizes a smart contract to encode its logic, which is referred to as a "chaincode". According to this article, the chaincode is responsible for evidence creation, along with its hash and metadata, evidence transfer to the blockchain, and evidence display. Although this proposal was a huge improvement from traditional blockchain implementations, it still suffered from different limitations. The project was implemented using a Hyperledger Composer, which later on suffered a deprecation in 2019 where the project was no longer supported [16], as Linux decided to discontinue the framework to move developers toward native Fabric SDK, which is used in recent designs. Additionally, storing handling was a common problem amongst all papers so far, as the authors did not mention how storing large evidence would be handled, as using the blockchain as a storing unit may result in reduced performance.

Another interesting implementation is proposed by Nischitha et al [17]. This system addresses Lone and Mir's issue by introducing native Fabric SDK and JavaScript chaincode instead of the Hyperledger Composer. The project relies on a CA for handling identity management instead of the Composer's Identity Registry, where each law enforcement agency operates its own CA server. The server is responsible for issuing certificates for each user, providing them with a cryptographic key pair and a set of permissions, creating an RBAC system. As a result, user authentication is handled through a public key infrastructure (PKI). Furthermore, this system implements "Membership Service Providers", or MSPs, to store the public keys of all users belonging to law enforcement agencies. For instance, when a user tries to perform a transaction, the MSP (org1MSP, org2MSP) will validate their private key signature using their stored public keys.

This system, however, still lacks some key components:

1. Off-Chain Storage: this paper doesn't address the off-chain storage system, but only focuses on storing metadata (case ID, investigator name, etc...).

2. Single Channel Usage: this article highlights the usage of one single channel for all participant law enforcement agencies, which means that all case data

are placed on the same ledger.

2.3 Scalability and Hot/Cold Architectures

Over time, the accumulation of ever-growing numbers of evidence entries and custody transactions represents a fundamental scalability challenge that must be addressed for blockchain-based COC systems. By design, permissioned blockchains like Hyperledger Fabric store the complete transaction history of all participating peers. Every custody transfer, evidence upload, and metadata update remains permanently stored on the ledger. Although such behavior ensures that evidence is immutable, performance bottlenecks occur as the blockchain grows in size. For instance, latency during queries increases, and storage requirements grow exponentially, resulting in slower network synchronization as peers must validate and replicate increasingly large ledgers. This is especially limiting in forensic environments where months- or years-long investigations generate thousands of transactions per case. Taking into consideration the single-channel architecture where all active and archived cases share the same ledger, the inevitable system performance degradation hinders real-time forensic operations.

To address such scalability concerns, several researchers have proposed the segregation of active and archived evidence by proposing a hot/cold chain architecture. Kim et al. [18] proposed a two-level blockchain system for digital crime evidence management using Hyperledger Fabric. The proposed architecture segregates the blockchain into two channels: a ”hot chain” for active investigations and a ”cold chain” for closed or archived cases. The hot blockchain retains investigation and identity information with frequent fluctuations in transactions, while the cold blockchain retains only assigned data, such as videos of criminal evidence. After an investigation is closed, evidence data is migrated from the hot chain to the cold chain in order to ensure that no degradation of performance occurs during active investigations due to historical data accumulated over time. The segregation leads to the generation of blocks that cannot be deleted or modified and are shared by all peers in the blockchain system, enhancing its transparency and reliability. The authors implemented this on a Hyperledger Fabric framework using Docker containers to generate multiple peers, authentication servers, orderers, and blockchain state databases.

Despite the promising results, the system developed by Kim et al. has a number of limitations. First, their archival mechanism lacks comprehensive auditability: once the cases are moved to the cold chain, their system does not provide efficient mechanisms for cross-referencing related evidence across both chains. For instance, when a closed case needs to be reopened or when the archived evidence becomes relevant in a new investigation, retrieving and confirming the complete custody history requires separately querying both chains without a unified interface. Secondly, the paper fails to discuss comprehensively how different types of evidence with varied sizes and formats are treated in the migration process. Thirdly, their access control model is static; the permission structure for the archived cases remains identical in the cold chain to what they were while active. Obviously, this does not fit the use cases of long-term storage, where access requirements might evolve over time.

2.4 Jumpserver

A jump server, also known as a jump or bastion host, is an intermediate component that acts as secure and controlled gateway to any network or device behind it. To the best of our knowledge after extensive research, there is no implementation that utilizes a jump server as the gateway to a blockchain system. However, one interesting alternative is that of Al Breiki et al. [19] which proposed a decentralized access control system for IoT data using blockchain and trusted oracles. These oracles, which are a third-party service that acts as the bridge between the blockchain and the outside world, act as the middle-man and enforce user verification and access control policies before allowing any data queries. Despite the innovative use of such technology as gateways, this system exhibits several critical limitations when it comes to applying it to a digital CoC system: Firstly, the oracle gateway relies on a single authentication method which is blockchain identity verification. Recent studies show that using MFA (multi-factor authentication) reduces risk of compromise by 99.22% [20]. Furthermore, while the blockchain transactions themselves are cryptographically secured, the communication between users and the oracle gateway lacks end-to-end encryption. This means that even though the oracle verifies permissions, the actual data transmission between the gateway and the user could be intercepted or manipulated by MITM (man-in-the-middle) attacks. These types of attacks are heavily mitigated by mTLS, which is designed to prevent them. Moreover, the system relies on a set of trusted oracle nodes to perform gateway functions. If these oracle nodes are compromised or collude, they could bypass access control policies and perform harmful actions. A digital CoC system requires that even with compromised or unlawful participants in the blockchain system, data integrity and validity remain intact.

2.5 TEE Private Chaincode

In previous sections, all blockchain implementations shared a fundamental security assumption: all blockchain nodes (peers, orderers, and validators) must be fully trusted. If a single peer, which represents a role and is responsible for its transactions, is compromised, the attacker may potentially gain sensitive metadata, CA keys, or chaincode transactions. As a solution, researchers have started implementing Trusted Execution Environments (TEEs), which are hardware isolated secure enclaves responsible for the protection of code and data. This section will explore different implementations of TEEs in blockchain COC systems.

Brandenburger et al. [21] presented Fabric Private Chaincode (FPC), which is a TEE implementation, in their blockchain design. The proposed system uses FPC to execute chaincodes inside Intel SGX chaincode enclaves, which are isolated CPU memory regions. This ensures that compromised peers (if Investigator peer was comprised for example) cannot access evidence data. The way it functions is that the evidence is stored and encrypted in a ledger, where only the enclave holds the decryption keys. Each chaincode is isolated, meaning that they run separate enclaves. As for the ledger enclave, this component is used to maintain the integrity of the blockchain state. It is placed outside the SGX, alongside the enclave registry, which is a component responsible for managing all active enclaves by performing a verification process, also known as remote attestation, with each chaincode enclave. All chaincode enclave transactions and attestation results are stored and verified by the Enclave Transaction Validator. In short, all these steps ensure that peers are trusted, transactions are secure, and evidence is encrypted.

This idea has paved the way for interesting implementations; however, there are still some limitations. The article doesn't mention where actual artifacts are stored, despite handling the chaincode effectively. Additionally, their FPC framework lacks DFIR specific aspects. While it protects the generic smart contract, the FPC framework doesn't define how custody paths are verified and how role-based policies are enforced inside the enclaves.

Müller et al. [22] presented an alternative to the previous paper by introducing a system that handles data from IoT devices, such as Raspberry Pis, mobiles phones, and sensors. In this paper, they propose the system "TZ4Fabric", which uses ARM TrustZone, which is TEE available on mobile phones and ARM processors. In this design, IoT devices run the chaincode enclaves locally, and all collected evidence are cryptographically signed. Additionally, the origin of the evidence is validated by remote attestation, checking if it's a real hardware or compromised device. This system is split into two different parts: the untrusted normal, which runs the regular Hyperledger Fabric components while a small wrapper forwards all interactions, and the trusted secure world, which has the sensitive parts like the chaincode execution and is protected by the ARM TrustZone. Thus, only the smart contract logic runs inside this secure enclave.

Despite this unique idea, this system still faces some limitations. The lack of chaincode replication is evident in this paper: instead of deploying the same chaincode for multiple peers, this system assumed one single ARM device for chaincode executions, making it a single point if failure. This project also lacks an off-chain storage system, despite it's efforts to handle blockchain logic efficiently.

2.6 Off-Chain Storage and IPFS

A recurring limitation across blockchain-based COC is the lack of efficient storage mechanisms. Digital evidence can range from kilobytes of data, such as small file artifacts, to massive terabytes of data, such as disk images. Although the mentioned blockchain technology can handle small file evidence securely, it is not designed to store large binary files on-chain. To address this common issue, researchers have reverted to off-chain solutions such as the IPFS [23], a decentralized peer-to-peer distributed file system that stores data using content-based identifiers, meaning that each file is referenced by its cryptographic hash instead of traditional locations.

Nyaletey et al. [24] proposed BlockIPFS, a blockchain based forensic framework with an integrated IPFS component. The article introduces IPFS technology to blockchains to build a decentralized storage for large files with a complete audit trail of all file-related activities. Their IPFS component uses content-addressing with cryptographic hashes to store files, which ensures that every receives a unique identifier based on its content. Additionally, every single interaction with these files, whether uploading or accessing, is recorded as a blockchain transaction with timestamps and user identities. Smart contracts are utilized for the blockchain logic, as they are responsible for the automation of the verification process of user permissions before allowing any access, which creates an RBAC system. Although the addition of an IPFS component improved traditional blockchain implementations, this system still suffers from a critical issue that the authors establish themselves: the system is incapable of detecting modified versions of existing evidence files, which can comprise evidence authenticity in forensic investigations where attackers can upload slightly altered copies to conceal or plagiarize original data.

2.7 Summary & Gap Analysis

The literature review in sections 2.2 through 2.6 reveals critical limitations across existing blockchain based COC systems. External single points of failure, lack of off-chain storage systems, and inefficient hardware and software components are all a few examples of the tackled problems. Table 2.1 provides a comprehensive overview of the reviewed systems, their limitations, and how our proposed solution addresses these gaps.

Table 2.1: Summary of Blockchain-Based Chain-of-Custody Systems

Article & Authors	**Main Points**	**Limitations**	**Our Solution**
Bonomi et al. (2019)	Permissioned Ethereum for COC; Smart contracts validate transfers; EVM implementation	Off-chain identity (single point of failure); Public chain overhead	On-chain identity via MSP/CA; CA keys in SGX; Private network
Lone & Mir (2017)	Ethereum smart contracts; Logs user, state, permissions, timestamp	Off-chain identity; No large storage; Theory only	IPFS for large files; Jumpserver with MFA; Full implementation
Lone & Mir (2019)	Identity Registry in blockchain; RBAC via Composer; Decentralized authority	Deprecated Composer; No off-chain storage	Native Fabric SDK; Private IPFS; Hot/cold chains
Nischitha et al. (2023)	Native Fabric SDK; CA identity; MSP validates via PKI	Metadata only; Single channel	Hot/cold channels; IPFS for artifacts; Multi-org
Kim et al. (2021)	Hot/cold chains; Active vs archived; Docker implementation	No cross-chain audit; No format handling; Static access	Unified interface; Format-aware migration; Dynamic permissions
Al Breiki et al. (2020)	Oracles as gateways; Smart contract policies; Decentralized control	Single auth method; No encryption; Oracle trust	Jumpserver with MFA/mTLS; Multilayer auth; TEE trust
Brandenburger et al. (2018)	FPC with SGX; Encrypted ledger; Remote attestation	No artifact storage; No DFIR workflows	Split contract; IPFS integration; Custody verification
Müller et al. (2020)	ARM TrustZone; Local enclaves; Crypto signing	No replication (single point); No off-chain storage	Replicated SGX; Private IPFS; Multi-peer
Nyaletey et al. (2019)	BlockIPFS; Content-addressed; RBAC contracts	Cannot detect modified files; No similarity check	Hash on upload; Similarity in chaincode; SGX validation

Our project addresses these gaps through an integrated system that combines various proposed solutions from different articles with new components.

Unlike early Ethereum based systems which rely heavily on external providers, our system implements on-chain identity management through Hyperledger's native MSP and CA infrastructure, where each organization can operate its own CA and issue certificates to peers and users. On top of that, we anchored the CA root keys in the SGX enclaves which can prevent "external key" vulnerabilities.

This system uses an external Jumpserver gateway which participates in user authentication via password-less MFA and mTLS session establishments. This can eliminate external identity providers and provide secure RBAC for each authenticated session. After this process is done, the users are allowed to utilize the blockchain and IPFS.

Off-chain storage systems was a critical problem in most implementations, where it was either ignored or integrated via IPFS with no access control. For this reason, our project uses private IPFS cluster accessible only through the Jumpserver over mTLS. Once an upload is complete, the uploaded file is hashed and the uploaded is verified, then the corresponding metadata is anchored on the hot chain, which is responsible for all active investigation operations. The hot chain is separated from the cold chain, which is responsible for archived cases, since single channel designs degrade as more investigations are added. This dual chain architecture addresses scalability concerns. Figure 2.2 below shows the architecture of the hot-cold blockchain component that we will tackle in the methodology section.

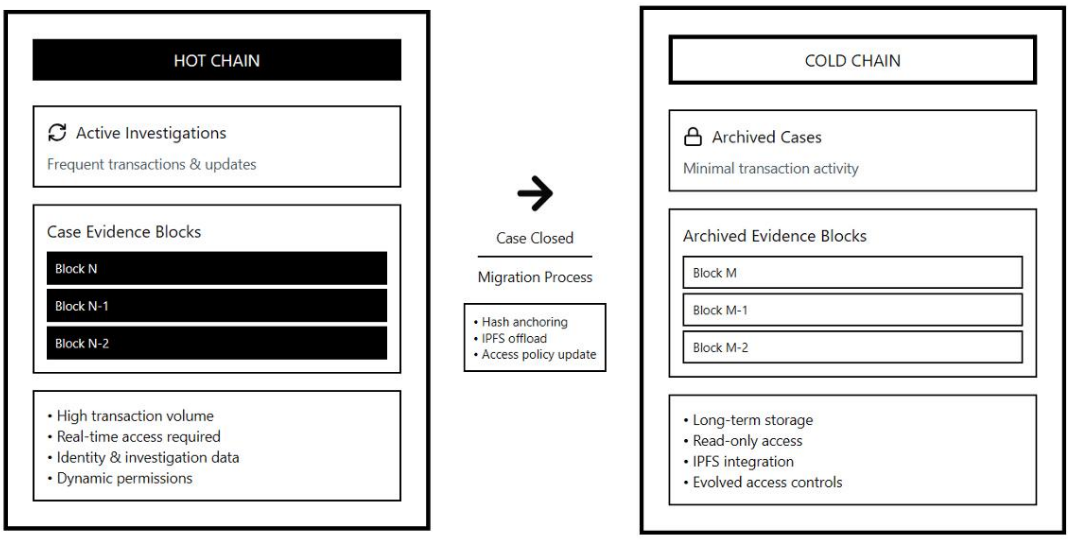


Figure 2.2: Hot/Cold chain architecture comparison

Although Brandenburger et al. [21] introduced FPC for protecting chaincode execution in SGX enclaves, their system lacks DFIR specific workflows. Our system extends this foundation with a split contract architecture: a public chaincode to record custody events and enforce MSP/role rules, and s private chaincode running in SGX enclaves to perform cryptographic checks, path verifications, and similarity analysis.

Chapter 3

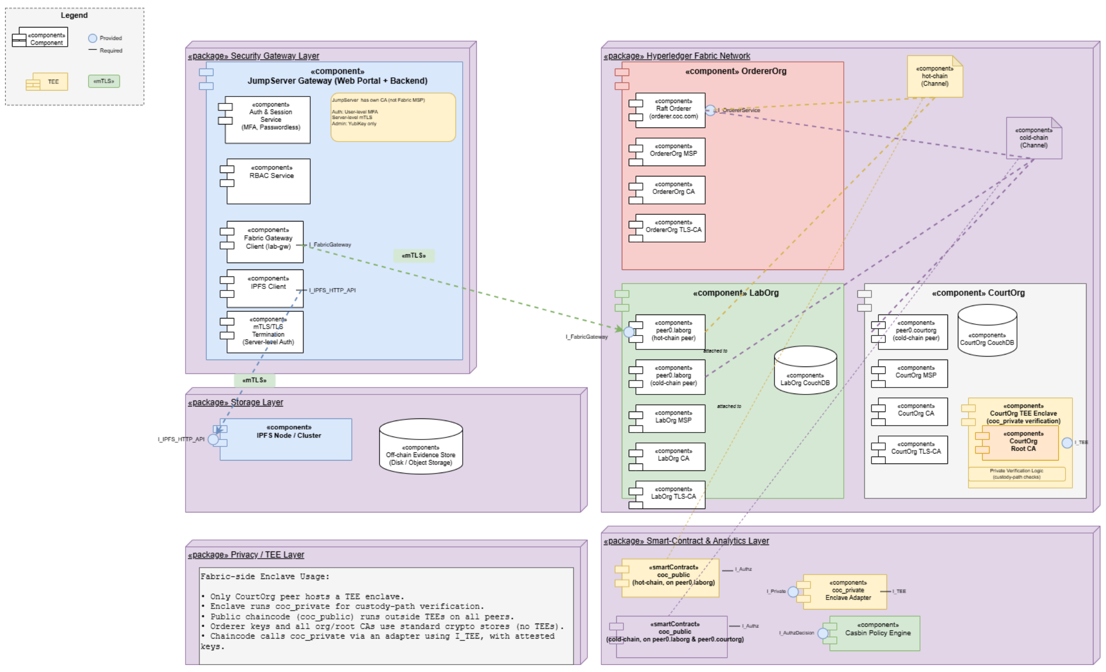
Proposed Solution Methodology

3.1 Methodological Overview and System Introduction

This chapter elucidates the rationale behind the methodological approaches employed in this work. It links back to the literature review and identification of existing gaps done in Chapter 2, and this is in order to substantiate the choices of the system design methods employed in this work which help us to fill in the literature gaps. In addition, the chapter builds on a methodological review showing the research methods employed by relevant studies in the literature and how and why this project embraced such methods.

Turning now to our proposed system, we introduce the overall methodology for designing the permissioned chain-of-custody system which combines the jumpserver as a controlled access gateway, a dual blockchain backend, and off-chain evidence storage. At a high level the system is constructed upon these three main components: Jumpserver which acts as a bastion host gateway through which investigators, auditors, and court users must pass, a two-layer Hyperledger Fabric network with a "hot" blockchain for active case work and a "cold" blockchain for long-term, archived court-facing records, and an IPFS layer that stores the actual evidence files while their hashes and forensic metadata are anchored on-chain. These components are integrated through a permissioned Hyperledger Fabric network, where organizations, MSPs, peers, and a RAFT ordering service define the trust and identity domain. The JumpServer connects to Fabric using the Fabric Gateway SDK over mutual TLS, and it connects to IPFS through the IPFS HTTP API to upload evidence files and read back their content hashes. Fabric itself never talks directly to IPFS; the JumpServer is the only component that bridges between the blockchain and the evidence storage layer. The remainder of this chapter outlines the overall system architecture and evidence flow, then details how each component is designed and implemented to address the gaps identified in Chapter 2.

3.2 System Architecture and Evidence Flow


Legend
«component» Component
Provided
Required
TEE
«mTLS»
«package» Security Gateway Layer
«component» JumpServer Gateway (Web Portal + Backend)
«component» Auth & Session Service (MFA, Passwordless)
JumpServer has own CA (not Fabric MSP)
Auth: User-level MFA
Server-level mTLS
Admin: YubiKey only
«component» RBAC Service
«component» Fabric Gateway Client (lab-gw)
I_FabricGateway
«mTLS»
«component» IPFS Client
I_IPFS_HTTP_API
«component» mTLS/TLS Termination (Server-level Auth)
«mTLS»
«package» Storage Layer
I_IPFS_HTTP_API
«component» IPFS Node / Cluster
«component» Off-chain Evidence Store (Disk / Object Storage)
«package» Privacy / TEE Layer
«package» Hyperledger Fabric Network
«component» OrdererOrg
«component» Raft Orderer (orderer.coc.com)
I_OrdererService
«component» OrdererOrg MSP
«component» OrdererOrg CA
«component» OrdererOrg TLS-CA
«component» hot-chain (Channel)
«component» cold-chain (Channel)
«component» LabOrg
I_FabricGateway
«component» peer0.laborg (hot-chain peer)
attached to
«component» peer0.laborg (cold-chain peer)
attached to
«component» LabOrg CouchDB
«component» LabOrg MSP
«component» LabOrg CA
«component» LabOrg TLS-CA
«component» CourtOrg
«component» peer0.courtorg (cold-chain peer)
«component» CourtOrg CouchDB
«component» CourtOrg MSP
«component» CourtOrg CA
«component» CourtOrg TLS-CA
«component» CourtOrg TEE Enclave (coc_private verification)
«component» CourtOrg Root CA
I_TEE
Private Verification Logic (custody-path checks)
«package» Smart-Contract & Analytics Layer
«smartContract» coc_public (hot-chain, on peer0.laborg)
I_Authz
I_Private
«component» coc_private Enclave Adapter
I_TEE
«smartContract» coc_public (cold-chain, on peer0.laborg & peer0.courtorg)
I_Authz
I_AuthzDecision
«component» Casbin Policy Engine

```
Fabric-side Enclave Usage:

• Only CourtOrg peer hosts a TEE enclave.
• Enclave runs coc_private for custody-path verification.
• Public chaincode (coc_public) runs outside TEEs on all peers.
• Orderer keys and all org/root CAs use standard crypto stores (no TEEs).
• Chaincode calls coc_private via an adapter using I_TEE, with attested
keys.
```

Figure 3.1: System Component Diagram

3.2.1 Overall System Architecture

As shown in Figure 3.1, the component diagram of the proposed permissioned chain-of-custody system is organised into three main layers: a Security Gateway Layer exposing all user-facing services, a Blockchain Layer recording chain-of-custody events, and a Storage Layer holding large evidence objects.

In terms of access control, users authenticate at the gateway through the Web UI using passwordless, MFA-based login, so each session is bound to a verified identity and then mapped to a role by the RBAC service. Court and auditor users follow the same user-level MFA process, whereas privileged administrative access to the JumpServer is restricted to the admin role which authenticates via a hardware security key known as the YubiKey and is operations-only, managing JumpServer infrastructure and access control (updates, certificates, user/role setup) with no direct forensic or evidence-handling privileges. For each backend call from the JumpServer's Fabric Gateway client and IPFS client, a mutual-TLS handshake is performed so that both endpoints authenticate using their CA-issued certificates and all subsequent traffic is carried over the resulting encrypted channel.

With regard to the Blockchain Layer, the Hyperledger Fabric network groups participants into LabOrg and CourtOrg, each with its own MSP and CA hierarchy; there is also the OrdererOrg, which operates the RAFT-based ordering service. A RAFT ordering service is responsible for block creation, while peers from different organisations join channels according to their duties: LabOrg peers participate in both the hot-chain and cold-chain, whereas CourtOrg peers join only the cold-chain. This separation ensures that day-to-day evidence uploads and updates on the hot-chain require only LabOrg endorsements, avoiding unnecessary latency or dependence on court availability, while archival and court-facing operations on the cold-chain explicitly require CourtOrg endorsements to provide legal oversight of long-term records. There is no separate auditor organisation or MSP, since auditors only need read-only access through the JumpServer; they query the ledger without endorsing transactions or hosting a full peer, and they rely on the existing CourtOrg trust anchor. Finally, in relation to storage, the system employs an IPFS cluster for off-chain evidence store: only CIDs, hashes, and structured metadata are written to the ledger, while the JumpServer performs authenticated upload and retrieval of full evidence files via the IPFS HTTP(S) API.

The TEE and Smart-Contract layers defined at the bottom of Figure 3.1 are integrated into the Blockchain Layer: the Smart-Contract layer implements both the public and private parts of the chaincode that record evidence actions and enforce authorization and endorsement rules, while the TEE layer provides a secure enclave for running confidential logic and protecting cryptographic keys.

3.2.2 Evidence Lifecycle and Data Flow

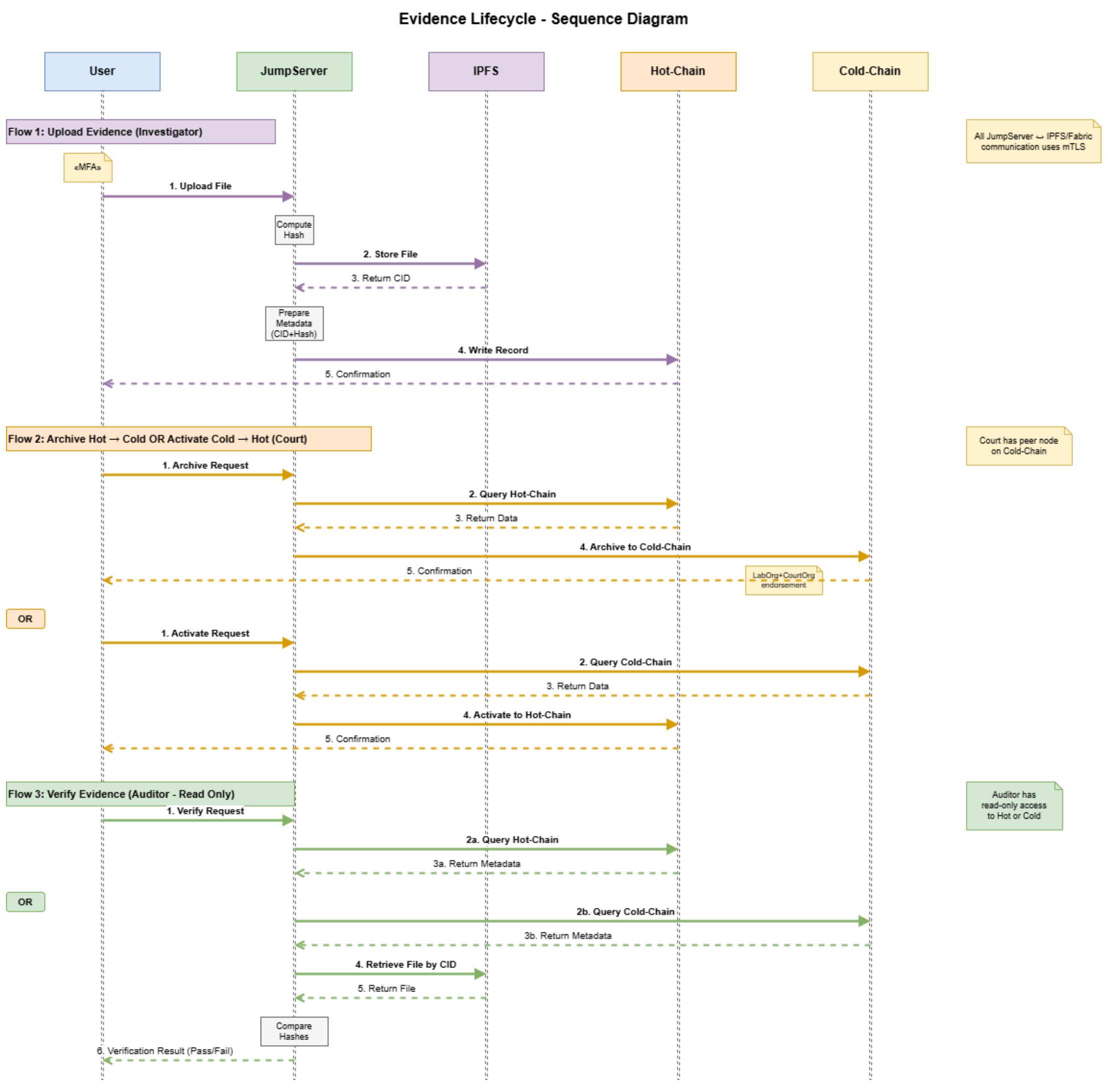


Figure 3.2: Evidence End-to-End Flow Sequence Diagram

Figure 3.2 illustrates the end-to-end lifecycle of digital evidence across the JumpServer, IPFS, Hot-Chain, and Cold-Chain components. The process unfolds in four stages: secure access, registration on the hot-chain, case-specific court-controlled archival on the cold-chain, and verification. What follows is a concise explanation of each stage as reflected in the figure.

1. Secure Access

As shown on the left side of Figure 3.2, an investigator, auditor, or court user connects to the JumpServer web portal and authenticates using passwordless, MFA-based login. The JumpServer binds the session to the user's assigned role, ensuring that all subsequent operations are authorised against the RBAC rules. Administrative functions are restricted to the designated operator who authenticates using YubiKey hardware device to manage infrastructure and access control without interacting with evidence. Once a session is active, every backend call from the JumpServer to Fabric or IPFS proceeds over mTLS, guaranteeing mutual certificate validation and encrypted machine-to-machine communication, as depicted across the participants in the sequence diagram.

2. Registration on the Hot-Chain

Flow 1 in Figure 3.2 corresponds to evidence registration. An authorised investigator uploads a file to the JumpServer, which computes its cryptographic hash and transfers the file to IPFS over an mTLS-secured API call. IPFS returns the CID, after which the JumpServer prepares a structured chain-of-custody record and submits a transaction to the hot-chain. The hot-chain smart contract records the item, emits a custody event, and supports rapid updates while the case is active.

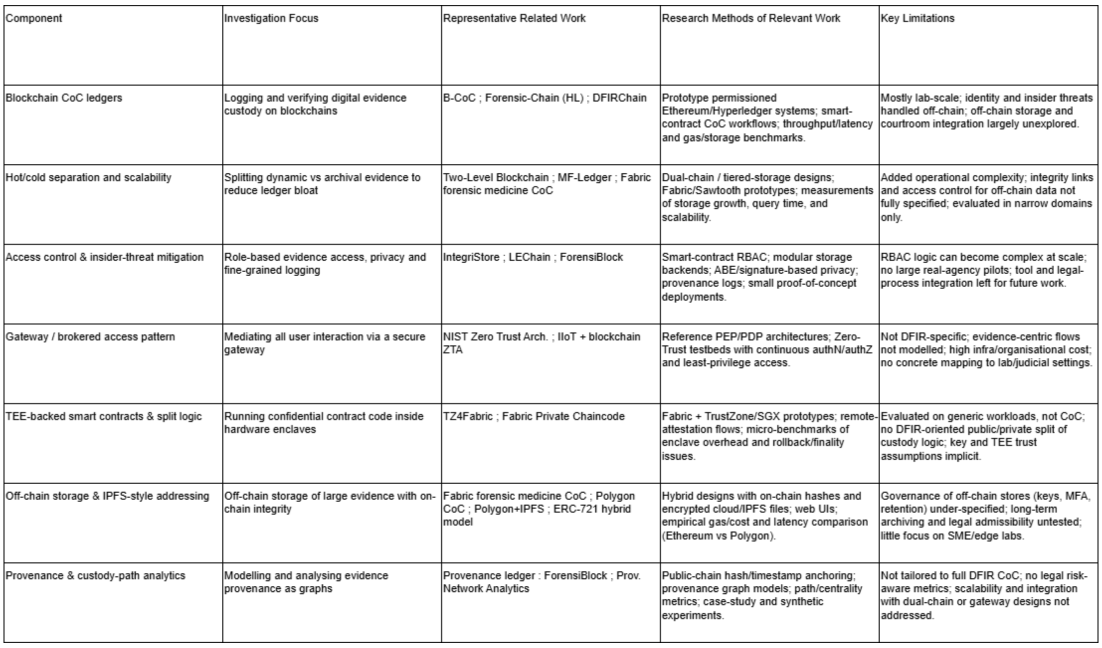
3. Court-Controlled Archival to the Cold-Chain

Flow 2 in Figure 3.2 depicts archival operations initiated exclusively by a court role. A court user issues an archive request, the JumpServer retrieves the relevant hot-chain entries, and archival transactions are submitted to the cold-chain. Because the cold-chain's endorsement policy requires signatures from both LabOrg and CourtOrg peers, archival actions complete only when court-level oversight is explicitly provided. This ensures that long-term forensic records cannot be created without judicial approval.

4. Verification and Audit

Flow 3 of Figure 3.2 corresponds to verification. A court user or auditor requests validation of a specific evidence item. The JumpServer queries the cold-chain, retrieves the associated CID, fetches the file from IPFS, recomputes the hash, and compares it with the value stored on-chain. When a custody path report is court mandated or required for anomalous reasons (role mismatch, irregular order, or any form of tampering), TEE-based private contract execution reconstructs the full custody trail via an mTLS-secured internal HTTPS API call from the JumpServer. The user's identity remains tied to an MFA-protected session throughout, with all backend actions carried over mTLS, providing an end-to-end, auditable, and cryptographically verifiable chain of custody.

3.3 Component-Level Methodology



Component	Investigation Focus	Representative Related Work	Research Methods of Relevant Work	Key Limitations
Blockchain CoC ledgers	Logging and verifying digital evidence custody on blockchains	B-CoC ; Forensic-Chain (HL) ; DFIRChain	Prototype permissioned Ethereum/Hyperledger systems; smart-contract CoC workflows; throughput/latency and gas/storage benchmarks.	Mostly lab-scale; identity and insider threats handled off-chain; off-chain storage and courtroom integration largely unexplored.
Hot/cold separation and scalability	Splitting dynamic vs archival evidence to reduce ledger bloat	Two-Level Blockchain ; MF-Ledger ; Fabric forensic medicine CoC	Dual-chain / tiered-storage designs; Fabric/Sawtooth prototypes; measurements of storage growth, query time, and scalability.	Added operational complexity; integrity links and access control for off-chain data not fully specified; evaluated in narrow domains only.
Access control & insider-threat mitigation	Role-based evidence access, privacy and fine-grained logging	IntegriStore ; LEChain ; ForensiBlock	Smart-contract RBAC; modular storage backends; ABE/signature-based privacy; provenance logs; small proof-of-concept deployments.	RBAC logic can become complex at scale; no large real-agency pilots; tool and legal-process integration left for future work.
Gateway / brokered access pattern	Mediating all user interaction via a secure gateway	NIST Zero Trust Arch. ; IIoT + blockchain ZTA	Reference PEP/PDP architectures; Zero-Trust testbeds with continuous authN/authZ and least-privilege access.	Not DFIR-specific; evidence-centric flows not modelled; high infra/organisational cost; no concrete mapping to lab/judicial settings.
TEE-backed smart contracts & split logic	Running confidential contract code inside hardware enclaves	TZ4Fabric ; Fabric Private Chaincode	Fabric + TrustZone/SGX prototypes; remote-attestation flows; micro-benchmarks of enclave overhead and rollback/finality issues.	Evaluated on generic workloads, not CoC; no DFIR-oriented public/private split of custody logic; key and TEE trust assumptions implicit.
Off-chain storage & IPFS-style addressing	Off-chain storage of large evidence with on-chain integrity	Fabric forensic medicine CoC ; Polygon CoC ; Polygon+IPFS ; ERC-721 hybrid model	Hybrid designs with on-chain hashes and encrypted cloud/IPFS files; web UIs; empirical gas/cost and latency comparison (Ethereum vs Polygon).	Governance of off-chain stores (keys, MFA, retention) under-specified; long-term archiving and legal admissibility untested; little focus on SME/edge labs.
Provenance & custody-path analytics	Modelling and analysing evidence provenance as graphs	Provenance ledger : ForensiBlock ; Prov. Network Analytics	Public-chain hash/timestamp anchoring; provenance graph models; path/centrality metrics; case-study and synthetic experiments.	Not tailored to full DFIR CoC; no legal risk-aware metrics; scalability and integration with dual-chain or gateway designs not addressed.

Table 3.1: Methodological Patterns and Limitations of Previous Work

Table 3.1 presents a summary of the methodological patterns and identified gaps of the reviewed corpora of work. The conclusions derived from the literature review fed the design and implementation of our permissioned dual-system architecture for documenting chain of custody using JumpServer. Our system strikes a balance between the CIA security triad and guarantees of authenticity and non-repudiation, while remaining a usable daily DFIR workflow.

It is ad rem to mention that, existing blockchain-based CoC prototypes already demonstrate technically promising metrics, such as acceptable end-to-end latency, sustainable throughput on permissioned ledgers, and orders-of-magnitude reductions in transaction costs when moving from public Ethereum to more efficient platforms, yet they often achieve these results without a rigorous integration of Zero-Trust access control, enclave-backed verification, or court-oriented admissibility criteria. This gap between "engineering metrics" (transactions per second, gas cost, storage growth) and "forensic metrics" (verifiable custody paths, role separation, risk of insider abuse) is precisely where our methodology positions itself.

The remainder of this section examines how each system component operationalizes our methodological choices.

3.3.1 Jump Server

The JumpServer functions as the system's sole controlled access gateway, enforcing multi-factor authentication, role-based access control, and a strict Zero-Trust mediation model. All investigator, court, and auditor operations traverse the JumpServer, which authenticates users, binds each session to a role, and performs all backend communication with Hyperledger Fabric and IPFS exclusively over mutual TLS. The JumpServer computes hashes, submits transactions, retrieves evidence, and triggers enclave-based verification, ensuring that no external user directly interacts with peers, orderers, or storage nodes. By consolidating access control, certificate validation, and audit logging into a hardened intermediary, the JumpServer provides a deterministic, tamper-evident, and fully traceable pathway for all chain-of-custody operations.

3.3.2 Hyperledger Fabric Network, Organisations, and Certificate Hierarchy

In blockchains, the distributed digital record-keeping system which is at the core of maintaining the chain of custody is the ledger. The core ledger in our project is implemented as a permissioned Hyperledger Fabric network based on the current LTS v3.0.0 line, consistent with Fabric's design as a permissioned, organization-scoped blockchain framework [25]. Hyperledger Fabric is an umbrella project at the Linux Foundation suitable for maintaining chain of custody, since unlike other projects like Ethereum it follows an execute–order–validate architecture [25], which decouples transaction execution from the consensus mechanism [21].

Put simply, Hyperledger Fabric guarantees that recorded transactions are finalized instantly and permanently through deterministic finality [25], providing a secure and private network where data is visible only to authorized organizations [25]. This ensures efficient, high-speed, and auditable record-keeping suitable for enterprise supply chains [21]. It is defined by deterministic finality where once a block is committed to the ledger, it is final and immutable [25].

3.3.2.1 Organisations, Peers, and Ordering service

The network comprises three organizations (2 application organizations and 1 ordering organization):

- LabOrg: digital forensics laboratory (investigators and analysts), represented by ForensicLabMSP

- CourtOrg: judicial body controlling archival and court-facing verification, represented by CourtMSP
- OrdererOrg: operates the ordering service.

In our deployment, these are instantiated twice: once for the hot blockchain under the `*.hot.coc.com` domain and once for the cold blockchain under `*.cold.coc.com`.

A detailed view reveals that the hot network includes `orderer.hot.coc.com:7050`, `peer0.court.hot.coc.com:7051`, `peer0.forensiclab.hot.coc.com:8051` on channel `hotchannel`. The cold network includes `orderer.cold.coc.com:8050`, `peer0.court.cold.coc.` `peer0.forensiclab.cold.coc.com:10051` on channel `coldchannel`.

Each organization runs one or more peers which act as endorsers and committers. The endorsers execute the chaincode on a proposed transaction and produce an endorsement: a signed response containing the read-set and write-set. The committers validate ordered transactions and apply state updates to their local ledger.

In practice, the endorsement flow is:

1. The JumpServer gateway sends a transaction proposal to one or more endorsing peers (e.g., `peer0.forensiclab.hot.coc.com` and `peer0.court.hot.coc.com`).
2. Each endorser simulates the chaincode call, producing a deterministic read/write set and signs it with its peer certificate under CourtMSP or ForensicLabMSP.
3. The gateway collects endorsements and checks that:
   - all endorsements agree on the read/write set;
   - the set of endorsers satisfies the endorsement policy
4. Only then is the transaction submitted to the ordering service.

This "execute-then-order" model is one of the reasons Fabric is suitable for chain of custody: it allows precise control over which organisations must endorse each custody event, as demonstrated in [13, 17].

The ordering service is implemented as a RAFT cluster operated by OrdererOrg. Raft is a consensus algorithm which follows the "one leader, several followers" model:

- One orderer node (e.g., `orderer.hot.coc.com`) is elected leader.
- Clients send proposed transactions to the leader.

- The leader batches transactions into blocks and replicates them to follower orderers.
- Once a majority of orderers acknowledge a block, it is committed and broadcast to all peers.

Unlike proof-of-work systems, RAFT provides deterministic finality: once a block is committed, it is never reorganised. This property is crucial both for legal admissibility (courts expect append-only histories [9]) and for TEE-backed contracts that assume a stable ledger view [21, 22]. Orderer nodes are deployed on separate machines and mutually authenticate using mTLS.

3.3.2.2 MSPs, CAs, certificate lifecycle and enclave-anchored private contracts

Hyperledger Fabric separates identity from application logic using Membership Service Providers (MSPs) and Certificate Authorities (CAs) [16, 21]. In our network, CourtOrg and LabOrg each run a Fabric CA that issues X.509 certificates to their peers and internal client applications, and each defines a corresponding MSP (CourtMSP and ForensicLabMSP). The JumpServer itself is not enrolled under these Fabric CAs; instead, it uses a separate gateway CA described at the Security Gateway layer, and authenticates to Fabric as a client under one of the existing MSPs.

During enrollment, peers receive short-lived X.509 (mTLS) certificates from their organisation's CA. Certificates are rotated regularly, and immediately if compromise is suspected. When rotation occurs, the channel MSP configuration is updated to trust the new CA certificates for a transition period before the old ones are removed, avoiding sudden loss of availability while still enforcing revocation hygiene. Revocation is handled through Certificate Revocation Lists (CRLs) issued and signed by the organisational CAs. All Fabric nodes (peers and orderers) check these CRLs during each mTLS handshake to ensure they do not trust certificates that have been revoked (i.e., explicitly marked as no longer valid) [16, 21].

Trusted execution environments are used only at the chaincode layer, following the Fabric Private Chaincode (FPC) model [21,22]. A private contract is compiled as an SGX enclave on endorsing court peers. Inside the enclave reside: the private chaincode logic for deep custody-path verification and confidential policy checks, an enclave-sealed signing key dedicated to producing verification verdicts, and an attestation key bound to the hardware trust chain. The enclave's memory is encrypted and isolated from the host OS, and the signing key is sealed to the enclave, so even a compromised peer or hypervisor cannot extract it [21, 22].

Before trusting any enclave output, the JumpServer performs remote attestation. It requests an attestation quote, verifies it with the TEE vendor, and checks that the reported measurement (hash of the enclave binary) matches a pinned reference value. Only enclaves whose measurements match this expected hash are allowed to produce signed verdicts that the public chaincode will later accept. TEEs are narrowly scoped to protecting the private verification contract and its signing key. This focused use of enclaves strengthens authenticity and integrity for court-grade verification, while preserving Fabric's permissioned, identity-rich model built on MSPs and CAs [16, 21, 22].

Finally, when an external auditor needs direct on-chain verification (for example, a court-appointed expert), they are given a Fabric client certificate under an existing MSP (typically CourtMSP) with an auditor attribute encoded in the X.509 extensions. Smart contracts can read both the MSP ID and this attribute, enforce auditor-specific rules, and record auditor queries as on-chain events, so their verification actions become part of the same auditable history instead of living only in off-chain logs [17, 18].

3.3.2.3 Hot-chain and cold-chain channels, and evidence invalidation

To separate noisy, high-frequency investigative activity from long-term court records and to control ledger growth, we follow a two-level ledger pattern similar to two-level blockchain and hot/cold storage designs proposed for digital evidence [17, 18]. Logically, we define two application channels:

Hot chain (hotchannel)
The hotchannel runs on `orderer.hot.coc.com` with peers `peer0.court.hot.coc.com` and `peer0.forensiclab.hot.coc.com`. It is used to record custody events and metadata while a case is active. ForensicLab roles operate autonomously for lab-internal updates so only forensic lab peers have to endorse, however joint endorsement is required for inter-organizational transfers or status changes that have legal impact.

Cold chain (coldchannel)
The coldchannel runs on `orderer.cold.coc.com` with peers `peer0.court.cold.coc.com` and `peer0.forensiclab.cold.coc.com`. It stores final custody paths and TEE-signed verification records for closed/archived cases. Here, endorsement is always done by the court and investigator MSPs so neither party can modify archival records on its own.

In practice, we run two Fabric networks which use different domains and ports. All traffic across both networks uses mTLS and the CA/MSP stack described in Section 3.3.2.2.

The same channel design supports explicit evidence invalidation without breaking immutability, as recommended in recent blockchain-based CoC process models that add “invalidate evidence” operations instead of destructive deletion [18]. If an investigator uploads the wrong or corrupted file, they cannot erase the original record. Instead, they call `InvalidateEvidence(evidenceID, reason, wrongTxID)`, which links to the original transaction and IPFS CID, sets the evidence status to INVALIDATED on hotchannel (and on coldchannel if already archived), and records the caller's identity, role, and timestamp. Read-side functions such as `GetEvidenceSummary` then treat INVALIDATED items as logically void while still exposing the full history of the mistake and its correction. Courts can see both the erroneous upload and its formal invalidation, and automated tools can safely ignore invalidated entries, preserving non-repudiation and auditability while providing a controlled way to handle human error [13, 14, 17, 18].

3.3.2.4 World State and CouchDB

In Hyperledger Fabric, the ledger has two layers: an immutable blockchain of blocks, and a world state that holds the latest value of each key. In our design, the world state is stored in CouchDB, a JSON document store. Each evidence item (keyed by `caseID:evidenceID`) is represented as a JSON document containing status, IPFS CID, hash, and minimal metadata.

Using CouchDB allows rich queries over evidence metadata (for example, “all ACTIVE items for case X” or “all evidence archived in a given date range”) without scanning the full blockchain. This is particularly important for high-volume DFIR workloads, where investigators need fast, filtered views of the current state of many artefacts, a limitation explicitly noted in earlier CoC systems that relied only on block-level scans [13–15].

3.3.3 Smart Contract Split Architecture

A smart contract is a self-executing digital agreement with the terms of the agreement written directly into lines of code and stored on a blockchain. It follows a model where if the conditions are met then a specific action is triggered. The later mentioned chaincode is the Hyperledger Fabric platform's specific implementation of smart contracts.

The forensics application logic is implemented as a split smart-contract architecture on Hyperledger Fabric: a small public chaincode that exposes custody operations and a TEE-backed private chaincode that runs sensitive checks and handles high-value keys. This directly addresses a gap in earlier CoC ledgers, which either keep all logic public on chain [13, 14] or use TEEs on generic workloads without DFIR-specific custody semantics [21, 22]. These smart-contract components build on the network and certificate hierarchy described in Section 3.3.2.

3.3.3.1 Public Chaincode

On both hotchannel and coldchannel, the public chaincode maintains the official custody log with a narrow scope. For each evidence item it stores a single key–value record keyed by (caseID, evidenceID), containing only three groups of fields: the status of the item (ACTIVE, ARCHIVED, REACTIVATED), the IPFS CID and cryptographic hash of the underlying file, and minimal metadata such as case identifier, acquisition and upload timestamps, originating lab, and current holder role. For every operation (create, transfer, access, archive, reactivate, invalidate), it also appends a structured custody event to that item's event list.

Operationally, functions like `CreateEvidence`, `TransferCustody`, `ArchiveToCold`, `ReactivateFromCold`, `InvalidateEvidence` and `GetEvidenceSummary` follow the same pattern. The JumpServer gateway submits a transaction proposal signed with its Fabric client certificate; the public chaincode verifies that the invoker belongs to a trusted MSP (CourtMSP or ForensicLabMSP) and corresponds to the gateway service principal rather than a raw human identity [10, 17]. The actual human user and their role (Investigator, LabAnalyst, CourtUser, Auditor) are passed in transient data (JSON key–value pairs sent only to endorsing peers). The chaincode then applies simple role checks: only Investigators may create evidence on hotchannel, only Court users may finalize archival on coldchannel, and only the current custodian organisation may transfer custody [13, 17]. If the checks succeed, the public chaincode updates the world state and appends a new custody event to its event list.

Earlier chain-of-custody contracts such as [13, 14] tended to mix detailed business logic, identity handling and storage responsibilities inside a single monolithic smart contract. In contrast, the public chaincode is intentionally coarse grained. Its role is to record an immutable and ordered sequence of custody events, to enforce organisation-level endorsement and role constraints that align with MSP policies, and to emit clean event streams for TEE-based verification. Keeping the public contract small and focused in this way makes it easier to audit and reduces the risk of over-engineered, tightly coupled RBAC logic inside chaincode, an issue that was highlighted in systems like [17, 18].

3.3.3.2 Private Chaincode in TEEs

Sensitive logic and key material are moved into trusted enclaves. A private contract is compiled as an enclave application on endorsing court peers that support Intel SGX, following the Fabric Private Chaincode model [21, 22]. The enclave hosts on the inside:

- confidential verification predicates (deep consistency checks over custody histories, role and time constraints, anomaly flags)

- cryptographic operations (verifying digital signatures on evidence files, computing HMACs over metadata, signing verification reports with an enclave-sealed key)
- selected high-value keys (e.g. a key dedicated to signing custody-verification results).

The enclave's memory is encrypted and isolated from the host OS, and secrets are sealed to the enclave, so even a compromised peer or hypervisor cannot extract them [21, 22]. Peers outside the TEE only ever see encrypted inputs and signed outputs.

Access rules inside the private chaincode are enforced with a Casbin policy engine (control library for enforcing authorization). It only embeds the policy matrix (P lines) and not the dynamic user-to-role bindings (G lines), as the G lines need constant updating which would require the enclave to be constantly updated; updating the enclave requires changing its measurement hash and forcing regeneration of trust and re-attestaion. One example of a policy line would be of the form: `p, BlockchainInvestigator, blockchain.evidence, create, *`.

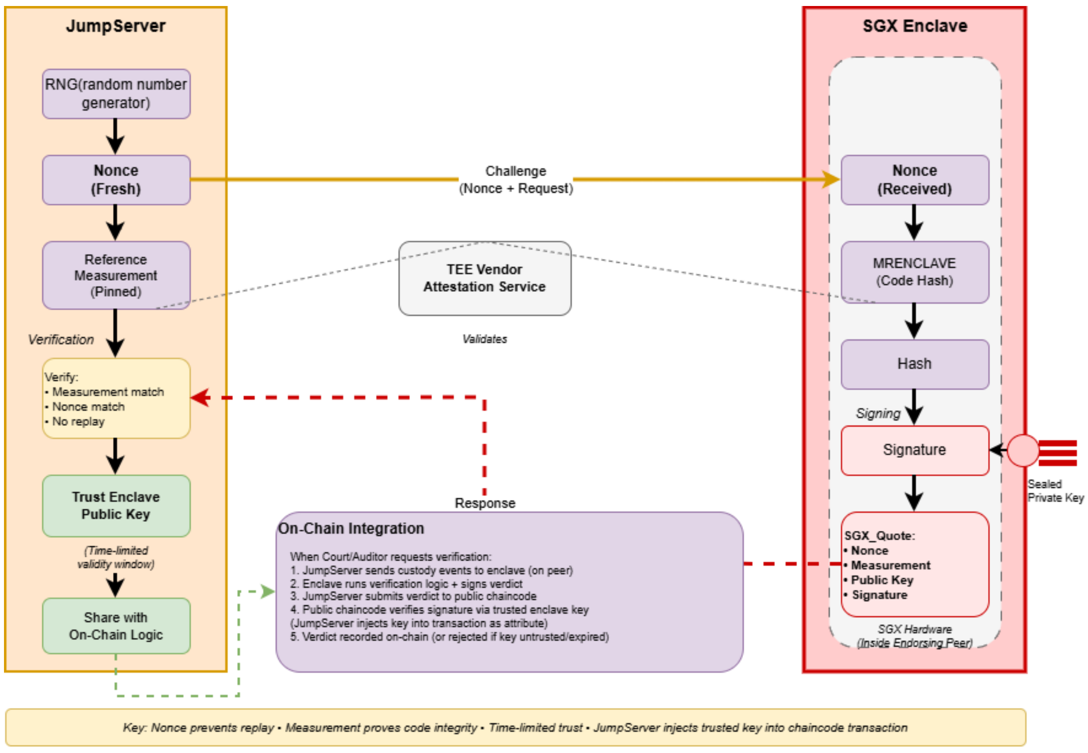
Looking at the above p line: `p` marks a policy entry, `BlockchainInvestigator` is the role, `blockchain.evidence` is the object (the resource domain), `create` is the action, and `*` is a wildcard for the resource scope (for example "any evidence item"). At runtime, the JumpServer maps each caller to a role and passes only the role/object/action to the enclave, which then checks them against this static p table. `g, user, role` lines are not included inside the enclave, to avoid the "RBAC baked into contracts" pattern criticised in systems like [17, 18]. This keeps enclave binaries stable and rarely changed, so that their measurement can serve as a long-lived, tamper-resistant trust anchor rather than something that must be rebuilt every time a user is added or removed.

Public chaincode and enclave-based private logic interact through a narrow interface that is always driven by the JumpServer. When a court user or auditor requests a deep custody check, the JumpServer first queries the ledger (hotchain for active cases or coldchain for archived ones), extracts the relevant custody events, and reduces them to a compact summary: evidence identifier, operation type, timestamps, organisation MSP, and caller role. This summary is sent to the enclave over an internal HTTPS connection protected by mTLS. Inside the enclave, the private verification code reconstructs the custody path, applies its confidential checks, and produces an aggregate verdict such as "custody history consistent" or "missing court endorsement at step k". The verdict is signed with an enclave-sealed key and returned to the JumpServer, which then submits a normal Fabric transaction on the appropriate channel. The public chaincode verifies the enclave signature and records the verdict as a verification record in the ledger. In this design, public chaincode remains the only component that updates on-chain state, while the enclave focuses on off-chain verification and key protection. This avoids giving TEEs direct ledger access, keeps the on-chain surface auditable, and fills the gap between fully public custody logic [13, 14] and TEE-based contracts that lack DFIR-specific semantics [21, 22].

Investigators interact exclusively with the public chaincode on the hotchain, which handles high-frequency custody events. In contrast, courts and auditors require full custody-path validation and legally admissible verification, so the JumpServer triggers the enclave logic only during court-mandated or audit-driven checks. This separation ensures performance for investigative workflows while reserving TEE resources for integrity-critical operations.

3.3.3.3 Remote Attestation and Enclave Trust

Earlier TEE-backed smart-contract prototypes often take the enclave's trustworthiness for granted, without clearly showing how that trust is established, renewed, or integrated into the application's own logic [21, 22]. In the suggested system, no enclave output is accepted until it passes a remote attestation process driven by the JumpServer. At scheduled intervals or upon demand, the JumpServer opens an mTLS connection to the enclave's attestation endpoint and sends an attestation challenge containing a fresh, unpredictable nonce together with its request (to avoid replay attacks). The enclave responds with a signed structure that includes this nonce, a measurement (the cryptographic hash of the private-chaincode binary and configuration), and a fresh public key bound to that enclave instance. The JumpServer verifies this quote using the TEE vendor's attestation service and checks that the reported measurement matches a pinned, operator-approved value and that the nonce is correctly reflected, ruling out replay. Only if these checks succeed is the enclave's public key recorded as "trusted" for a limited validity window and shared with the on-chain verification logic. When a court user or auditor later requests a custody-path verification, the JumpServer sends a reduced summary of ledger events to the enclave over mTLS; the enclave runs the confidential verification logic, signs the verdict with its sealed private key (for identity verification), and returns it. The JumpServer then submits this verdict as a normal Fabric transaction, and the public chaincode verifies that the signature matches one of the currently attested enclave keys before recording the verdict as a verification record on the ledger. Unattested, expired, or mismatched keys cause the transaction to be rejected. In this way, the enclave becomes a genuine cryptographic trust anchor: its code measurement, attestation certificate, and keys are explicitly tied both to the hardware trust chain and to the ledger's own validation rules, closing a methodological gap in earlier TEE–Fabric work where attestation remained largely external to the application's access-control and logging model [21, 22].



On-Chain Integration

When Court/Auditor requests verification:
1. JumpServer sends custody events to enclave (on peer)
2. Enclave runs verification logic + signs verdict
3. JumpServer submits verdict to public chaincode
4. Public chaincode verifies signature via trusted enclave key
(JumpServer injects key into transaction as attribute)
5. Verdict recorded on-chain (or rejected if key untrusted/expired)

Key: Nonce prevents replay • Measurement proves code integrity • Time-limited trust • JumpServer injects trusted key into chaincode transaction

Figure 3.3: Remote Attestation Protocol with On-Chain Integration

Figure 3.3 visually summarises this remote-attestation pipeline and shows how it is anchored in the overall verification flow. On the left, the JumpServer generates a fresh nonce, sends the challenge to the enclave, and checks the returned quote against the pinned measurement and the TEE vendor's attestation service. On the right, the enclave uses its hardware root of trust to bind the nonce, its code hash, and its public key into a signed quote. The lower part of the figure then links this to the on-chain phase: once the key is marked trusted for a limited window, custody-verification requests are sent to the enclave, and only verdicts signed with that attested key are forwarded to the public chaincode and recorded in the ledger. In this way the diagram reinforces how remote attestation, enclave keys, and on-chain verification are tied together into one coherent trust loop rather than remaining a separate, purely infrastructural step.

3.3.3.4 Off-chain Storage with IPFS

The InterPlanetary File System (IPFS) is a distributed, peer-to-peer content-addressable storage protocol. Instead of addressing data by location as done via URLs, IPFS retrieves files by a Content Identifier (CID), a cryptographic hash that uniquely represents the file's content. Any modification produces a new

CID, giving IPFS strong tamper-evidence semantics. Internally, IPFS organises data as a Merkle-DAG, where file blocks are stored in a local blockstore (not in the SQL db sense) and metadata in a lightweight key–value datastore, with versioning supported through links between DAG nodes [4, 23].

Following the hybrid blockchain–IPFS pattern used in earlier chain-of-custody prototypes [13, 17, 24], our design embeds IPFS directly into the DFIR workflow but adds strict identity controls, authenticated gateway access, and explicit linkage to ledger events. Large evidence artefacts (disk images, memory captures, logs) never enter the blockchain; instead, investigators upload them to the JumpServer, which computes the file hash, transmits the file to a private IPFS gateway over mTLS, and receives the resulting CID. The JumpServer then verifies that the CID's multihash matches the locally computed hash and submits a `CreateEvidence` transaction that anchors the CID and metadata on the hotchain under the appropriate endorsement policy. In the provided architecture, an HTTP gateway is in front of the cluster which has a server certificate signed by the JumpServer CA and verifies the JumpServer client certificate for mTLS.

Retrieval reverses the same path: the JumpServer queries the ledger for the CID, fetches the file from IPFS over mTLS, recomputes its hash, and raises an integrity alert if a mismatch is detected. Because all IPFS access flows through the JumpServer, every upload, download, or retrieval becomes a custody-logged action, tightly binding access to authenticated user sessions. This directly addresses gaps in earlier systems where IPFS access control and auditability were left to off-chain assumptions or informal trust models [1, 24, 26].

As a peer-to-peer protocol, IPFS depends on reachable peers that pin a given CID. In public IPFS environments, content may become temporarily unreachable when hosting peers go offline. The provided design mitigates this by deploying a private IPFS cluster governed jointly by the lab and court organizations. Evidence objects are pinned across multiple controlled nodes, and the cluster accepts requests only from the JumpServer's mTLS identity. This ensures availability while retaining IPFS's content-addressed integrity properties. Even in extreme downtime scenarios, the CID and metadata remain immutably recorded on the blockchain, and evidence becomes retrievable once at least one cluster node comes back online.

Figure 3.4 situates IPFS within the full DFIR evidence pipeline. All evidence flows originate at the JumpServer, which is the only component allowed to communicate with the IPFS cluster. Investigators upload files to the JumpServer, which hashes them, transmits them to IPFS over mTLS, and anchors the CID and metadata on the hotchain. Court users and auditors follow the same controlled path through the JumpServer during retrieval or verification, ensuring that no user ever interacts with IPFS directly.

The lower part of Figure 3.4 shows the nodes of the physical cluster while the upper part represents the same cluster logically as an organization-controlled IPFS cluster. The private cluster consists of four IPFS nodes:

- Lab organization node (primary laboratory storage)
- Court organization node (judicial domain replica)
- Redundant node (cross-domain failover for resilience)
- Replica node (ensures availability even if multiple nodes go offline)

Files are pinned across these nodes, mitigating the typical “IPFS unreachable” issue that affects public swarms. Figure 3.4 thus visualizes a closed, auditable evidence pipeline where files pass through a security gateway, are stored redundantly across a controlled IPFS cluster, and are referenced on Fabric hot/cold chains as part of an end-to-end verified chain of custody.

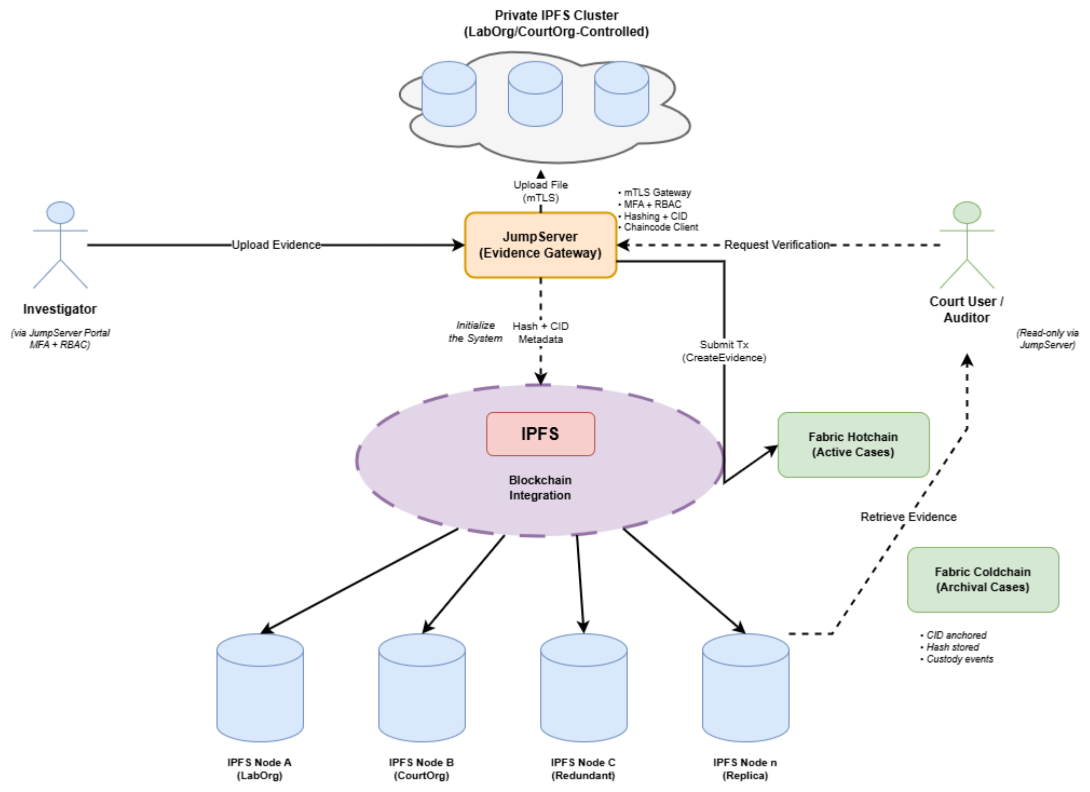


Figure 3.4: Private IPFS Cluster Integration within Evidence Pipeline

3.4 Requirement Specifications

This section defines the main functional and non-functional requirements of the proposed JumpServer-mediated, dual-chain Hyperledger Fabric system for digital forensics chain of custody. Requirements are derived both from forensic/legal criteria (integrity, traceability, admissibility) and from the technical limitations observed in existing blockchain-based CoC prototypes (limited access-control integration, weak treatment of off-chain storage, and scarce court-oriented workflows).

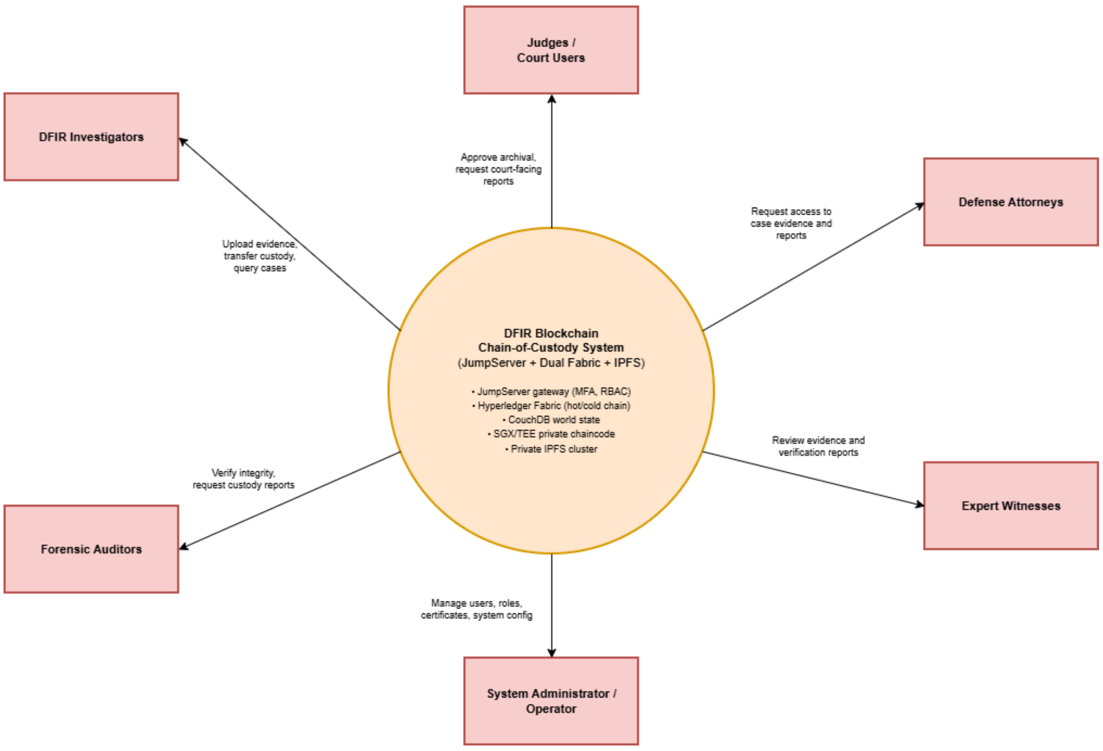

Judges / Court Users
DFIR Investigators
Defense Attorneys
Approve archival, request court-facing reports
Upload evidence, transfer custody, query cases
Request access to case evidence and reports
DFIR Blockchain Chain-of-Custody System (JumpServer + Dual Fabric + IPFS)
• JumpServer gateway (MFA, RBAC)
• Hyperledger Fabric (hot/cold chain)
• CouchDB world state
• SGX/TEE private chaincode
• Private IPFS cluster
Verify integrity, request custody reports
Review evidence and verification reports
Forensic Auditors
Expert Witnesses
Manage users, roles, certificates, system config
System Administrator / Operator

Figure 3.5: DFIR Blockchain System Context Diagram

Figure 3.5 represents the system context for the proposed work. The system is shown as a secure platform which stakeholders defined in section 1.1 all access through the JumpServer gateway, while the underlying Fabric networks and IPFS storage remain internal. This view clarifies the system boundary and the external actors whose needs drive the functional and non-functional requirements that follow.

3.4.1 Functional Requirements

Gateway-only access and RBAC
All investigator, auditor, and court interactions shall pass through the JumpServer gateway, which authenticates users with passwordless multi-factor login and binds each session to one of the roles: Investigator, Court, Auditor, Admin. No external user shall have direct network access to Fabric peers or IPFS nodes. The JumpServer shall enforce role-based access control (RBAC) for every operation and use mutual TLS (mTLS) with X.509 certificates for all backend connections to Fabric and IPFS. This addresses the lack of integrated, evidence-centric access control in many earlier CoC prototypes, and aligns with Zero-Trust access-mediation patterns and forensic-gateway concepts discussed in NIST SP 800-207 and related IoT/forensics frameworks [9, 26, 27].

Evidence registration on the hot-chain with off-chain storage
The system shall allow authorised Investigators (LabOrg users) to register new digital evidence through JumpServer. On upload, JumpServer shall compute a cryptographic hash, send the file to the private IPFS cluster over mTLS, verify that the returned CID corresponds to the computed hash, and then invoke a `CreateEvidence` transaction on the hot-chain's public chaincode to store CID, hash, case identifier, timestamps, and minimal metadata in the world state. Only Investigator role shall be permitted to create or transfer evidence on the hot-chain. This general pattern, on-chain hashes and metadata, off-chain evidence files, is consistent with prior blockchain-IPFS or cloud-backed designs [13, 23, 24], but here it is explicitly bound to per-role policies and organisational MSPs, which several reviews indicate is often missing or only loosely sketched [1, 10, 26].

Immutable custody log, world state, and invalidation without deletion
For each custody action (create, transfer, archive, invalidate, reactivate, access), the public chaincode shall append a structured custody event to the evidence history and update the world-state document keyed by `caseID:evidenceID` with fields such as status (ACTIVE, ARCHIVED, INVALIDATED), CID, hash, current holder organisation/role, and timestamps. CouchDB JSON documents and indexes shall support queries like "list all ACTIVE evidence for case X" without scanning all blocks. The system shall also provide an `InvalidateEvidence` operation: when invoked by an authorised Investigator, it shall mark the evidence as INVALIDATED, record the reason and reference to the original transaction (or CID), and preserve both the original upload and the invalidation event on the ledger. Invalidated items shall be hidden from normal investigator listings but remain fully visible to auditors and courts. This addresses a limitation in earlier work, where mistaken uploads were either ignored as a practical problem or handled off-chain, by providing a non-destructive invalidation pattern recommended in recent evidence-management and admissibility discussions [1, 9, 10, 18, 26].

On-chain role/organisation checks and court-controlled archival
Smart contracts shall enforce both organisation-level and role-level constraints before any state change. At minimum:

- Only Investigator role in LabOrg shall create or transfer evidence on the hot-chain.
- Only Court roles shall finalise archival on the cold-chain.
- Only the current custodian organisation shall invoke `TransferCustody` for a given item.

The system shall allow Court to initiate archival of all evidence in a case. For each item, JumpServer shall retrieve the relevant hot-chain history and submit an `ArchiveToCold` transaction on the cold-chain. The cold-chain endorsement policy shall require signatures from both LabOrg and CourtOrg peers before committing the archival, so that no single party can create, modify, or reactivate long-term records on its own. This requirement directly responds to the need for multi-party control and judicial oversight highlighted in B-CoC, Forensic-Chain, and more recent court-oriented frameworks such as two-level blockchain CoC systems and sustainability-focused CoC proposals [13–15, 17, 18].

Evidence verification and custody-path checking
The system shall allow Court and Auditors to verify evidence on demand. For a basic integrity check, JumpServer shall read the CID and hash from the ledger, fetch the file from IPFS over mTLS, recompute the hash, and flag any mismatch as an integrity failure. For deeper audits (e.g. court-mandated custody-path reports, anomaly investigations), JumpServer shall call a private smart-contract component running inside a TEE to reconstruct the full custody path from on-chain events, apply confidential predicates (such as role- and time-based checks), and return a signed verdict. This addresses the limited support for provenance analytics and risk-aware path validation in many first-generation CoC systems, where the ledger stores events but does not provide structured, court-friendly verification workflows [13–15, 18, 21, 22].

Split public/private smart-contract logic with TEE attestation
The system shall separate smart-contract responsibilities into:

1. A public chaincode that records custody events, enforces visible endorsement and role rules, and updates the CouchDB world state.
2. A private chaincode executed inside SGX enclaves on endorsing peers, responsible for sensitive operations such as verification predicates, cryptographic checks, and protection of high-value keys.

Before any private result is accepted, JumpServer shall perform remote attestation of the enclave and ensure that verification verdicts are signed by a currently attested enclave key; the public chaincode shall reject unverifiable signatures. This split architecture directly answers critiques in TEE-blockchain integrations (e.g. TZ4Fabric and Hyperledger-Fabric trusted-computing analyses) where enclave trust was assumed but not bound tightly to application-level verification and logging [21, 22], and complements court-oriented admissibility frameworks that call for explicit, machine-checkable justification of verification steps [1, 26].

3.4.2 Non-functional Requirements

Performance
The system shall provide performance suitable for DFIR workflows. As a target, end-to-end custody transactions (excluding bulk file upload time) should normally complete within roughly 1–2 seconds, which is consistent with reported latencies in Fabric- and Ethereum-based CoC prototypes operating at modest transaction rates [13–15, 18]. The Fabric ordering service (RAFT) shall be tuned (block size, batch timeout, endorsement policies) to sustain at least tens of transactions per second for custody events, leaving headroom relative to experimental setups that have demonstrated higher throughput on permissioned ledgers [13, 14, 18].

Security
All communication paths (JumpServerFabric, JumpServerIPFS, peerorderer) shall use mTLS with X.509 certificates. Certificates shall be short-lived, rotated periodically, and revoked via CRLs if compromise is suspected. SGX enclaves shall protect selected private keys (Fabric CA's, orderer, verification keys) against extraction, and all enclave binaries shall be remotely attested before use [21, 22]. RBAC shall be enforced at both the JumpServer layer and in chaincode, reducing the risk that a single misconfigured component bypasses access-control rules. Evidence integrity shall always be verifiable through on-chain hashes and CIDs, in line with preservation and integrity requirements emphasized in blockchain-forensics surveys and proposals for digital evidence handling [1, 8–10, 26].

Scalability and Reliability
The architecture shall scale horizontally by adding Fabric peers, orderer nodes, and IPFS nodes as case volume grows. The hot/cold separation is intended to limit ledger growth on active channels and has been proposed in similar form in two-level or tiered CoC architectures to prevent slow queries [13, 17, 18]. The ordering service shall run RAFT with at least three nodes to provide deterministic finality and tolerate individual failures [18, 21]. Evidence objects shall be pinned on multiple IPFS nodes in the private cluster so that loss of a single node does not affect availability, addressing durability and governance concerns raised about public IPFS and cloud-only backends in prior hybrid designs [13, 23, 24]. Ledger

data, world state, and IPFS content shall be backed up according to institutional retention policies.

The JumpServer gateway shall support horizontal scaling behind a load balancer so multiple JumpServer instances can share user traffic and avoid being a single point of failure. The load balancer shall perform health checks and route around failed instances to avoid any DFIR workflow interruptions. The load balancer deployment shall support redundant pair or clustering in front of the JumpServers to avoid the carried-on single point of failure issue.

Compliance and Auditability
The system shall produce an append-only, chronological record of all evidence-related actions, including reads, to support legal criteria of integrity, traceability, and non-repudiation for digital evidence [1, 8, 9, 26]. Every operation shall be attributable to a specific user identity and role, as required by chain-of-custody guidelines and by analyses of blockchain applicability in forensic casework [1, 8]. Archival and reactivation actions on the cold-chain shall always bear explicit court-level endorsements, aligning with proposals that stress judicial oversight, transparent justification, and admissibility-oriented metadata in blockchain-based CoC frameworks [1, 10, 26].

3.4.3 Testing and Verification Requirements

The system shall support:

- Unit tests for all chaincode functions (e.g. `CreateEvidence`, `TransferCustody`, `InvalidateEvidence`), checking correct state transitions and enforcement of MSP/role rules.
- End-to-end workflow tests that exercise typical scenarios: investigator upload → IPFS store → hot-chain record; court-driven archival → cold-chain record; auditor verification → hash comparison and TEE-based custody-path verdict.
- Ledger consistency checks, periodically comparing block height and state hashes across peers, to detect consensus or implementation faults.
- Tamper-detection tests, such as deliberately altering an IPFS file or attempting unauthorised operations, which must be detected and logged.
- Attestation tests, where enclaves with non-whitelisted measurements are presented and must be rejected by JumpServer and by public chaincode when verifying signatures.

These testing requirements are intended not only to validate correctness but also to demonstrate, in a reproducible way, that the implementation closes specific gaps reported in previous CoC prototypes, such as weak off-chain handling, implicit access-control assumptions, and unenforced TEE trust models [13, 14, 17, 18, 21, 22].

3.4.4 Deployment

The initial deployment target is on-premises infrastructure (local servers or VMs), with no dependency on public cloud services. At minimum, the system shall be deployable on:

- at least 2 JumpServer VM's/instances behind an internal load balancer
- separate Fabric CAs for LabOrg and CourtOrg
- a RAFT orderer cluster (3 nodes)
- one peer per organisation per channel (hot and cold) with CouchDB world state
- a small private IPFS cluster (3–4 nodes) on the same LAN
- SGX-capable hosts for private chaincode

During the implementation phase, all components were deployed as Docker containers and orchestrated via Docker Compose for reproducible setup in testing system connections.

3.5 Deliverables

At the end of the project, the following concrete deliverables are expected:

1. Operational DFIR CoC Platform
A working instance of the JumpServer-mediated system in a lab environment, including:

- JumpServer gateway with MFA, RBAC, and web UI
- Dual Hyperledger Fabric networks (hot and cold channels) with RAFT ordering and CouchDB world state
- Private IPFS cluster integrated via the secured gateway

- Deployed TEE-backed private smart-contract components (enclave binaries and attestation setup).

2. Source Code and Configuration Artefacts
Full source code for:

- JumpServer backend and web interface
- Public and private chaincode
- Enclave verification logic and Casbin policies

Config files for Fabric, CAs, IPFS cluster, and JumpServer.

3. Sample DFIR Case Package
Demo cases including:

- Sample evidence files ingested through the full pipeline
- Corresponding on-chain custody histories on hot and cold chains
- Example invalidation, transfer, archival, and verification scenarios.

4. Testing and Evaluation Report
A concise report summarizing:

- Functional test results (upload, transfer, archival, invalidation, verification)
- Performance measurements (end-to-end latency and basic throughput under test load)
- Security/attestation checks (enclave verification, hash mismatch detection, RBAC enforcement).

5. Technical Documentation and User Guides

- System architecture and trust diagrams (including context, sequence, and split-contract views)
- Deployment and operations guide (setup steps, certificate and key management, backup/restore)
- Short usage guides for key stakeholders (investigators, court users, auditors, and admins).

Chapter 4

Technical/Non-Technical Constraints and Applicable Standards

This chapter highlights the technical and non-technical constraints that have shaped our project, as well as the applicable standards from global organizations that will govern our implementation. The constraints outlined here directly stem from the architectural decisions and requirements established in Chapter 3, and they reflect both the capabilities and limitations inherent in deploying a permissioned blockchain-based chain of custody system for digital forensic evidence.

4.1 Project Constraints

The constraints below are categorized by priority to reflect their impact on system viability and legal admissibility. **Critical constraints** must be satisfied for the system to function in a forensically sound manner and meet court admissibility standards. **Important constraints** significantly affect system performance, scalability, and operational costs but can be managed through design trade-offs, and **desirable constraints** improve system quality but are not strictly required for minimal viable deployment.

4.1.1 Technical Constraints

4.1.1.1 Computational Resources (Critical)

Blockchain operations are computationally expensive due to hashing, signature verification, and consensus with other nodes. Our implementation requires at least 4 to 6 nodes to sustain BFT (Byzantine fault tolerance) or RAFT consensus with tolerable latency [28, 29]. Each node must provide a minimum of 4 vCPUs and 8GB RAM, which equates to 24 vCPUs and 48GB RAM for the hot chain cluster of 6 nodes, and 12 vCPUs and 24GB RAM for the cold chain cluster of 3 nodes [29, 30]. For the IPFS storage, 3 nodes are required, and each node must have at least 2 vCPUs and 6GB RAM, and higher RAM is advisable for large evidence sets and orchestration via IPFS Cluster [31, 32].

These constraints are crucial to keep in mind as resource demand increases linearly with block size and transaction rate. Fabric benchmarking shows that transaction latency doubles beyond 3000 TPS without upgrading hardware [29], and peak load scenarios may require burst capacity: for example, during simultaneous evidence uploads from multiple investigations, CPU utilization may spike to 90%+. Our design accommodates this through elastic scaling of non-validator nodes, such as client-facing peers that handle evidence submission and retrieval requests without participating in consensus.

4.1.1.2 Storage and Data Retention (Critical)

Each investigation or case may generate tens or hundreds of GBs of data, and chain metadata adds overhead due to replication across nodes. For a 6-node network maintaining both a hot chain and a cold chain, there is a 6x replication factor of total storage [33]. On-chain data grows at roughly 2–5 GB per 1 million transactions [29, 34], and off-chain evidence requires persistent pinning, which is telling an IPFS node to store the data indefinitely. It was shown that a 1TB IPFS cluster with 3x replication factor requires around 3TB of physical storage, in addition to roughly 20GB of metadata [31, 35].

Furthermore, for long-term retention, we must employ multi-tier storage: SSDs for active evidence and HDDs for archives to ensure a balance between costs and performance. Additionally, retention policies must balance legal requirements, which range from 2–5 years for active cases, against storage costs [36]. Our tiered approach uses SSDs for hot chain metadata and recent evidence, with automatic migration to lower-cost HDD or object storage for cold chain archives older than a set amount of days. All in all, our project requires approximately 5–7TB of storage, as well as budget to increase storage by 1TB per 100 cases retained.

4.1.1.3 Network Bandwidth and Latency (Important)

In our implementation, speed is governed by the network, as is the nature of blockchain. Every transaction has to be endorsed by peers, ordered, then sent to all peers to be committed to the blockchain. If the link between nodes is slow or has high delay, meaning high RTT (round trip time), then everything becomes much slower: throughput decreases and confirmation time increases [29, 33]. Then, Fabric requires low-latency, high-bandwidth links so as to not reduce performance [37].

The same principle goes for off-chain storage using IPFS, as upon adding or pinning large evidence files, the total bytes sent is multiplied by the number of copies, which is the number of IPFS nodes. We should expect bursts of network traffic during ingest and replication, and this is made faster by increasing link speed and throughput, and by placing IPFS nodes closer together geographically to shorten the physical distance packets need to travel between them [32, 35].

Baseline bandwidth requirements are as follows: 100 Mbps minimum for hot chain consensus traffic, 1 Gbps recommended for IPFS nodes handling multi-GB evidence files. All in all, the hot chain necessitates the best links to ensure performance, the cold chain's link requirements are not too stringent but still need good link speed and throughput for good performance, while the IPFS links must have large throughput and bandwidth to support large file transfers (such as large disk images).

4.1.1.4 Cloud Subscription and Operational Costs (Important)

In case we decide to host the VMs running the hot and cold nodes, the IPFS nodes and database, as well as the jump servers, then the monthly bill could add up from compute charges based on per-instance pricing, block storage billed at GB per month rates, and network egress traffic charged per GB [38, 39]. Cloud costs scale with the number of nodes, storage volume, and data transfer, particularly when clients download evidence, resulting in possibly massive per-GB data transfer out costs. Budget planning must account for ongoing operational expenses rather than one-time capital expenditure.

4.1.1.5 Enclave-Specific Requirements (Important)

Enclaves are constrained by hardware and instance availability and strict I/O models. For example, AWS Nitro Enclaves have no network or storage and only communicate with the parent VM, so all external calls must be proxied by the jump server [40]. They also have tight secure-memory budgets, around 100MB for example for SGX enclaves; exceeding it causes paging and large slowdowns, which forces enclave code to stay small [41]. Planning to use an enclave on every jump server, costs would add up linearly with amount of redundancies. These constraints limit which chaincode logic can realistically execute inside TEEs and require careful memory profiling during development.

4.1.1.6 Disaster Recovery and Geographic Distribution (Important)

To ensure availability under node failures or regional outages, our consortium must distribute validators and IPFS storage across at least two geographically separated data centers or cloud regions. This introduces latency trade-offs, as inter-region network links typically exhibit 50–200 ms RTT compared to sub-10 ms within a single region [42]. Consensus protocols like RAFT are sensitive to high latency, so validator placement must balance fault tolerance against performance degradation. Additionally, backup and recovery procedures must ensure that evidence and blockchain state can be restored within a defined Recovery Time Objective (RTO) of 4 hours and Recovery Point Objective (RPO) of 1 hour, consistent with NIST SP 800-34 guidelines for contingency planning [43].

4.1.2 Non-Technical Constraints

4.1.2.1 Health and Safety (Critical)

Beyond software, real-life evidence handling needs safe procedures, from collecting, moving, storing, and documenting evidence. It is also crucial to consider protecting the participating parties from hazardous material and/or disturbing content during acquisition or analysis, so clear SOPs (standard operating procedures) must be in place [44, 45].

4.1.2.2 Political, Legal, and Jurisdictional (Critical)

Where we store the evidence and who may access it depends on jurisdiction: deployments should choose regions consciously, align with prosecutors and courts on admissibility and CoC requirements, as well as agree concerning data retention obligations [46]. All of these vary across countries, so it is important to learn about the law in the country in question and stick to its procedures.

4.1.2.3 Economic (Important)

Running our hot and cold nodes, IPFS storage, databases, and jump servers in the cloud means ongoing costs for compute, storage, and data egress, as highlighted in the previous section. The costs especially increase when clients download evidence, resulting in possibly massive per-GB data transfer out costs. Budget for the cloud approach must be calculated and differs from vendor to vendor [38,39]. Economic feasibility also depends on the number of participating organizations, which means decreased total cost per additional participant.

4.1.2.4 Sustainability (Desirable)

The system, by nature, keeps always-on nodes and retains evidence for long periods due to the cold chain functionality, so we must manage storage growth and energy use. It is important for us to clearly identify what data is permissible to be deleted from the cold chain as per the stakeholder requirements, or else storage and its footprint only grows [47].

4.1.2.5 Social Impact and Access to Justice (Desirable)

It is believed that this system affects societal trust in law enforcement and the judicial process. Blockchain transparency, when properly scoped, can improve public confidence by providing verifiable audit trails that demonstrate evidence was not tampered with. However, the digital divide poses a risk: smaller jurisdictions with limited IT resources or funding may be disadvantaged, unable to deploy or maintain the infrastructure required to participate in the consortium. This could create a two-tier justice system where well-resourced agencies benefit from advanced CoC protections while under-resourced jurisdictions continue relying on vulnerable paper-based or centralized systems. To the best of our knowledge, deployment strategies should consider this gap to ensure equitable access for all.

4.1.3 Constraint Interdependencies and Trade-offs

Some constraints interact and create design trade-offs that must be managed carefully:

- **IPFS replication vs. storage costs:** More IPFS replication improves availability and fault tolerance, ensuring evidence remains accessible even if nodes fail, but it linearly increases storage costs.
- **Geographic distribution vs. consensus latency:** Distributing validators across regions improves disaster recovery but increases inter-node latency, which slows consensus and reduces overall TPS.

- **Retention duration vs. sustainability:** Longer retention periods satisfy legal obligations but drive storage growth and energy consumption, requiring tiered archival strategies.

It can be inferred that these trade-offs are navigated through careful capacity planning, performance benchmarking, and policy decisions encoded in Fabric channel configurations and private data collection settings.

4.2 Applicable Standards

In this section, we outline how our project scope is governed by international standards, such as ISO, IEEE, NIST, and FIPS. We explain how each one relates to specific components of our architecture and ensure that our system meets both technical and legal requirements for digital evidence admissibility.

4.2.1 ISO 23257:2022 – Blockchain Reference Architecture

This international standard defines a general reference architecture for blockchain systems; it provides a common blueprint for designing permissioned ledgers and their roles, such as nodes, users, admins, and external actors. Our system's dual-chain design, consisting of a hot chain for active investigative metadata and a cold chain for archival records, aligns with this reference architecture's principles. By following this standard's framework, our system ensures its blockchain layers are structured according to industry best practices, facilitating interoperability and governance across multi-organizational deployments [48].

4.2.2 IEEE 2418.2-2020 – Blockchain Data Format

This international standard specifies blockchain records' data structure and format requirements. The aim is to ensure that transactions, blocks, and related metadata are represented consistently; the hot and cold chains can adopt this standard's format to remain internationally recognized as a formal system, and this is by implementing standardized block headers, hash linking, and metadata fields [49].

4.2.3 FIPS 180-4 (2015) – Secure Hash Standard

This NIST standard defines the Secure Hash Standard, as it specifies approved cryptographic hash algorithms like SHA-1 and SHA-2 algorithm families, although NIST has decided to revise this standard by removing the SHA-1 specification. Our proposed system relies heavily on hash algorithms, whether it be to link blocks in hot and cold chains, computing CIDs of evidence stored on IPFS, hashing log entries and audit records, and in general, relying on secure hashing to detect tampering. Then, this standard directly maps to the integrity guarantees required for CoC, and complying with the denoted hashing algorithms results in evidence validation and maintaining court-admissibility [50]. We exclusively use SHA-256 and SHA-512 from the SHA-2 family, avoiding deprecated SHA-1.

4.2.4 FIPS 186-5 (2023) – Digital Signature Standard

This NIST standard defines the digital signature standard, which defines the algorithms that can and must be used to generate a digital signature, such as RSA, ECDSA, and EdDSA. These signatures provide authenticity and integrity for electronic data, as per the standard. In our proposed system, we rely on digital signatures in multiple aspects: investigators and devices sign evidence submissions, blockchain transactions are signed by clients and endorsed by peers in Fabric, the jump server uses mutual TLS with certificates, and so on. Aligning with this standard also provides non-repudiation, as a signer cannot later deny their action. This approach favors the credibility of evidence and court-admissibility since it follows an exact standard designed for legal-grade security [51].

4.2.5 TLS 1.3: Transport Layer Security (RFC 8446, 2018) – IETF

TLS 1.3 is the latest standard protocol for securing communications, including mutual authentication, encryption, and integrity for data communicated. This is especially useful for our proposed system's jump server implementation, as its mutual TLS handshake, known as mTLS, for client authentication and establishing a secure communication is an implementation of TLS 1.3. By adhering to RFC 8446, we ensure that all evidence transfers remain encrypted and protected against eavesdropping or tampering by MITM attacks. Therefore, the gateway, which is the jump server, aligns with industry best practices for secure communication [52].

4.2.6 NIST SP 800-207 (2020) – Zero Trust Architecture

This special publication outlines the zero-trust security model, which asserts that no user or system should be trusted by default, even if inside a network perimeter, and that access to resources must be continuously authenticated and authorized. In our proposed system, the jump server implementation is explicitly built on zero trust principles: it requires mTLS certificates before allowing any access to ledgers or storage. By following NIST's zero trust model, we ensure strong identity verification and strict access control, which results in a securely hardened digital CoC system [53].

4.2.7 ISO/IEC 27041:2015 and ISO/IEC 27042:2015 – Forensic Analysis

These international standards provide guidance for forensic examiners to ensure that their tools and methods are fit for the purpose, and that their analysis of evidence preserves data integrity and is properly documented. Our proposed system aligns well with these standards, as there is no access or modification to any component behind the jump server that is not logged, authenticated, and timestamped. Moreover, the use of standard hashing also adheres to these standards as integrity is always validated, and tampering can be easily detected by comparing hashes in conjunction with digital certificates and timestamps. Our future work will also stem from these standards, as they are the base of our scope and proposed solution [54].

4.2.8 ISO/IEC 27037:2012 – Digital Evidence Guidelines

ISO/IEC 27037 provides guidelines for the identification, collection, acquisition, and preservation of digital evidence in a manner that ensures admissibility in legal proceedings. It specifies that evidence handlers must document every action taken, maintain integrity through cryptographic hashing, and ensure continuity of custody. Our blockchain-based CoC system operationalizes these guidelines: every custody event is logged with cryptographic signatures and timestamps, evidence hashes are computed at acquisition and verified at each transfer, and the immutable ledger provides a complete audit trail. Compliance with ISO/IEC 27037 strengthens the legal defensibility of evidence managed through our platform [55].

4.2.9 IEEE P2418.6 – Blockchain Cybersecurity Framework (Emerging)

IEEE P2418.6 is an emerging standard focused on blockchain cybersecurity, addressing threat models, risk management, and security controls specific to distributed ledger technologies. Although still under development, we monitor this standard to ensure our access control mechanisms, consensus security, and enclave-based confidentiality measures align with evolving best practices. Once finalized, this standard will provide additional validation criteria for our Jump Server authentication and role-based access control implementations [56].

4.2.10 NIST SP 800-34 – Contingency Planning (Disaster Recovery)

NIST SP 800-34 provides guidelines for information system contingency planning, including backup strategies, disaster recovery procedures, and business continuity. Our architecture incorporates these principles through geographically distributed nodes and automated state backups. This standard ensures that our CoC system can recover from catastrophic failures without loss of evidence integrity or availability [43].

4.2.11 Summary of Constraints and Standards

Table 4.1 summarizes the key constraints by category, priority, and design implication. Table 4.2 maps applicable standards to specific system components and their verification methods. These tables provide a quick reference for understanding how constraints and standards shape our architecture and guide implementation decisions.

Table 4.1: Summary of Project Constraints

Constraint Category	**Priority**	**Key Requirement**	**Design Implication**
		Critical Constraints	
Computational Resources	Critical	24 vCPUs, 48 GB RAM (hot chain); 12 vCPUs, 24 GB RAM (cold chain)	Multi-node deployment with elastic scaling for client-facing peers
Storage & Retention	Critical	5–7 TB initial, +1 TB per 100 cases; 2–5 year retention	Tiered storage (SSD for active, HDD for archive); automated migration
Health & Safety	Critical	Safe evidence handling; protect from hazardous content	SOPs per SWGDE guidelines; psychological support protocols
Political, Legal & Jurisdictional	Critical	Compliance with local laws, international regulations	Region-conscious storage; purge policies aligned with regulations
		Important Constraints	
Network Bandwidth	Important	100 Mbps (consensus), 1 Gbps (IPFS)	Geographic node placement; high-bandwidth links
Cloud Costs	Important	Ongoing compute, storage, egress fees	Cost-sharing among consortium members; budget planning
Enclave Requirements	Important	100 MB memory limit; proxy I/O via Jump Server	Minimal enclave code; memory profiling during development
Disaster Recovery	Important	RTO 4 hrs, RPO 1 hr; multi-region deployment	Geographic validator distribution; automated backups
Economic Feasibility	Important	Shared costs across agencies	Consortium funding model; cloud vs. on-premise trade-offs
		Desirable Constraints	
Sustainability	Desirable	Manage storage growth and energy use	Purge policies for private data collections; tiered archival
Social Impact	Desirable	Equitable access across jurisdictions	Shared infrastructure; subsidies for under-resourced agencies

Table 4.2: Summary of Applicable Standards and Their Role in the System

Standard	**Scope**	**System Component(s)**	**Verification Method**
ISO 23257:2022	Blockchain reference architecture	Hot/Cold chain architecture	Architecture review; peer assessment
IEEE 2418.2-2020	Blockchain data formats	Block headers, metadata fields	Format validation; interoperability tests
FIPS 180-4	Cryptographic hashing (SHA-2)	Block linking, IPFS CIDs, audit logs	Hash algorithm verification; code review
FIPS 186-5	Digital signatures (RSA, ECDSA, EdDSA)	Transaction signing, mTLS certs	Signature validation; certificate inspection
TLS 1.3 (RFC 8446)	Secure communication	Jump Server mTLS handshake	TLS version check; cipher suite analysis
NIST SP 800-207	Zero trust architecture	Jump Server authentication (mTLS, RBAC)	Access control audit; penetration testing
ISO/IEC 27041:2015	Forensic tool assurance	Logging, integrity checks	Forensic tool validation; chain of custody audit
ISO/IEC 27042:2015	Forensic analysis procedures	Evidence handling workflows	Process compliance audit
ISO/IEC 27037:2012	Digital evidence guidelines	Evidence acquisition, preservation	Custody event logs; hash verification
IEEE P2418.6 (draft)	Blockchain cybersecurity	Access control, consensus security	Ongoing monitoring; draft compliance review
NIST SP 800-34	Contingency planning	Disaster recovery, backups	RTO/RPO testing; failover drills

Chapter 5

Progress Description

This chapter focuses on the iterative design process for our digital chain-of-custody (CoC) system. From the beginning, several core security fundamentals guided our choices—confidentiality, integrity, availability, authentication, and non-repudiation. These constraints shaped our evaluation of multiple design alternatives before ultimately selecting a permissioned Hyperledger Fabric architecture.

5.1 Preliminary Design Alternatives

5.1.1 Design Alternative 1: FSM-based Logging Chain

Our first proposed alternative was a finite state machine (FSM) based event ledger. Each evidence item is represented as an "entity" whose state evolves through transitions such as "collected," "analyzed," "archived," and "presented," which aligns directly with traditional CoC timelines. Each transition would create a digitally signed state update stored in a centralized database. This direction was initially appealing due to our familiarity with FSM structures and their simple computational properties.

The FSM approach showed several advantages:

- **Low computational overhead:** Because no consensus mechanism is required, transitions rely only on local validation and digital signatures.
- **Easy querying and rollback:** Evidence histories stored in relational or NoSQL databases allow simple query operations and snapshot-based rollback capabilities.

- **Simple software stack:** FSM-based systems can leverage widely supported technologies such as Apache Kafka or MongoDB with versioning logs, reducing deployment complexity.

Despite these strengths, several critical limitations made the FSM approach infeasible for a forensically sound CoC system:

- **Lack of immutability:** Centralized logs can be silently modified by administrators, since no cryptographic linkage exists between entries.
- **No distributed trust:** Because FSMs rely on a single authoritative operator, they fail to satisfy multi-party trust requirements in forensic collaboration.
- **Weak non-repudiation:** Although transitions may be signed, privileged actors could still delete or overwrite logs.
- **Poor cross-organizational scalability:** Without distributed consensus, expanding the system to multiple agencies creates fragile manual trust relationships rather than provable guarantees.

While the FSM model offered simplicity, its security and governance limitations made it unsuitable for a robust CoC environment.

5.1.2 Design Alternative 2: Ethereum-Based Blockchain System

Our second design alternative evaluated the use of Ethereum to record evidence-related transactions. Ethereum supports smart contracts for executing custody operations on a public, globally replicated ledger. Each event would be committed as a transaction containing metadata such as case ID, owner address, and hash of the evidence.

Advantages of this approach included:

- **Immutability and non-repudiation:** Every Ethereum block is cryptographically linked, providing strong tamper-evidence and append-only guarantees.
- **Public auditability:** Any party could independently verify the authenticity of custody records, which aligns with traceability requirements.

However, significant disadvantages prevented Ethereum from being suitable for forensic use:

- **Metadata exposure:** Even if raw evidence is stored off-chain, Ethereum still exposes metadata—hashes, timestamps, wallet addresses—publicly. This compromises confidentiality for active investigations.
- **Lack of jurisdictional control:** Ethereum is permissionless; no authority can control validator behavior or enforce legal governance requirements. For law enforcement use, this is unacceptable.
- **Non-deterministic transaction ordering:** Miners may reorder transactions based on gas fees, disrupting the precise chronological sequence required for admissible forensic timelines.

While Ethereum demonstrated strong correctness guarantees, its transparency and governance limitations led us to explore private blockchain frameworks such as Hyperledger Fabric, which provide controlled access and permissioned consensus.

5.2 Preliminary Implementation and Testing

Although not completed fully, preliminary implementation focused on assembling the core environment using Docker and Docker Compose, launching the dual Fabric networks, and integrating the JumpServer with the IPFS cluster. Early testing centred on validating connectivity, mTLS configuration, and basic chaincode functions such as evidence registration and retrieval. Debugging efforts mainly involved resolving container networking issues, certificate mismatches, and endorsement-policy failures, which are typical in Fabric-based deployments. Open-source tools and reference implementations were reviewed extensively to ensure correct configuration of RAFT, CouchDB world state, and IPFS pinning. These initial tests confirm that the architectural components interoperate as intended and establish a stable foundation for full system integration.

Chapter 6

Conclusion

This report summarizes the work completed in the first semester of our Final Year Project: a permissioned blockchain-based system to record the chain of custody in digital forensic investigations. The system was designed with four main components: a dual-chain Hyperledger Fabric architecture, secure access control to JumpServer, private chaincode verification based on TEE, and file storage of large forensic artifacts in off-chain IPFS. During the semester, we were committed to establishing a solid technical base through extensive literature review, architectural design, and preliminary implementation of major system modules.

From a deployment perspective, significant strides were made in all the components:

- **Hyperledger Fabric Network:** Successful instantiations of both the hot and cold blockchain networks with appropriate organizations (LabOrg, CourtOrg, and OrdererOrg), and peer nodes configured on separate domains were achieved. Communication channels were secured with mTLS.
- **Identity and Access Management:** The certificate authority infrastructure and MSP settings became operational, allowing certificate-based identity management. The RBAC system is being designed to differentiate roles such as investigators, auditors, and court users, with permissions granted accordingly.
- **JumpServer Gateway:** A functional prototype was developed featuring passwordless MFA authentication, certificate-based authorization, and mTLS connections to the backend blockchain and IPFS services—without relying on any external identity provider.
- **Architecture of Smart Contracts:** Chaincode for recording custody events was developed and deployed on both hot and cold channels. Pre-

liminary testing succeeded in evidence creation, transfer, archiving, and invalidation operations.

- **Off-chain Storage:** The private IPFS cluster was configured and tested for content-addressed storage and retrieval. It integrates well with the blockchain metadata layer for cryptographic hash verification.

We made critical architectural decisions throughout the semester, most notably adopting a split contract architecture that separates public custody logging from TEE-based private verification. This approach provides transparency for routine operations while keeping sensitive verification logic protected inside Intel SGX enclaves. We also designed a dual-chain hot/cold architecture that addresses scalability concerns by separating high-frequency investigative operations from low-frequency archival records that require judicial oversight.

Through extensive technical background research presented in Chapter 2, we identified crucial shortcomings in existing blockchain-based chain-of-custody systems: reliance on external identity providers, unscalable off-chain data storage, lack of TEE integration, and weak separation between active and archived evidence. Our design systematically addresses these limitations using integrated security mechanisms and forensic-specific architectural patterns.

Future work will focus on a production-grade deployment model for the entire framework, including replacing the current Docker-based setup with a more robust orchestration environment suitable for institutional adoption. In addition, to the best of our knowledge, no prior DFIR chain-of-custody system has applied graph-based optimisation—such as Dijkstra-driven anomaly detection—to highlight inefficiencies or irregularities in evidence handling. This represents a new research direction that we intend to further formalise and evaluate, especially regarding security, computational overhead, and evidentiary value when executed inside a TEE.

Appendix A

List of Resources and Engineering Tools Needed

A.1 List of Resources and Engineering Tools Needed

This appendix presents the preliminary list of hardware and software resources required for the proposed permissioned blockchain-based chain-of-custody system.

A.1.1 Hardware Resources

A.1.1.1 Current Semester

- **Personal Laptops (x4)** - Used for initial development and prototyping (team-owned, $0)

A.1.1.2 Required for Spring Semester

Due to SSD storage limitations and performance requirements for running multiple blockchain nodes, IPFS clusters, and TEE components, we require:

- **AUB Computer Science Laboratory Resources** or **AUB HPC Access** ($0, requires permission)

A.1.1.3 Specific Components

- **Fabric Network Nodes (x6)** - Hot/cold chain peers and orderers

- **JumpServer Instances (x2)** - Gateway servers with load balancing
- **SGX-Capable Hardware (x2)** - Intel Xeon processors for TEE
- **IPFS Cluster Nodes (x4)** - Distributed storage infrastructure
- **YubiKey Security Keys (x2)** - Hardware authentication (˜$300total)

A.1.2 Software Tools

A.1.2.1 Blockchain (New Learning)

- Hyperledger Fabric v3.0.0 LTS, Fabric CA, CouchDB

A.1.2.2 Smart Contracts (New Learning)

- Go Programming Language, Intel SGX SDK, Fabric Private Chaincode (FPC)

A.1.2.3 Security

- Casbin (RBAC), OpenSSL (mTLS)

A.1.2.4 Storage (New Learning)

- IPFS v0.20+, IPFS Cluster

A.1.2.5 Development (Familiar)

- Docker, Docker Compose, Git/GitHub, VS Code, LaTeX, Draw.io

A.1.3 Total Estimated Cost: ˜$300

(All software is open-source; lab/HPC access requires permission but is free)

Appendix B

Detailed Project Schedule

This chapter provides a concise overview of the full Spring 2025 schedule for completing our blockchain-based chain of custody system. The semester is organized into five phases that progress from TEE integration and private chaincode development, to cross-platform deployment, anomaly detection implementation, full testing and performance evaluation, and finally documentation and presentation preparation. These phases span 17 weeks and collectively define the project's structured development path.

B.1 Gantt Chart Visualization

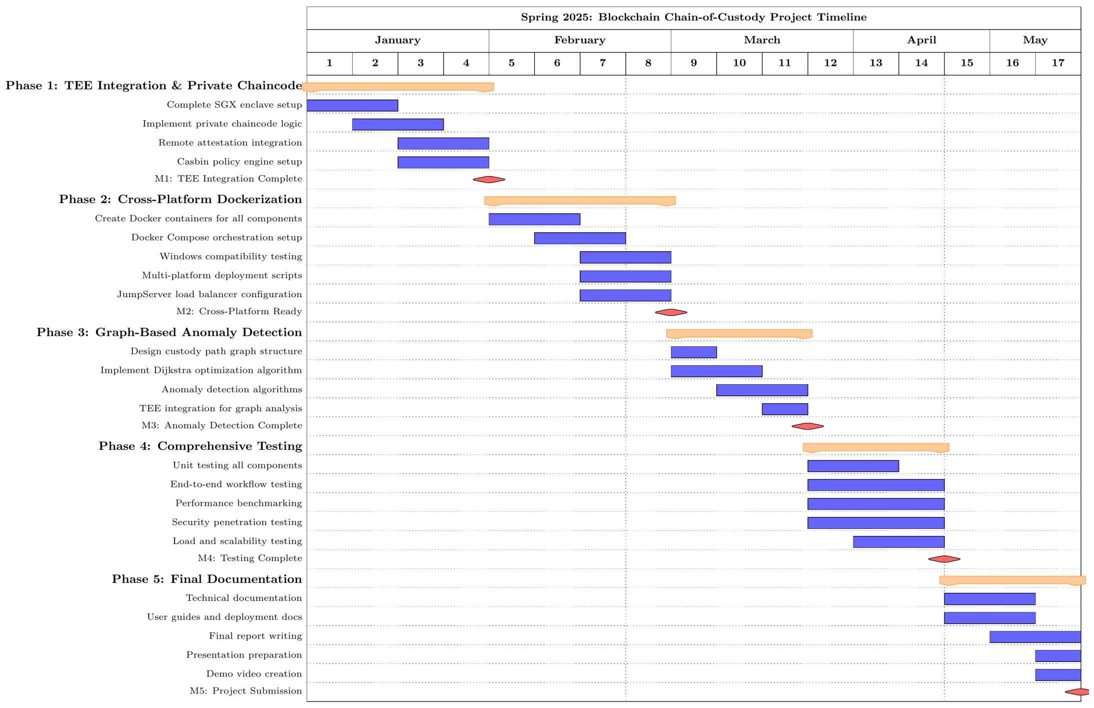


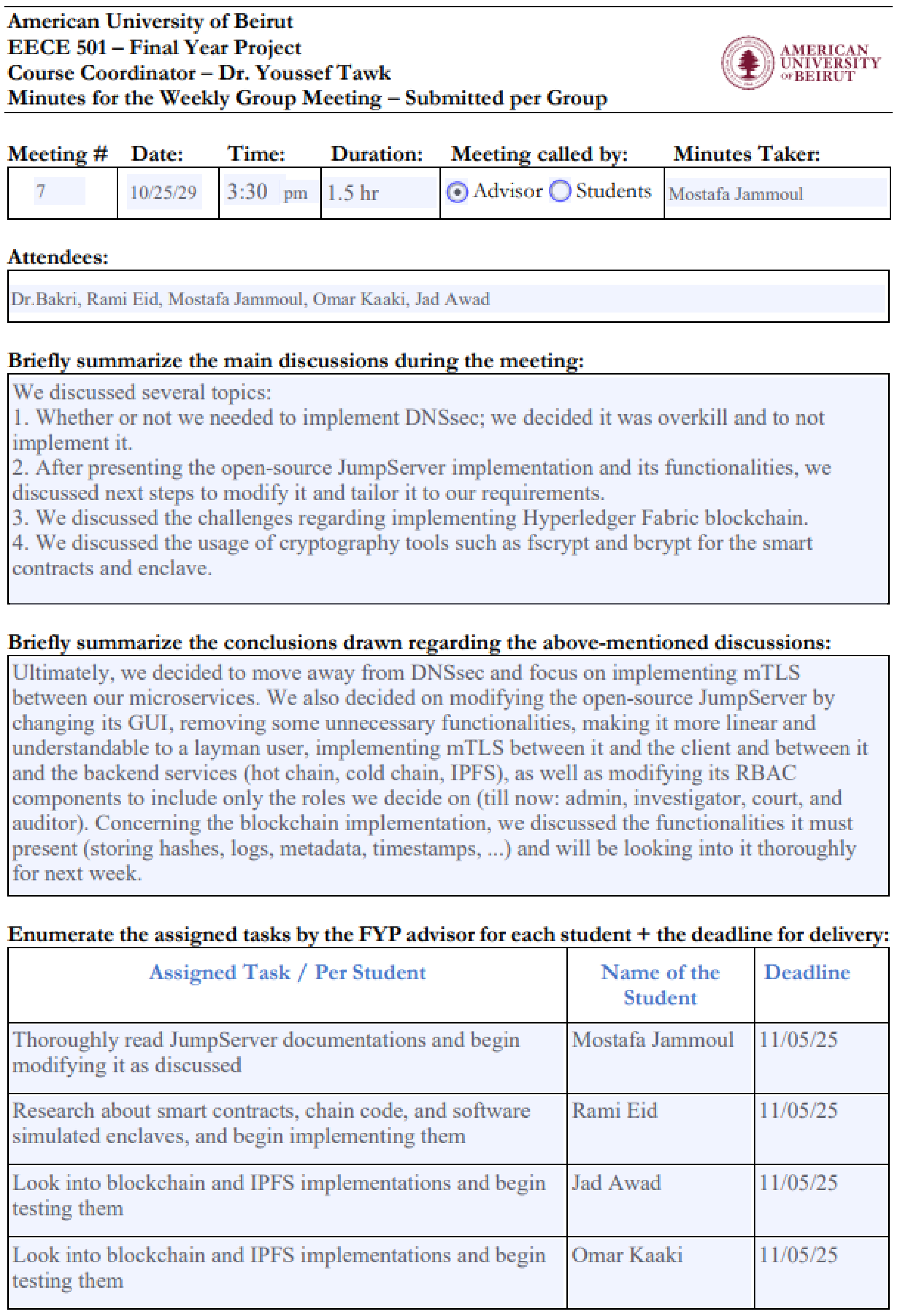
Figure B.1: Spring 2025 Project Gantt Chart

As shown in Figure B.1, the Gantt chart outlines the timeline and dependencies of all five phases, indicating where tasks run in parallel and when major milestones (M1–M5) must be completed. It provides a high-level roadmap that ensures the team remains aligned with deadlines, supports planning across the semester, and guides the progression from development to testing and final delivery.

Appendix C

Weekly Meeting Minutes

We originally uploaded all weekly meeting minutes directly to Moodle. However, since the older submissions were removed from Moodle, we were unable to retrieve the remaining files for which we did not keep local copies (Weeks 2, 4, 8, and 9). The weeks included below (Weeks 1, 3, 5, 6, 7, 10, and 11) are the ones for which we still had copies available, and these were uploaded to Overleaf for inclusion in this report.



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
7	10/25/29	3:30 pm	1.5 hr	[✓ Advisor] Students	Mostafa Jammoul

Attendees:

Dr.Bakri, Rami Eid, Mostafa Jammoul, Omar Kaaki, Jad Awad

Briefly summarize the main discussions during the meeting:

We discussed several topics:
1. Whether or not we needed to implement DNSsec; we decided it was overkill and to not implement it.
2. After presenting the open-source JumpServer implementation and its functionalities, we discussed next steps to modify it and tailor it to our requirements.
3. We discussed the challenges regarding implementing Hyperledger Fabric blockchain.
4. We discussed the usage of cryptography tools such as fscrypt and bcrypt for the smart contracts and enclave.

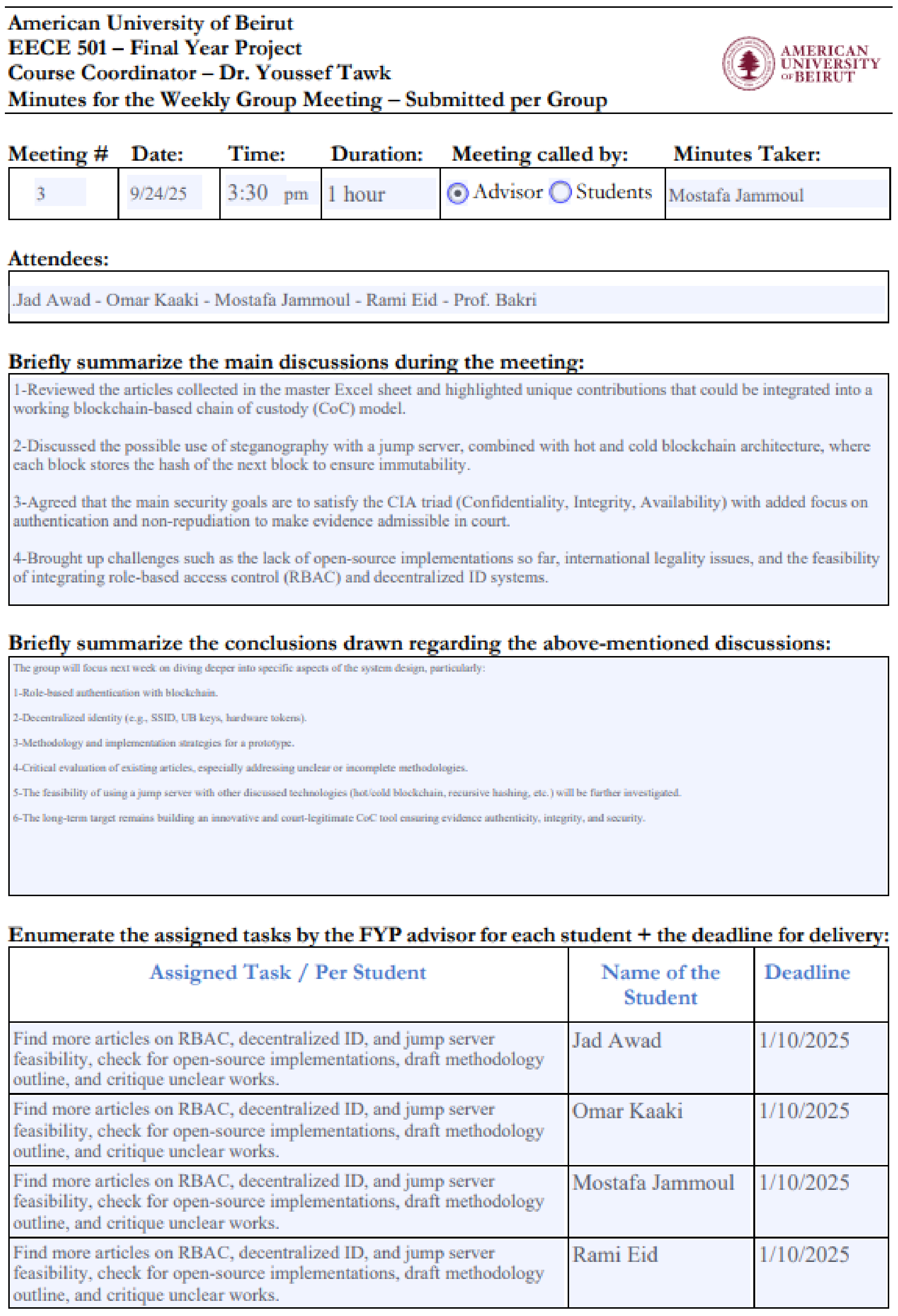
Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

Ultimately, we decided to move away from DNSsec and focus on implementing mTLS between our microservices. We also decided on modifying the open-source JumpServer by changing its GUI, removing some unnecessary functionalities, making it more linear and understandable to a layman user, implementing mTLS between it and the client and between it and the backend services (hot chain, cold chain, IPFS), as well as modifying its RBAC components to include only the roles we decide on (till now: admin, investigator, court, and auditor). Concerning the blockchain implementation, we discussed the functionalities it must present (storing hashes, logs, metadata, timestamps, ...) and will be looking into it thoroughly for next week.

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Thoroughly read JumpServer documentations and begin modifying it as discussed	Mostafa Jammoul	11/05/25
Research about smart contracts, chain code, and software simulated enclaves, and begin implementing them	Rami Eid	11/05/25
Look into blockchain and IPFS implementations and begin testing them	Jad Awad	11/05/25
Look into blockchain and IPFS implementations and begin testing them	Omar Kaaki	11/05/25

Figure C.1: Weekly Meeting Minutes – Week 1



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

AMERICAN UNIVERSITY OF BEIRUT

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
3	9/24/25	3:30 pm	1 hour	[✓ Advisor] Students	Mostafa Jammoul

Attendees:

.Jad Awad - Omar Kaaki - Mostafa Jammoul - Rami Eid - Prof. Bakri

Briefly summarize the main discussions during the meeting:

1-Reviewed the articles collected in the master Excel sheet and highlighted unique contributions that could be integrated into a working blockchain-based chain of custody (CoC) model.

2-Discussed the possible use of steganography with a jump server, combined with hot and cold blockchain architecture, where each block stores the hash of the next block to ensure immutability.

3-Agreed that the main security goals are to satisfy the CIA triad (Confidentiality, Integrity, Availability) with added focus on authentication and non-repudiation to make evidence admissible in court.

4-Brought up challenges such as the lack of open-source implementations so far, international legality issues, and the feasibility of integrating role-based access control (RBAC) and decentralized ID systems.

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

The group will focus next week on diving deeper into specific aspects of the system design, particularly:

1-Role-based authentication with blockchain.

2-Decentralized identity (e.g., SSID, UB keys, hardware tokens).

3-Methodology and implementation strategies for a prototype.

4-Critical evaluation of existing articles, especially addressing unclear or incomplete methodologies.

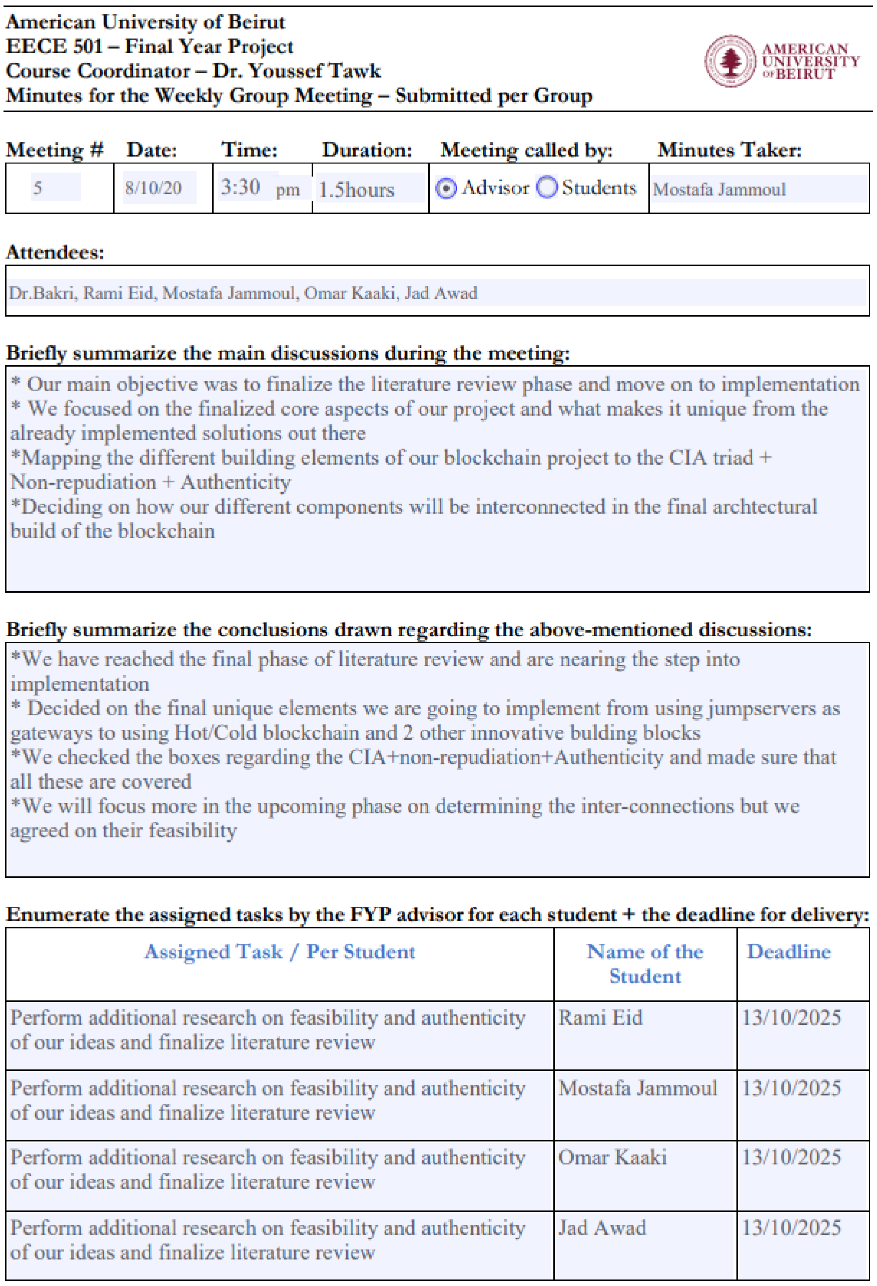
5-The feasibility of using a jump server with other discussed technologies (hot/cold blockchain, recursive hashing, etc.) will be further investigated.

6-The long-term target remains building an innovative and court-legitimate CoC tool ensuring evidence authenticity, integrity, and security.

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Find more articles on RBAC, decentralized ID, and jump server feasibility, check for open-source implementations, draft methodology outline, and critique unclear works.	Jad Awad	1/10/2025
Find more articles on RBAC, decentralized ID, and jump server feasibility, check for open-source implementations, draft methodology outline, and critique unclear works.	Omar Kaaki	1/10/2025
Find more articles on RBAC, decentralized ID, and jump server feasibility, check for open-source implementations, draft methodology outline, and critique unclear works.	Mostafa Jammoul	1/10/2025
Find more articles on RBAC, decentralized ID, and jump server feasibility, check for open-source implementations, draft methodology outline, and critique unclear works.	Rami Eid	1/10/2025

Figure C.2: Weekly Meeting Minutes – Week 3



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
5	8/10/20	3:30 pm	1.5hours	[✓ Advisor] Students	Mostafa Jammoul

Attendees:

Dr.Bakri, Rami Eid, Mostafa Jammoul, Omar Kaaki, Jad Awad

Briefly summarize the main discussions during the meeting:

* Our main objective was to finalize the literature review phase and move on to implementation
* We focused on the finalized core aspects of our project and what makes it unique from the already implemented solutions out there
*Mapping the different building elements of our blockchain project to the CIA triad + Non-repudiation + Authenticity
*Deciding on how our different components will be interconnected in the final archtectural build of the blockchain

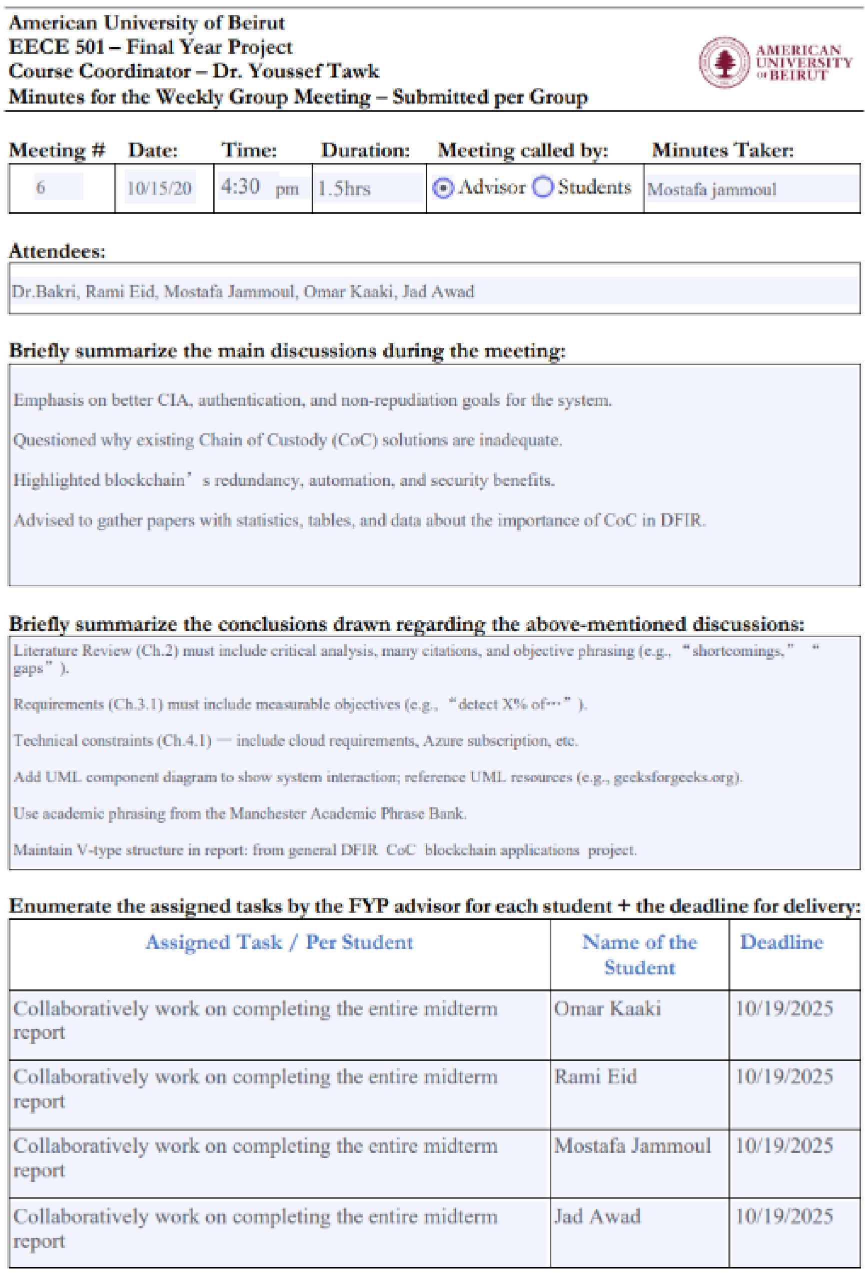
Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

*We have reached the final phase of literature review and are nearing the step into implementation
* Decided on the final unique elements we are going to implement from using jumpservers as gateways to using Hot/Cold blockchain and 2 other innovative bulding blocks
*We checked the boxes regarding the CIA+non-repudiation+Authenticity and made sure that all these are covered
*We will focus more in the upcoming phase on determining the inter-connections but we agreed on their feasibility

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Perform additional research on feasibility and authenticity of our ideas and finalize literature review	Rami Eid	13/10/2025
Perform additional research on feasibility and authenticity of our ideas and finalize literature review	Mostafa Jammoul	13/10/2025
Perform additional research on feasibility and authenticity of our ideas and finalize literature review	Omar Kaaki	13/10/2025
Perform additional research on feasibility and authenticity of our ideas and finalize literature review	Jad Awad	13/10/2025

Figure C.3: Weekly Meeting Minutes – Week 5



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
6	10/15/20	4:30 pm	1.5hrs	[✓ Advisor] Students	Mostafa jammoul

Attendees:

Dr.Bakri, Rami Eid, Mostafa Jammoul, Omar Kaaki, Jad Awad

Briefly summarize the main discussions during the meeting:

Emphasis on better CIA, authentication, and non-repudiation goals for the system.

Questioned why existing Chain of Custody (CoC) solutions are inadequate.

Highlighted blockchain’ s redundancy, automation, and security benefits.

Advised to gather papers with statistics, tables, and data about the importance of CoC in DFIR.

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

Literature Review (Ch.2) must include critical analysis, many citations, and objective phrasing (e.g., “shortcomings,” “gaps”).

Requirements (Ch.3.1) must include measurable objectives (e.g., “detect X% of…”).

Technical constraints (Ch.4.1) — include cloud requirements, Azure subscription, etc.

Add UML component diagram to show system interaction; reference UML resources (e.g., geeksforgeeks.org).

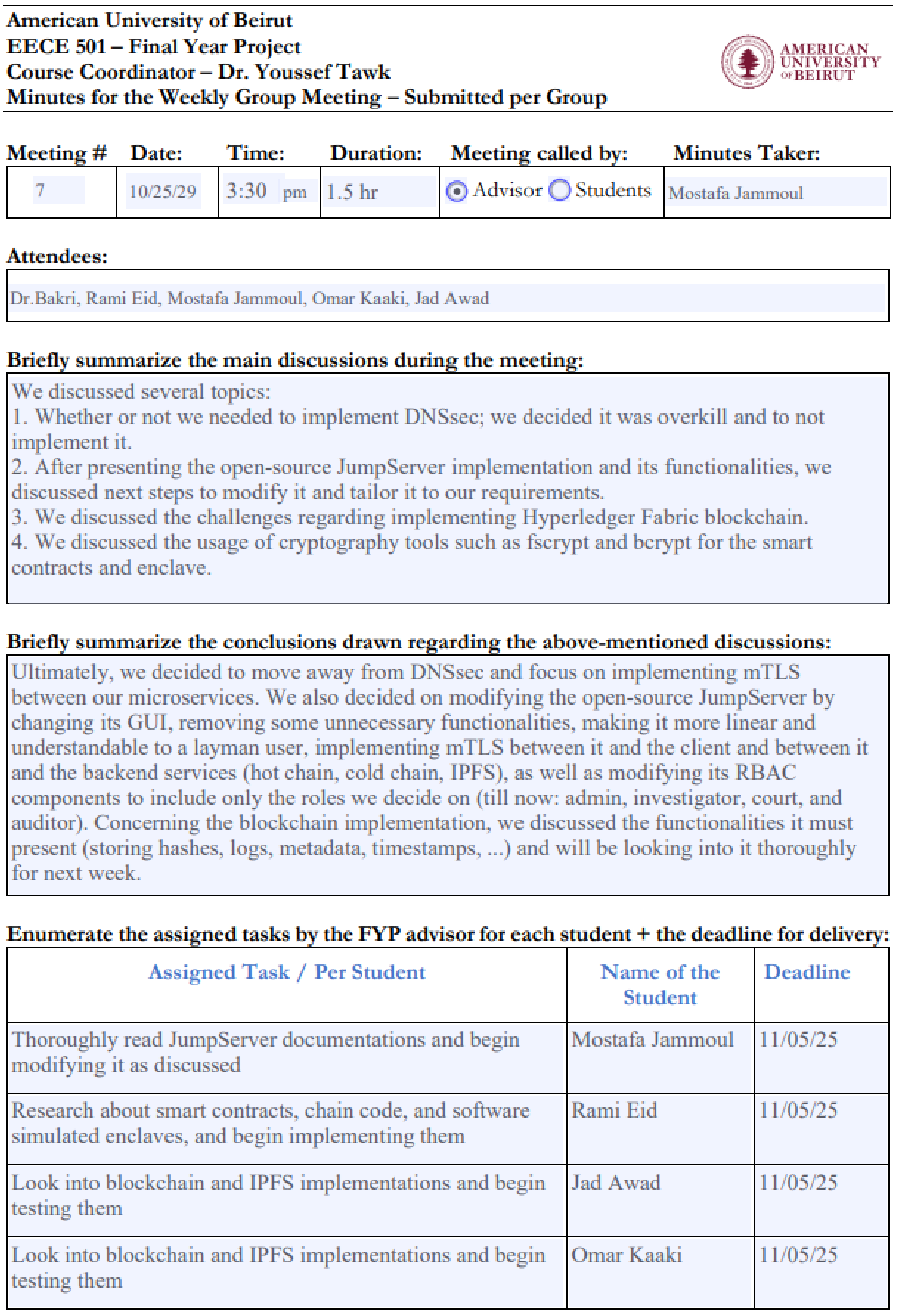
Use academic phrasing from the Manchester Academic Phrase Bank.

Maintain V-type structure in report: from general DFIR CoC blockchain applications project.

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Collaboratively work on completing the entire midterm report	Omar Kaaki	10/19/2025
Collaboratively work on completing the entire midterm report	Rami Eid	10/19/2025
Collaboratively work on completing the entire midterm report	Mostafa Jammoul	10/19/2025
Collaboratively work on completing the entire midterm report	Jad Awad	10/19/2025

Figure C.4: Weekly Meeting Minutes – Week 6



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
7	10/25/29	3:30 pm	1.5 hr	[✓ Advisor] Students	Mostafa Jammoul

Attendees:

Dr.Bakri, Rami Eid, Mostafa Jammoul, Omar Kaaki, Jad Awad

Briefly summarize the main discussions during the meeting:

We discussed several topics:
1. Whether or not we needed to implement DNSsec; we decided it was overkill and to not implement it.
2. After presenting the open-source JumpServer implementation and its functionalities, we discussed next steps to modify it and tailor it to our requirements.
3. We discussed the challenges regarding implementing Hyperledger Fabric blockchain.
4. We discussed the usage of cryptography tools such as fscrypt and bcrypt for the smart contracts and enclave.

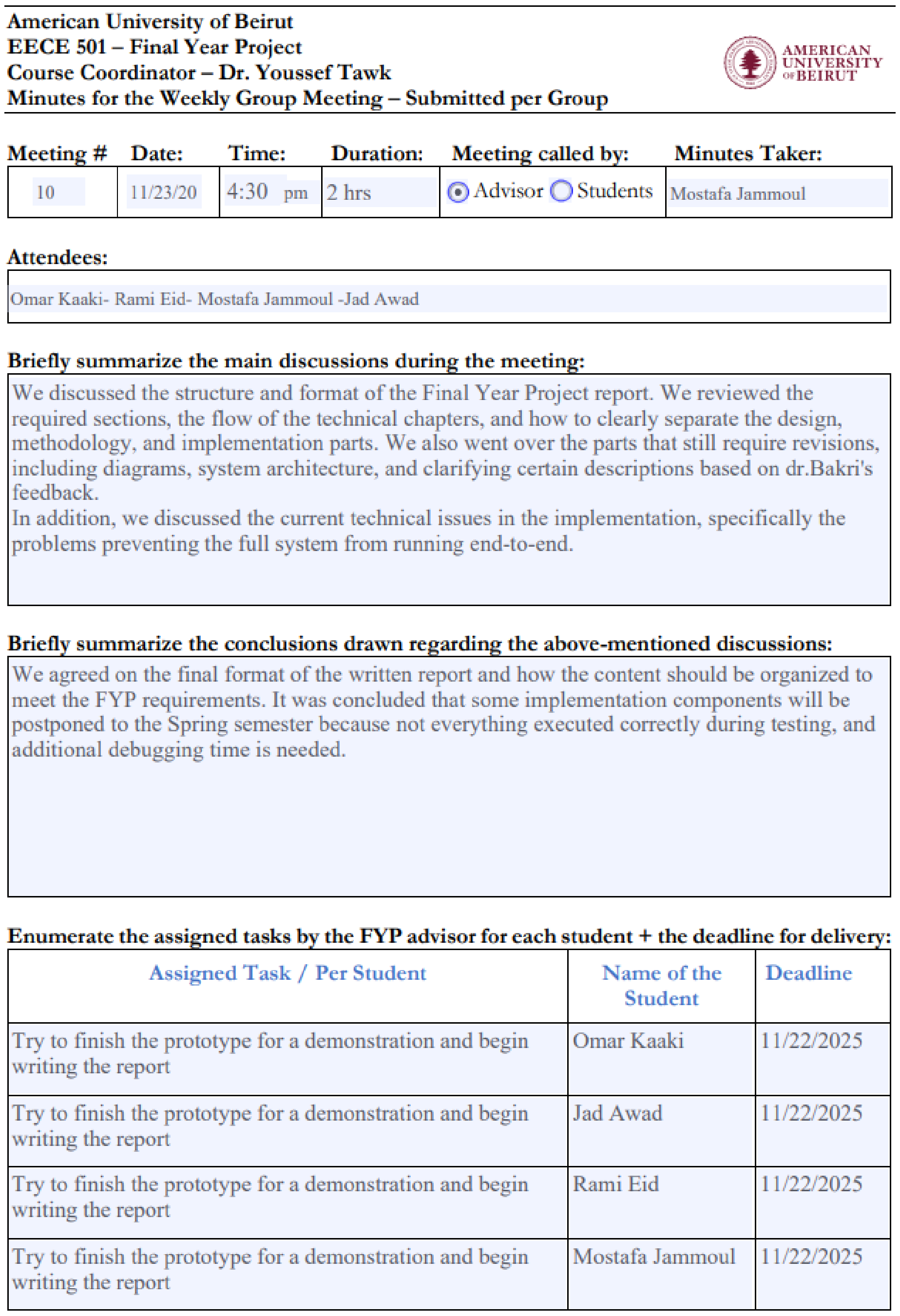
Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

Ultimately, we decided to move away from DNSsec and focus on implementing mTLS between our microservices. We also decided on modifying the open-source JumpServer by changing its GUI, removing some unnecessary functionalities, making it more linear and understandable to a layman user, implementing mTLS between it and the client and between it and the backend services (hot chain, cold chain, IPFS), as well as modifying its RBAC components to include only the roles we decide on (till now: admin, investigator, court, and auditor). Concerning the blockchain implementation, we discussed the functionalities it must present (storing hashes, logs, metadata, timestamps, ...) and will be looking into it thoroughly for next week.

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Thoroughly read JumpServer documentations and begin modifying it as discussed	Mostafa Jammoul	11/05/25
Research about smart contracts, chain code, and software simulated enclaves, and begin implementing them	Rami Eid	11/05/25
Look into blockchain and IPFS implementations and begin testing them	Jad Awad	11/05/25
Look into blockchain and IPFS implementations and begin testing them	Omar Kaaki	11/05/25

Figure C.5: Weekly Meeting Minutes – Week 7



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
10	11/23/20	4:30 pm	2 hrs	[✓ Advisor] Students	Mostafa Jammoul

Attendees:

Omar Kaaki- Rami Eid- Mostafa Jammoul -Jad Awad

Briefly summarize the main discussions during the meeting:

We discussed the structure and format of the Final Year Project report. We reviewed the required sections, the flow of the technical chapters, and how to clearly separate the design, methodology, and implementation parts. We also went over the parts that still require revisions, including diagrams, system architecture, and clarifying certain descriptions based on dr.Bakri's feedback.
In addition, we discussed the current technical issues in the implementation, specifically the problems preventing the full system from running end-to-end.

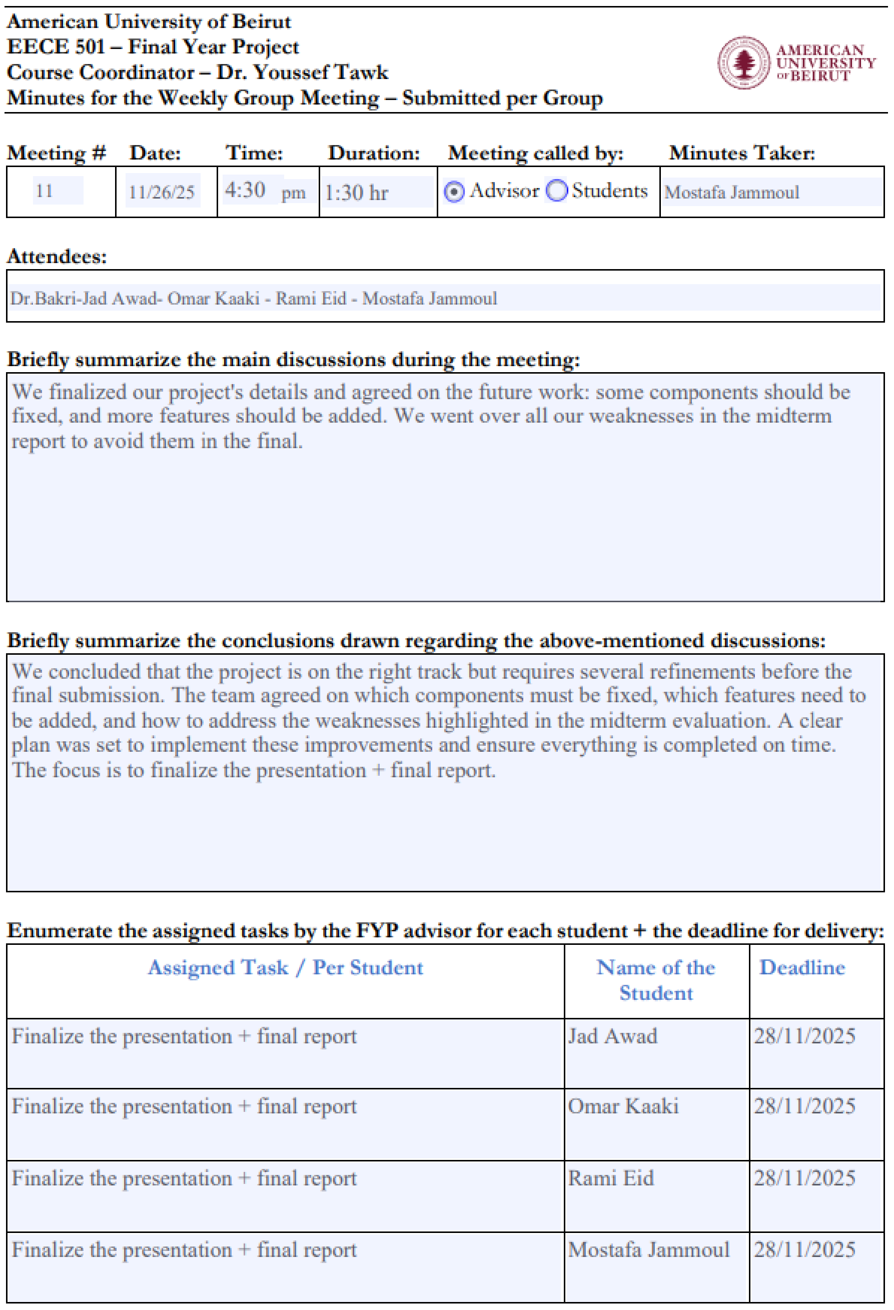
Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

We agreed on the final format of the written report and how the content should be organized to meet the FYP requirements. It was concluded that some implementation components will be postponed to the Spring semester because not everything executed correctly during testing, and additional debugging time is needed.

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Try to finish the prototype for a demonstration and begin writing the report	Omar Kaaki	11/22/2025
Try to finish the prototype for a demonstration and begin writing the report	Jad Awad	11/22/2025
Try to finish the prototype for a demonstration and begin writing the report	Rami Eid	11/22/2025
Try to finish the prototype for a demonstration and begin writing the report	Mostafa Jammoul	11/22/2025

Figure C.6: Weekly Meeting Minutes – Week 10



American University of Beirut
EECE 501 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

AMERICAN UNIVERSITY OF BEIRUT

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
11	11/26/25	4:30 pm	1:30 hr	[✓ Advisor] Students	Mostafa Jammoul

Attendees:

Dr.Bakri-Jad Awad- Omar Kaaki - Rami Eid - Mostafa Jammoul

Briefly summarize the main discussions during the meeting:

We finalized our project's details and agreed on the future work: some components should be fixed, and more features should be added. We went over all our weaknesses in the midterm report to avoid them in the final.

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

We concluded that the project is on the right track but requires several refinements before the final submission. The team agreed on which components must be fixed, which features need to be added, and how to address the weaknesses highlighted in the midterm evaluation. A clear plan was set to implement these improvements and ensure everything is completed on time. The focus is to finalize the presentation + final report.

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
Finalize the presentation + final report	Jad Awad	28/11/2025
Finalize the presentation + final report	Omar Kaaki	28/11/2025
Finalize the presentation + final report	Rami Eid	28/11/2025
Finalize the presentation + final report	Mostafa Jammoul	28/11/2025

Figure C.7: Weekly Meeting Minutes – Week 11

Bibliography

[1] R. Stoykova, “Digital evidence: Unaddressed threats to fairness and the presumption of innocence,” *Computer Law & Security Review*, vol. 42, 2021. [Online]. Available: https://doi.org/10.1016/j.clsr.2021.105575

[2] FBI Internet Crime Complaint Center (IC3), “2024 internet crime report,” 2025. [Online]. Available: https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf

[3] B. Krebs, “DEA portal breach exposes sensitive law enforcement data,” 2022, krebs on Security. [Online]. Available: https://krebsonsecurity.com/2022/05/dea-investigating-breach-of-law-enforcement-data-portal/

[4] J. Benet, “IPFS — content addressed, versioned, P2P file system,” 2014. [Online]. Available: https://arxiv.org/abs/1407.3561

[5] MarketsandMarkets. (2025) Digital forensics market global forecast to 2030. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/digital-forensics-market-230663168.html

[6] A. H. Lone and R. N. Mir, “Forensic-chain: Blockchain based digital forensics chain of custody with poc in hyperledger composer,” *Digital Investigation*, vol. 28, pp. 44–55, March 2019.

[7] E. Batista. (2020, August) Why blockchain technology can be useful for the processing of digital evidence? LOCARD Project Consortium. Summary of LOCARD D4.3 Deliverable: State of the Art report on blockchain technologies. EU H2020 Project Grant Agreement No. 832735. [Online]. Available: https://apwg.eu/why-blockchain-technology-can-be-useful-for-the-processing-of-digital-evidence/

[8] Champlain College Online, “The evolution of digital forensics,” 2024. [Online]. Available: https://online.champlain.edu/blog/evolution-digital-forensics

[9] Evidence Management, "The crucial role of chain of custody: Ensuring evidence integrity and quality assurance," 2024. [Online]. Available: https://evidencemanagement.com/the-crucial-role-of-chain-of-custody-ensuring-evidence-integrity-and-quality-assurance/

[10] N. Sharma and R. Kumar, "Digital evidence chain of custody: Navigating new realities of digital forensics," 2024. [Online]. Available: https://www.researchgate.net/publication/386361522_Digital_Evidence_Chain_of_Custody_Navigating_New_Realities_of_Digital_Forensics

[11] J. Frankenfield, "Blockchain: What it is, how it works, why it matters," 2024. [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp

[12] H. Hasanova, U.-J. Baek, M.-G. Shin, K. Cho, and M.-S. Kim, "Blockchain applications in finance," *Journal of Risk and Financial Management*, vol. 16, no. 8, p. 360, 2023. [Online]. Available: https://www.mdpi.com/1911-8074/16/8/360

[13] S. Bonomi, M. Casini, and C. Ciccotelli, "B-CoC: A blockchain-based chain of custody for evidences management in digital forensics," 2018, arXiv preprint arXiv:1807.10359. [Online]. Available: https://arxiv.org/abs/1807.10359

[14] A. H. Lone and R. N. Mir, "Forensic-chain: Ethereum blockchain based digital forensics chain of custody," *Scientific Practical Cyber Security Journal*, vol. 1, no. 2, pp. 21–27, 2017. [Online]. Available: https://www.researchgate.net/publication/321746762_Forensic-chain_Ethereum_blockchain_based_digital_forensics_chain_of_custody

[15] ——, "Forensic-chain: Blockchain based digital forensics chain of custody with poc in hyperledger composer," *Digital Investigation*, vol. 28, pp. 44–55, 2019.

[16] R. Sharma, "Hyperledger composer: What it was, how it worked," 2024. [Online]. Available: https://www.investopedia.com/terms/h/hyperledger-composer.asp

[17] K. Nischitha, M. Krishna, P. V. Pavani, and M. Bhavana, "Blockchain based cyber forensics chain of custody management with hyperledger fabric," in *International Conference on Advances in Computing and Communication*, 2023. [Online]. Available: https://www.academia.edu/118905756/Blockchain_based_Cyber_Forensics_Chain_of_Custody_Management_with_Hyperledger_Fabric

[18] D. Kim, S.-Y. Ihm, and Y. Son, "Two-level blockchain system for digital crime evidence management," *Sensors*, vol. 21, no. 9, p. 3051, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/9/3051

[19] H. Al Breiki, L. Al Qassem, K. Salah, M. H. U. Rehman, and D. Svetinovic, "Decentralized access control for IoT data using blockchain and trusted oracles," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom)*, 2020. [Online]. Available: https://www.semanticscholar.org/paper/Decentralized-Access-Control-for-IoT-Data-Using-and-Breiki-Qassem/e23e4d5977761a44cd9a0e88a29100d5304c5813

[20] L. A. Meyer, S. Romero, G. Bertoli, T. Burt, A. Weinert, and J. Lavista Ferres, "How effective is multifactor authentication at deterring cyberattacks?" *arXiv preprint arXiv:2305.00945*, 2023, preprint. [Online]. Available: https://arxiv.org/abs/2305.00945

[21] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," 2018, arXiv preprint arXiv:1805.08541. [Online]. Available: https://arxiv.org/abs/1805.08541

[22] L. Müller, D. Kühne, S. Lins, and A. Sunyaev, "TZ4Fabric: Executing smart contracts with ARM TrustZone," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–8. [Online]. Available: https://arxiv.org/abs/2008.11601

[23] Protocol Labs, "What is IPFS?" 2024. [Online]. Available: https://docs.ipfs.tech/concepts/what-is-ipfs/

[24] E. Nyaletey, R. M. Parizi, Q. Zhang, and K.-K. R. Choo, "BlockIPFS - blockchain-enabled interplanetary file system for forensic and trusted data traceability," in *IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 18–25. [Online]. Available: https://www.researchgate.net/publication/334454601_BlockIPFS_-_Blockchain-Enabled_Interplanetary_File_System_for_Forensic_and_Trusted_Data_Traceability

[25] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, M. Manevich, P. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15. [Online]. Available: https://dl.acm.org/doi/10.1145/3190508.3190538

[26] Digital Innocence Initiative, "Digital evidence in criminal proceedings: Access and equity challenges," 2024.

[27] Oracle Corporation, "Bastion hosts: Protected access for virtual cloud networks," Oracle, Tech. Rep., 2018. [Online]. Available: https://www.oracle.com/a/ocom/docs/bastion-hosts.pdf

[28] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm (Raft)," in *USENIX Annual Technical Conference*, 2014.

[29] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing Hyperledger Fabric blockchain platform," in *IEEE International Conference on Blockchain*, 2018.

[30] A. Lohachab *et al.*, "Blockchain-based systems for resource management," *IEEE Access*, 2021.

[31] Protocol Labs, "InterPlanetary File System official documentation," IPFS Docs, 2025.

[32] ——, "IPFS cluster guide for distributed storage," IPFS Cluster Docs, 2025.

[33] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, others, and J. Yellick, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proceedings of EuroSys*, 2018.

[34] Hyperledger Foundation, "Hyperledger Fabric documentation v2.5," Linux Foundation, 2025.

[35] J. Benet, "IPFS—content addressed, versioned, P2P file system," arXiv:1407.3561, 2014.

[36] NiCE Public Safety, "Digital evidence retention policies," n.d.

[37] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second," in *IEEE International Conference on Blockchain*, 2019.

[38] Amazon Web Services (AWS), "Amazon EC2 on-demand pricing," 2025.

[39] Microsoft, "Azure virtual machines pricing," 2025.

[40] Amazon Web Services (AWS), "AWS Nitro Enclaves documentation," 2025.

[41] M. Taassori *et al.*, "VAULT: Reducing paging overheads in SGX with efficient integrity verification structures," in *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018.

[42] Amazon Web Services (AWS), “AWS global infrastructure,” 2025.

[43] National Institute of Standards and Technology (NIST), “NIST SP 800-34: Contingency planning guide for information systems,” U.S. Department of Commerce, Tech. Rep., 2010.

[44] SWGDE, “Best practices for evidence collection,” 2023.

[45] ——, “Best practices for digital evidence acquisition,” 2023.

[46] S. E. Goodison, R. C. Davis, and B. A. Jackson, “Digital evidence and the U.S. criminal justice system: Identifying technology and other needs to more effectively acquire and utilize digital evidence,” RAND Corporation, Tech. Rep., 2015.

[47] Amazon Web Services (AWS), “AWS Well-Architected sustainability pillar,” 2025.

[48] International Organization for Standardization (ISO), “ISO 23257:2022 – Blockchain and distributed ledger technologies – Reference architecture,” Tech. Rep., 2022.

[49] IEEE, “IEEE 2418.2-2020 – Standard for blockchain data format,” Tech. Rep., 2020.

[50] National Institute of Standards and Technology (NIST), “FIPS 180-4: Secure Hash Standard,” U.S. Department of Commerce, Tech. Rep., 2015.

[51] ——, “FIPS 186-5: Digital Signature Standard,” U.S. Department of Commerce, Tech. Rep., 2023.

[52] Internet Engineering Task Force (IETF), “RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3,” Tech. Rep., 2018.

[53] National Institute of Standards and Technology (NIST), “NIST SP 800-207: Zero trust architecture,” U.S. Department of Commerce, Tech. Rep., 2020.

[54] International Organization for Standardization (ISO/IEC), “ISO/IEC 27041:2015 and ISO/IEC 27042:2015 – Forensic analysis guidelines,” Tech. Rep., 2015.

[55] ——, “ISO/IEC 27037:2012 – Guidelines for identification, collection, acquisition and preservation of digital evidence,” Tech. Rep., 2012.

[56] IEEE, “IEEE P2418.6 – Blockchain cybersecurity framework (draft),” 2024.