ForLens – A Lightweight AI-Driven Endpoint Security Solution for Real-Time Threat Detection in SMEs

Project ID: 25_26J_076

Project Proposal Report

Author: Shanka De Silva
Student ID: IT22887580

Supervisor: Mr. Kanishka Yapa
Co-Supervisor: Mr. Harinda Fernando

B.Sc. (Hons) in Information Technology
Specializing in Cyber Security

Faculty of Computing
Sri Lanka Institute of Information Technology
Sri Lanka

August 2025

DECLARATION

Declaration by the Candidate

We declare that this is our own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name: De Silva H S

Signature: Shanika .

Date: 28/08/2025

Declaration by the Supervisor

Name: Mr. Kanishka Yapa

Signature: KPY

Date: 28/08/2025

Declaration by the Co-Supervisor

Name: Mr. Harinda Fernando

Signature:

Date: 28/8/25

Abstract

The foundation of security monitoring, digital forensics and regulatory compliance is system logs. Nevertheless, they are the best targets of being tampered and deleted by attackers, particularly insiders who have high privileges and thus make them useless as evidence. The SMEs, which the larger ForLens project targets, are especially susceptible because of resource limitations that do not allow implementing enterprise-level, tamper-proof logging systems. More traditional techniques such as hash chaining offer simple integrity but have linear verification cost (O(n)) and are inapplicable to large-scale log analysis. Although forward-secure logging and blockchain-based schemes provide better security, they usually impose a serious performance penalty, complexity, and cost, which is unacceptable to the SME market. The study suggests a lightweight, cryptographically secured logging system specifically created to work in SME settings as a fundamental part of ForLens. The solution combines Merkle tree chaining to verify efficiently and scalably (O(log n)) and forward-secure key evolution to ensure that past logs cannot be compromised by future key compromises and crash-resilient protocols to ensure that tampering cannot be hidden by system resets. In addition, it supports selective encryption to provide confidentiality to sensitive log data, which meets the requirements of GDPR and HIPAA compliance. Security analysis (resistance to tampering, insider threats, crash-attacks) and performance benchmarking (latency, throughput, storage overhead) will be strictly checked against the state-of-the-art approaches. The desired result is a secure, efficient and cost-effective logging infrastructure that will offer enterprise-level security and evidential integrity to SMEs, enabling them to have reliable forensic functionality that was previously unavailable to them because of cost and complexity.

Keywords: Secure Logging, Cryptography, Merkle Tree, Forward Security, SME Cybersecurity, Tamper-Evidence

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABE-Attribute-Based Encryption

CAGR -Compound Annual Growth Rate

EDR - Endpoint Detection and Response

FssAgg -Forward-secure Sequential Aggregate Signatures

GDPR-General Data Protection Regulation

HIPAA -Health Insurance Portability and Accountability Act

HSM-Hardware Security Module

MSP -Manage Service Provider

O(log n)-Logarithmic Time Complexity

O(n)-Linear Time Complexity

PII -Personally Identifiable Information

SIEM - Security Information and Event Management

1. INTRODUCTION

1.1 Background and Literature Survey

System logs are essential in the digital era to ensure that there is visibility of the IT infrastructure, identify security breaches, forensic analysis, and demonstrate adherence to the regulations such as GDPR, HIPAA, and PCI-DSS [1]. They give a permanent record of system events, user actions and application states. But it is this immutability that is an illusion when the logs themselves are not secured against interference. When attackers compromise a system, they often start with logs as their initial step in order to leave no traces of their intrusion, conceal their footprints, and continue to have unlimited access [2]. The insider attacks also contribute to this threat because privileged users can easily alter logs with minimal effort [3]. The scholarly and commercial reaction to this issue has changed through the decades. Early background research by Schneier and Kelsey [4] put forward the idea of secure audit logs, proposing schemes in which the entries in the log are connected by cryptographic hash functions in a chain. Every entry contains the hash of the last one, and any change is noticed. Although conceptually sound, this hash chaining scheme has a fatal flaw, namely, to verify it, one must process the whole chain up to the point of interest, which is O(n) time-consuming. This is computationally infeasible to the large amounts of logs produced in contemporary systems, making it infeasible with real-time auditing in SME contexts [5]. In a bid to deal with major management weaknesses, the concept of forward security was introduced. This was first introduced by Bellare and Yee [6], who constructed signature schemes in which cryptographic keys change with time. When a key is compromised, then it cannot be used to generate signatures in the past. This was furthered by Ma and Tsudik [7] with Forward-secure Sequential Aggregate (FssAgg) signatures, which permit compact verification of log sequences. Nevertheless, such systems are susceptible to crash-attacks, which is a more advanced form of intrusion, in which an attacker will cause a system crash or power outage by modifying logs, then use the recovery mechanism to conceal his/her changes [8]. The emergence of blockchain technology motivated the use of the technology to ensure the security of logging. Some projects such as BlockAudit [9] use the immutable nature of distributed ledgers to store log hashes. Although it is useful in tamper-resistance, blockchain presents a large amount of latency, storage overhead and energy consumption, which is frequently inappropriate in resource-constrained and cost-sensitive SMEs [10]. Moreover, storing the hashes in a blockchain does not necessarily

assure confidentiality of the contents of the logs themselves. One of the most powerful data structures that are not used to their full potential in this field is the Merkle tree (or hash tree) [11]. Applied widely in blockchain systems such as Bitcoin to verify the integrity of a single piece of data (a log entry) in O(log n) time, a Merkle tree enables a verifier to check the integrity of a single piece of data (a log entry) in O(log n) time by producing a small cryptographic verification (a Merkle proof) against an authenticated root hash. This provides a scalability benefit that is monumental compared to linear hash chains. Although efficient, the use of Merkle trees to create an endpoint-wide, forward-secure and crash-resilient logging system is a relatively new field of research. At the same time, the issue of log privacy has become prominent. Logs usually hold sensitive PII (Personally Identifiable Information) [12]. Laws such as GDPR require that this information should be secured. The majority of secure logging studies focus on integrity and not confidentiality, which introduces a compliance gap. Such techniques as Attribute-Based Encryption (ABE) [13] may permit fine-grained, role-based access to encrypted log data, enabling auditors to establish integrity without viewing sensitive material, but this introduces an additional complexity to the design. The wider ForLens initiative is to offer a lightweight, AI-based security product to SMEs. In any security solution, the capability to trust its own operational logs is the most important. When an attacker is capable of disabling or altering the logs of the ForLens agent itself, then all detection and response capabilities are compromised. Thus, a cryptographically secured logging system is not a side feature but a basic trust anchor of the whole ForLens ecosystem.

1.2 Research Gap

The literature reveals a fragmented landscape where solutions optimize for one aspect of secure logging at the expense of others. There is no unified framework that delivers the efficiency required by SMEs, the robust security against modern attacks, and the privacy features needed for compliance, all within a lightweight and cost-effective architecture.

The table below synthesizes the critical gaps by comparing existing approaches against the desired features for an SME-ready solution:

Table 1.1: Comparative Analysis of Secure Logging Techniques

Feature	Traditional Logging	Hash Chain	Forward-Secure	Blockchain-Based	Merkle Tree Only	Proposed System
Tamper Evidence	✗	✓	✓	✓	✓	✓✓(Multi-Layer)
Crash-Attack Resilience	✗	✗	Partial	Partial	✗	✓ (Integrated)
Forward Security	✗	✗	✓	Partial	✗	✓ (Strong)
Privacy / Selective Access	✗	✗	✗	✗	✗	✓
Verification Complexity	-	High	High	Very High	Low	Low
Storage Overhead	Low	Medium	Medium	Very High	Low	Low-Medium

SME Cost-Effectiveness	High	High	Medium	Very Low	High	High
Cloud/IoT Readiness	✓	Limited	Limited	✗	✓	✓✓(Optimized)

The specific research gaps identified are:

1. Absence of Crash-Resilient Merkle Trees

Although Merkle trees provide efficient verification, their implementation does not provide mechanisms to counter crash-consistency attacks, which Blass and Noubir have noted [8].

2. Performance-Security Trade-off

Current solutions present a trade-off between performance (hash chains) and strong security (blockchain).Blockchain is immutable but very expensive to compute and store. Linear-time verification is needed in hash chains, and this is not practical in large-scale or real-time log analysis. There is a lack of a hybrid model that provides high security and logarithmic verification scaling.

3. Failure to protect the Log Confidentiality

The majority of systems are integrity-secure but fail to protect confidentiality, making sensitive log information available to unauthorized users. Mainstream logging libraries lack a holistic framework capable of offering tamper-evidence as well as configurable confidentiality to sensitive log fields.

4. Hybrid Framework Absence -No current system combines forward security, Merkle tree chaining, crash-resilience, and selective disclosure into one unified architecture

1.3 Research Problem

Primary Research Question

The main research problem is that there is no lightweight, cryptographically secured logging system that offers efficient, scalable, and verifiable integrity of security logs and at the same time offers confidentiality and resistance to advanced attacks such as crash-consistency exploits at the resource and cost constraints of Small and Medium Enterprises.

Specific Problem Statement

The particular issue that this research aspect of the ForLens project will tackle is as follows.

Existing secure logging systems are not suitable to deploy in SMEs. Hash chains are expensive to check at scale. Blockchain logging is too heavy. Current Merkle tree designs are not forward secure and crash resilient. Moreover, none of these solutions is sufficient in the privacy of the log contents by selective encryption. This poses a serious disconnection point in the chain of trust of SME security solutions such as ForLens that need a lightweight, tamper-proof, and auditable record of the identified threats and system actions without impacting on performance or cost.

The proposed research aims to fill this gap by developing and deploying a new logging architecture that is synergistically designed based on efficiency using Merkle trees, long-term integrity using forward-secure cryptography, robustness using crash-resilient protocols, and privacy using selective encryption.

2. OBJECTIVES

2.1 Main Objective

To design, implement, and evaluate a lightweight, cryptographically secured logging system that integrates Merkle tree chaining, forward-secure cryptography, crash-resilient storage, and selective encryption to provide tamper-evident, scalable, and privacy-preserving log management for the ForLens endpoint security platform in SME environments.

2.2 Specific Objectives

The main objective will be achieved through the following specific, measurable, achievable, relevant, and time-bound (SMART) objectives:

Table 2.1: Specific Objectives and Measurable Outcomes

Objective	Description	Measurable Outcome
SO1	To design a hybrid cryptographic logging architecture.	A complete system design document detailing the integration of Merkle trees, forward-secure key evolution (e.g., based on [6][7]), a crash-resilient protocol (e.g., inspired by SLiC [8]), and a model for selective encryption (e.g., using ABE concepts [13]).
SO2	To develop a functional prototype of the system.	A working software prototype implemented in Python/Go, capable of ingesting log data, generating Merkle proofs, evolving keys, and allowing for privacy-preserving audits. The prototype will be hosted on a GitHub repository.
SO3	To conduct a comprehensive performance evaluation.	Benchmarking results showing: • Log insertion latency <

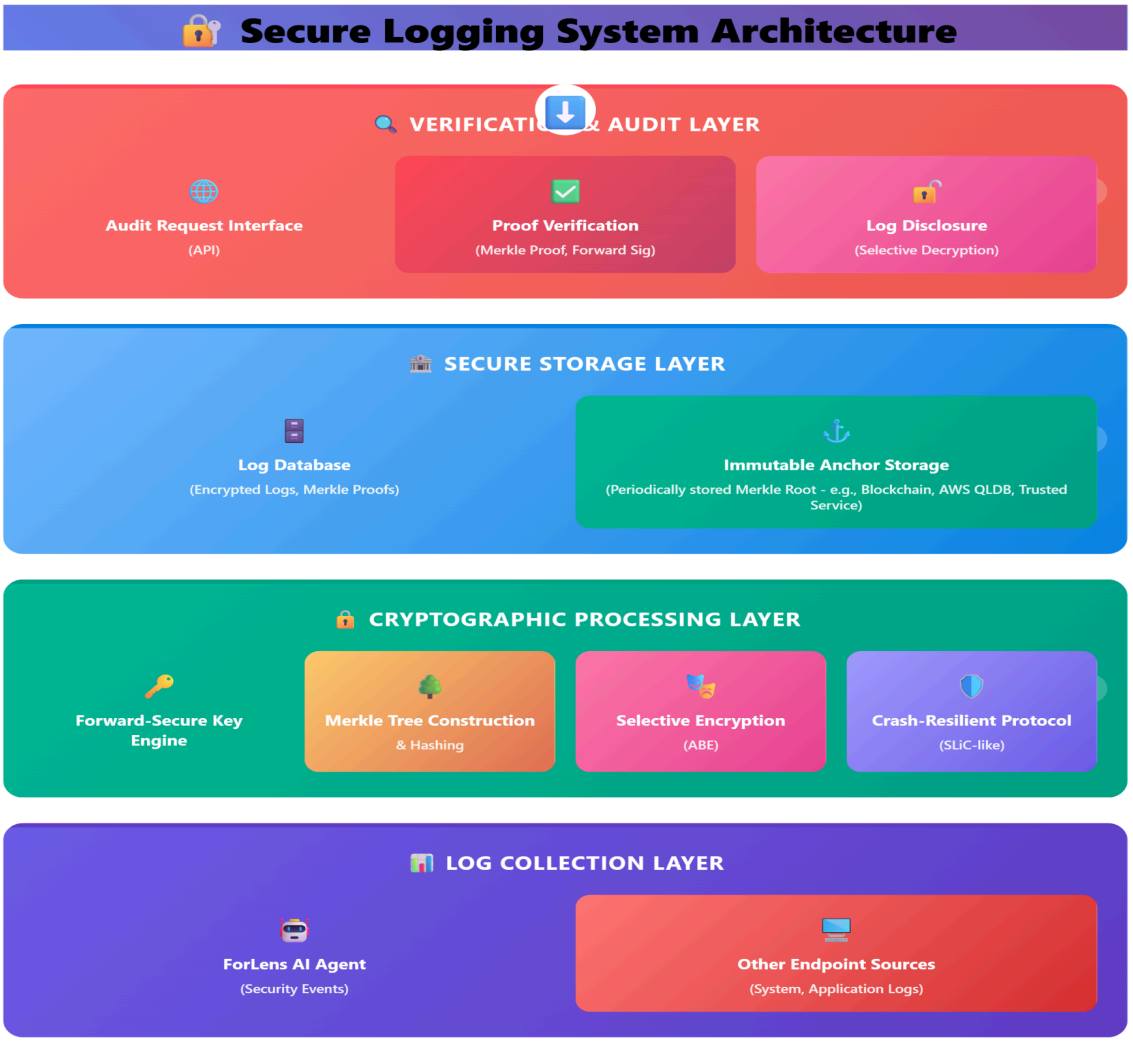
		5ms per entry. • Verification of a single entry in **< 10ms** for a log set of 1 million entries. • Storage overhead not exceeding **15%** compared to plain text logs.
SO4	To perform a rigorous security analysis.	A security assessment report demonstrating resistance to: • **Tampering:** 100% detection rate of modified/deleted entries. • **Crash-attacks:** Successful recovery and detection of tampering after simulated crashes. • **Insider Threats:** Inability to forge past logs with compromised current-era keys.
SO5	To execute a comparative analysis against existing methods.	A published paper or detailed report containing tables and graphs (e.g., Figure 3.2) quantitatively comparing the proposed system's performance and security against hash chaining, forward-secure logging, and a baseline blockchain-based method.
SO6	To demonstrate integration and applicability within the ForLens framework.	A successfully integrated module where ForLens AI alerts are seamlessly ingested and secured by the logging system. A case study document outlining its operation in a simulated SME cloud/IoT scenario.

3. METHODOLOGY

3.1 Technical Architecture Design

The proposed system is structured into four decoupled layers to ensure modularity, scalability, and maintainability. The high-level architecture is depicted in Figure 3.1.

Figure 3.1: High-Level Architecture of the Cryptographically Secured Logging System



Layer 1: Log Collection Layer

This layer collects raw log events of various origins. The ForLens AI Agent streaming real-time security alerts (e.g., process anomalies, network connections) will be used as the main source. Moreover, the endpoint standard system and application logs will be gathered.

This layer will standardize the data to a standardized form of JSON where timestamps and event types are consistent.

Layer 2: Cryptographic Processing Layer

This is the main driving force of the system.

1. Forward-Secure Key Engine

This module will handle a key which advances after a specified time (e.g. after every hour or after every 1000 entries). It will apply a powerful algorithm (e.g. a hash-based key derivation function) to create a new signing key out of the old one, making the old keys useless. This makes sure that an attacker cannot forge logs of earlier epochs with a compromise of the current key.

2.Merkle Tree Construction

Every normalized log record will be hashed (SHA-256). Leaves of a Merkle tree will be composed of these hashes. The tree will be constructed in batches (i.e. after every 1000 entries or after every minute). The base of every tree will be signed using the present forward-secure key.

3.Selective Encryption

Log entries will be scanned to find sensitive fields (e.g., user IDs, email addresses, command arguments based on a pattern match) before being stored. A lightweight encryption scheme will be used to encrypt these fields. The design will consider the use of Attribute-Based Encryption (ABE) to enable issuance of decryption keys to auditors who have certain attributes (e.g., "ForensicTeam").

4. Crash-Resilient Protocol

The system will adopt a protocol that is based on SLiC [8] to reduce crash-attacks. This is the strategic persistence of state information (e.g. the most recent Merkle root and key epoch) in a crash-resistant way. On reboot, the system will check the integrity of the log store with this persisted state and identify any form of tampering that took place immediately before the crash.

Layer 3: Secure Storage Layer

The processed logs (encrypted where needed), their Merkle proofs, and the signatures are stored in a database (e.g. SQLite because it is lightweight or PostgreSQL because it is scalable). Importantly, the signed Merkle root hashes are occasionally (e.g. daily) anchored to an immutable external medium. This may be a cheap blockchain (e.g. a private Ethereum chain, IOTA), a hosted service such as Amazon QLDB, or even a basic public timestamping authority to be used by cost-effective SMEs. This is an external anchor that gives a point of reference that can be trusted in all future verifications.

Layer 4: Verification & Audit Layer

This layer offers an API that auditors and system administrators can use to check the integrity of any entry in the logs. To check an entry, the auditor asks its Merkle proof of the database. With this evidence and the known Merkle root (fetched out of the immutable anchor storage), they can themselves check the integrity of the entry in O(log n) time. In case sensitive data was encrypted, the auditor may demand decryption (assuming that they have the appropriate ABE key attributes) once integrity is established.

3.2 Implementation Framework

The prototype will be built based on a stack that is selected because it is robust, popular in security applications and applicable to SMEs.

Programming Languages - Python 3.x will be used as the main language to rapidly prototype the cryptographic logic, Merkle tree library and API server because it has a rich ecosystem. Go (Golang) is a contender to create a more performant, deployable agent since it has a strong concurrency support and a small footprint.

Cryptographic Libraries- PyCryptodome (Python) or the crypto package of Go will include standard cryptographic primitives (SHA-256, AES, RSA). In the case of more sophisticated schemes, such as ABE, the Charm-Crypto model or a more basic symmetric-key implementation will be considered on the basis of performance.

Database- SQLite for initial lightweight testing and deployment on a single endpoint. The testing of centralised log collection among various endpoints in a simulated SME network will be done using PostgreSQL.

External Anchoring- In the case of prototyping, a basic file-based system or a local private Ethereum/Pantheon blockchain will be employed. Cloud anchoring through the free tier of AWS QLDB will be investigated in order to demonstrate it more realistically.

Methodology Development- It will use an Agile approach with two-week sprints. Git on GitHub will be used as a version control tool, which will guarantee collaboration and code history.

3.3 Trust and Integrity Assurance Framework

The trust is built in a multi-layered cryptographic manner.

Merkle Tree to Efficient Verification: Allows to verify any single log entry against a trusted root hash in O(log n) complexity which is a fundamental improvement over O(n) chains.

Forward-Secure Signatures on Roots: Signing of each Merkle root with a forward-secure key ensures that even in the event of a theft of the current key, all previously signed roots (and the logs they signify) are unforgeable.

Anchoring of roots: This is to ensure that the root is published to an immutable storage (e.g. blockchain) so that an attacker cannot just re-publish the entire database with a forged one since the authentic root can be verified publicly elsewhere.

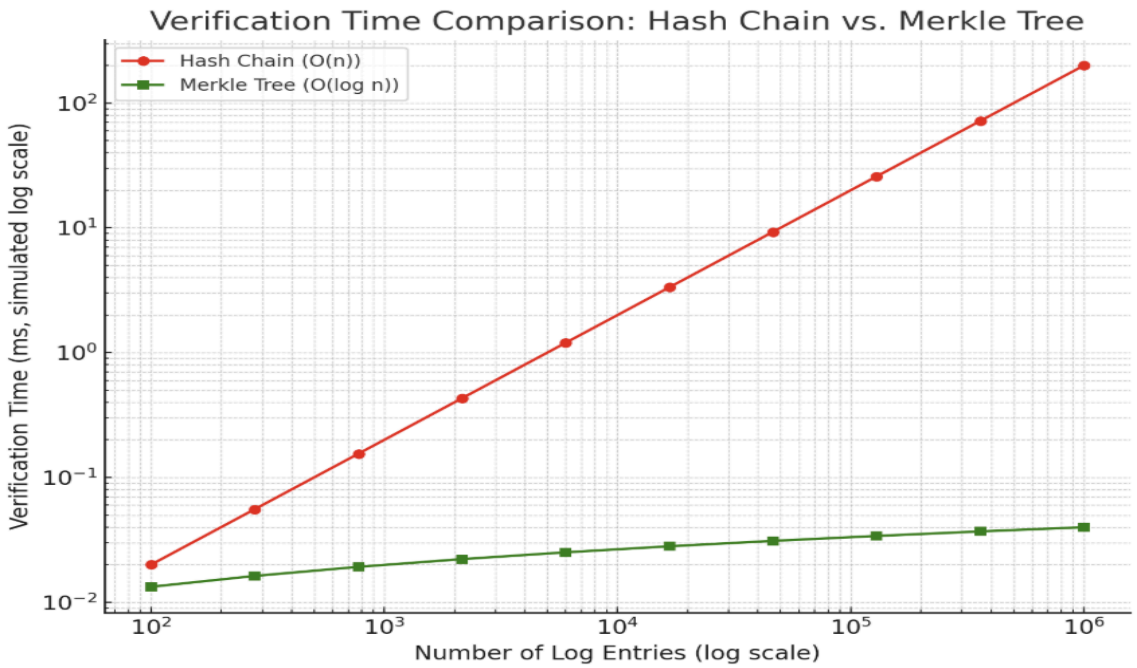
Crash-Consistency: The SLiC-inspired protocol guarantees that the system is capable of detecting tampering that takes advantage of system crashes, sealing a major vulnerability in most logging systems.

Selective Confidentiality: The encryption of sensitive fields will guarantee that integrity checks can be conducted by a broader audience without data privacy, as required by regulations.The following graph illustrates the performance advantage of the chosen Merkle tree approach, which is central to the SME-friendly design.

The following graph illustrates the performance advantage of the chosen Merkle tree approach, which is central to the SME-friendly design.

Figure 3.2: Simulated Verification Time: Hash Chain vs. Merkle Tree

A line graph showing verification time (ms, log scale) on the Y-axis against the number of log entries (log scale) on the X-axis. The red line for "Hash Chain (O(n))" shows a steep, linear increase. The green line for "Merkle Tree (O(log n))" shows a much flatter, logarithmic increase, remaining low even at 106 entries.



3.4 Security Evaluation Methodology

The security of the system will be evaluated by testing its resilience against a comprehensive set of attack vectors. The evaluation will involve both theoretical analysis and practical penetration testing.

Table 3.4: Security Attack Vectors and Evaluation Metrics

Attack Vector	Description	Evaluation Method	Success Metric
Data Tampering	Attacker modifies or deletes log entries in the database.	Inject modified logs directly into the DB.	System must detect inconsistency during verification and reject the tampered entries
Replay Attack	Attacker re-inserts old, valid log entries.	Capture and re-inject old log data.	System must detect duplicate or out-of-order entries using timestamps and sequence numbers.
Key Compromise	Attacker gains access to the current forward-secure key.	Simulate theft of the current key.	Attacker must be unable to forge signatures for past log epochs (Forward Security property).
Crash-Attack	Attacker tampers with logs and forces a system crash.	Use a kernel module or hard power cycle to crash the system after tampering.	System must recover and detect the log tampering during the integrity check on reboot.
Insider Threat	A privileged user (e.g., admin) attempts to cover their tracks.	Simulate an admin using their access to alter logs and system state.	The external anchoring of Merkle roots must make this detectable by an external auditor.

Tools such as Metasploit, custom Python scripts, and manual code injection will be used to simulate these attacks. The results will be documented in a security assessment report, highlighting any vulnerabilities found and the mitigation strategies employed.

3.5 Performance Evaluation Framework

Performance is critical for SME adoption. The system will be benchmarked under a simulated load representative of an SME environment (10-100 endpoints generating ~10,000 events per hour).

- Metrics:
    - **Log Insertion Latency(ms):** The time taken to process a log event (encrypt, hash, insert into tree, update DB).
    - **Verification Time(ms):** The time taken to generate and verify a Merkle proof for a single random log entry.
    - **Throughput(logs/sec):** The maximum number of log events the system can process per second.
    - **Storage Overhead(MB/GB):** The additional storage required for cryptographic metadata (hashes, proofs, signatures) compared to plain text logs.
- Method: Tests will be conducted on a standard SME-grade machine (CPU: Intel i5, RAM: 8GB, SSD). A log generation tool will simulate event streams of varying sizes (from 10^3 to 10^6 events). Results will be compared against a baseline hash chain implementation and a simple blockchain-based logging prototype (e.g., storing hashes on Ethereum).

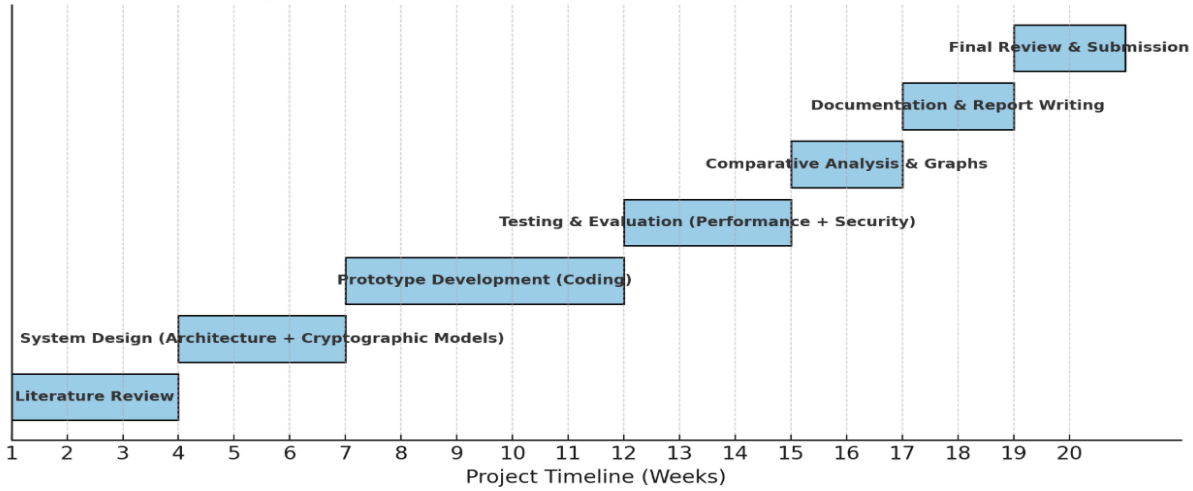
3.6 Work Breakdown Structure

The project will be executed over a 20-week period, following the Work Breakdown Structure (WBS) and timeline outlined in the Gantt chart below. The project is broken down into seven primary phases.

Figure 3.6: Work Breakdown Structure (WBS) with Timeline

(A Gantt chart showing the following tasks and their timelines over 20 weeks:)

Figure 3.3 – Work Breakdown Structure (WBS) with Timeline



- Literature Review (Weeks 1-3)
- System Design (Weeks 4-6)
- Prototype Development (Core Crypto Modules) (Weeks 7-9)
- Prototype Development (Storage & API Integration) (Weeks 10-11)
- Testing & Evaluation (Performance) (Weeks 12-13)
- Testing & Evaluation (Security) (Week 14)
- Comparative Analysis & Graph Generation *(*Week 15)
- Documentation & Report Writing (Weeks 16-18)
- Final Review, Integration with ForLens & Submission (Weeks 19-20)

4. DESCRIPTION OF PERSONNEL AND FACILITIES

Project Personnel

The project will be conducted by De Silva H S, a final-year undergraduate student specializing in Cyber Security at SLIIT. The student brings relevant coursework knowledge in network security, cryptography, and software engineering, along with practical experience in Java and Python programming from previous academic projects.

Supervision and Guidance

Mr. Kanishka Yapa will provide primary supervision, bringing expertise in Cryptography. Regular weekly meetings will ensure project progress alignment with objectives and technical guidance for complex implementation challenges. The supervisor will also facilitate access to research resources and industry connections for validation purposes.

Technical Facilities

The project will be executed utilizing the following software, hardware, and academic resources:

- Software Resources:
    - Development Tools: JetBrains PyCharm (Python IDE), Visual Studio Code, Git version control for source code management, and Docker for containerized deployment and testing.
    - Cryptographic Libraries: The PyCryptodome library for Python will be used for implementing core cryptographic operations (SHA-256, AES). For potential advanced features, the Charm-Crypto framework will be explored for Attribute-Based Encryption (ABE).
    - Databases: SQLite for initial lightweight prototyping and PostgreSQL for testing centralized log collection in a multi-endpoint SME simulation.
    - Simulation & Testing: Custom Python scripts will be developed for log generation and load testing. Penetration testing tools like Metasploit will be used for security evaluation in a controlled lab environment.
- Hardware Resources:

    - Primary Development Machine: A personal laptop (Intel i7 processor, 16GB RAM, 512GB SSD) running Windows 11 with WSL2 (Windows Subsystem for Linux) to provide a robust development environment.
    - Testing Lab: Access to the university's computing labs will be utilized to simulate a small SME network environment using virtual machines (VMWare Workstation) to deploy multiple endpoint agents and a central logging server.

Library and Research Resources

SLIIT library provides access to IEEE Xplore, ACM Digital Library, and other academic databases essential for literature review and technical documentation. Remote access enables continuous research capability throughout the project duration.

Collaboration Infrastructure

Effective collaboration within the ForLens team is supported by a modern digital workflow:

- Code Repository: A GitHub organization has been established for the ForLens project. All code for this logging component will be maintained in a dedicated private repository, enabling version control, code review, and seamless integration with other components.
- Communication: Microsoft Teams is used for daily communication, weekly virtual meetings, and file sharing among team members and supervisors.
- Project Management: Trello boards are utilized to track tasks, manage the Work Breakdown Structure (WBS), and monitor deadlines, ensuring the entire team remains synchronized and on schedule.

This combination of skilled personnel, expert supervision, robust technical facilities, and efficient collaborative infrastructure provides a solid foundation for the successful execution of this project.

5. COMMERCIALIZATION AND BUSINESS POTENTIAL

5.1 Target Markets

The primary target market is Small and Medium-sized Enterprises (SMEs) across sectors such as:

- Healthcare SMEs : clinics, pharmacies handling patient data (GDPR/HIPAA compliance).
- Financial SMEs :small banks, fintech startups requiring auditable logs.
- Retail & E-commerce SMEs : transaction logging and fraud detection.
- Technology SMEs : software companies, managed service providers (MSPs) who must maintain compliance.

Why SMEs?

- Over 90% of global businesses are SMEs, but most cannot afford enterprise-scale cybersecurity solutions.
- Compliance regulations (GDPR, HIPAA, PCI-DSS) increasingly require secure, tamper-proof audit logs.
- The solution's lightweight nature (Merkle trees, forward security, selective logging) makes it practical for SMEs with limited resources.

Market Insight:

- The global SME cybersecurity market is projected to grow from USD 68 billion (2024) to USD 120+ billion (2030) (CAGR ~10%).
- Secure logging solutions form a key subcomponent of this market, especially in regulated industries.

5.2 Competitive Advantages

The proposed cryptographically secured logging system provides several distinct advantages over existing solutions:

Key Differentiators

1. Lightweight Design for SMEs – Unlike heavy blockchain-based logging, the system uses Merkle tree chaining + forward security, which is computationally efficient and suitable for SME-scale infrastructure.

2. Scalability with Low Verification Cost – Log verification is O(log n) with Merkle proofs, far more efficient than hash chains (O(n)).

3. Crash Resilience – Protects logs even during abnormal shutdowns (a limitation in traditional schemes).

4. Privacy-Preserving Logging – Selective encryption allows compliance with GDPR/HIPAA while still enabling verifiable audits.

5. Integration with ForLens – Provides a trusted audit trail for AI-driven endpoint detection, strengthening overall security posture.

6. Cost-Effectiveness – Built on open-source tools, making it affordable compared to enterprise SIEM or blockchain systems.

5.3 Commercialization Pathway

The pathway to commercialize the proposed cryptographically secured logging system as part of ForLens involves several strategic steps:

Phase 1 – Prototype Development & Validation (Academic Stage)

- Build and test a working prototype integrated with ForLens AI threat detection engine.
- Conduct performance and security evaluations (as defined in Section 3).
- Publish results in academic venues to validate novelty and credibility.

Phase 2 – Pilot Deployment with SMEs (Proof-of-Concept Stage)

- Collaborate with local SMEs (e.g., healthcare clinics, small financial firms) to deploy the prototype.
- Gather real-world performance data (log volume, verification speed, storage usage).
- Adjust system for usability and compliance needs (GDPR/HIPAA).

Phase 3 – Productization (Pre-Commercial Stage)

- Develop a lightweight, user-friendly deployment package (Docker/Kubernetes).
- Add GUI-based dashboard for log visualization and integrity verification.
- Provide API endpoints for integration with SME security systems.

Phase 4 – Commercial Launch (Go-to-Market Stage)

- Launch ForLens-SecureLogs as an affordable security add-on for SMEs.
- Partner with Managed Service Providers (MSPs) who serve SMEs with limited IT staff.
- Offer both on-premise and cloud-based versions, depending on SME preferences.

Phase 5 – Scaling and Expansion (Growth Stage)

- Expand to regional/global SME markets.
- Offer additional premium features (e.g., AI-driven anomaly detection of log tampering attempts).
- Integrate with regulatory compliance frameworks to position as a compliance enabler (GDPR/ISO 27001/PCI-DSS).

5.4 Market Size and Growth Potential

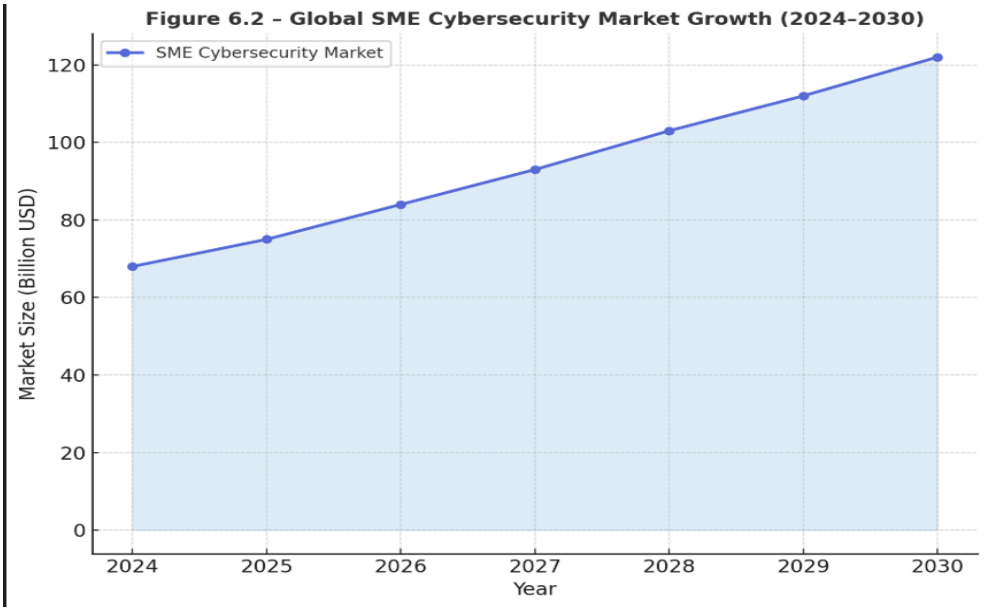


Figure 5.4 – Global SME Cybersecurity Market Growth (2024–2030)

- The SME cybersecurity market is projected to grow from $68B (2024) to $122B (2030).

- CAGR ≈ 10%, driven by compliance regulations and rising SME cyber threats.

- Secure logging systems like yours fall under compliance + auditability solutions, making it a strong niche.

5.5 Potential Revenue Streams

The proposed logging system can generate revenue through multiple business models:

1. Subscription Model (SaaS) – SMEs pay a monthly/annual fee to use a cloud-hosted version of the logging service.
    - Example: $20–$50 per endpoint per year.
2. On-Premise Licensing – One-time or annual license fee for SMEs who prefer to keep logs locally.
3. Integration Add-On for ForLens – Sold as an additional module to the ForLens endpoint security solution.
4. Managed Services (MSPs) – Partner with managed service providers who bundle the solution into their SME cybersecurity offerings.
5. Compliance-as-a-Service – Provide secure log storage with audit-ready reports for SMEs subject to GDPR, HIPAA, PCI-DSS.

5.6 Long-Term Vision

The long-term vision is to establish the logging system as a global standard for SME secure logging.

- Phase 1 (Short-Term): Deploy system as part of ForLens, validating it in SME pilot programs.
- Phase 2 (Medium-Term): Evolve into a modular secure logging platform that can integrate with other cybersecurity products (SIEMs, IDS/IPS).
- Phase 3 (Long-Term): Position as a compliance enabler, helping SMEs demonstrate regulatory adherence at low cost.
- Potential to expand beyond SMEs to IoT/IIoT systems, where lightweight, tamper-proof logging is also critical.

5.7 Market Viability Assessment

TThe market viability of the proposed solution is strong due to:

1. Rising Cybersecurity Awareness in SMEs
    - Increasing cyberattacks on SMEs (ransomware, phishing, insider threats) make tamper-proof logging essential.
2. Regulatory Push
    - Global data protection laws mandate secure record-keeping. SMEs need affordable compliance solutions.
3. Competitive Gap
    - Existing solutions (blockchain logging, SIEMs) are either too resource-heavy or too expensive for SMEs.
    - The proposed system fills the gap with a lightweight, cost-effective alternative.
4. Scalability
    - The architecture ensures performance at SME log volumes (10^3 – 10^6 logs/day) without excessive costs.

REFERENCES

[1] R. Ko et al., "TrustCloud: A framework for accountability and trust in cloud computing," in *Proc. 2011 IEEE World Congress on Services*, 2011, pp. 584-588. doi: 10.1109/SERVICES.2011.85.

[2] F. L. Greitzer and D. A. Frincke, "Combining traditional cyber security audit data with psychosocial data: Towards predictive modeling for insider threat mitigation," in *Insider Threats in Cyber Security*. Springer, 2010, pp. 85–113. doi: 10.1007/978-1-4419-7133-3_5.

[3] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," in *Insider Attack and Cyber Security*. Springer, 2008, pp. 69–90. doi: 10.1007/978-0-387-77322-3_5.

[4] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 159–176, May 1999. doi: 10.1145/317087.317089.

[5] J. Holt, "Logcrypt: Forward security and public verification for secure audit logs," in *Proc. 2006 Australasian Conf. Information Security and Privacy (ACISP)*, 2006, pp. 203-215. doi: 10.1007/11780656_18.

[6] M. Bellare and B. S. Yee, "Forward integrity for secure audit logs," University of California, San Diego, Tech. Rep., 1997. [Online]. Available: https://cseweb.ucsd.edu/~bsy/pub/forward.pdf

[7] D. Ma and G. Tsudik, "A new approach to secure logging," *ACM Trans. Storage*, vol. 5, no. 1, pp. 1–21, Mar. 2009. doi: 10.1145/1502777.1502782.

[8] E.-O. Blass and G. Noubir, "SLiC: Secure logging with crash tolerance," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Dallas, TX, USA, 2017, pp. 23–34. doi: 10.1145/3133956.3134004.

[9] I. Ahmad et al., "Towards blockchain-driven, secure and transparent audit logs," *arXiv preprint arXiv:1811.09944*, 2018. [Online]. Available: https://arxiv.org/abs/1811.09944

[10] Y. Wang, Y. Liu, H. Zhang, and X. Li, "Blockchain-based forensic logging for tamper-proof evidence management," *Future Gener. Comput. Syst.*, vol. 110, pp. 756–766, Sep. 2020. doi: 10.1016/j.future.2019.10.024.

[11] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Springer Berlin Heidelberg, 1988, pp. 369–378. doi: 10.1007/3-540-48184-2_32.

[12] J. C. Davis, et al., "An empirical study of sensitive information in open-source logs," in *Proc. 2025 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2025, (to be published). [Preprint]. Available: https://arxiv.org/abs/2401.00001

[13] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. 2007 IEEE Symposium on Security and Privacy (SP '07)*, 2007, pp. 321-334. doi: 10.1109/SP.2007.11.

[14] A. A. Yavuz, P. Ning, and M. K. Reiter, "Efficient, compromise-resilient and append-only cryptographic constructions for secure logging," in *Proc. 12th Int. Conf. Financial Cryptography Data Security (FC)*, 2012, pp. 1-20. doi: 10.1007/978-3-642-32946-3_18.

[15] K. Kuznetsov, A. Gurtov, and V. Ipatov, "Merkle trees: Analysis of collision probability," *arXiv preprint arXiv:2402.04367*, 2024. [Online]. Available: https://arxiv.org/abs/2402.04367

[16] Ponemon Institute, "2022 Cost of Insider Threats Global Report," 2022. [Online]. Available: https://www.ponemon.org/

[17] L. Chen, Y. Xu, Z. Yang, and Y. Zhou, "Lightweight blockchain-based secure logging for IoT," *J. Netw. Comput. Appl.*, vol. 189, p. 103113, Sep. 2021. doi: 10.1016/j.jnca.2021.103113.

[18] "GDPR: General Data Protection Regulation," Official Journal of the European Union, 2016. [Online]. Available: https://gdpr-info.eu/

[19] "Health Insurance Portability and Accountability Act of 1996 (HIPAA)," CDC, 1996. [Online]. Available: https://www.cdc.gov/phlp/publications/topic/hipaa.html

[20] P. Maniatis and M. Baker, "Secure history preservation through timeline entanglement," in *Proc. 11th USENIX Security Symposium*, San Francisco, CA, 2002, pp. 1-13.

APPENDICES

Appendix A: Risk Assessment Matrix

Risk ID	Risk Description	Likelihood	Impact	Risk Level	Mitigation Strategy
R1	**Log Tampering** – Attacker modifies/deletes log entries to hide traces	High	Critical	High	Use forward-secure keys + Merkle tree chaining to ensure tampering is immediately detectable
R2	**Crash/Rollback Attack** – Logs erased during abnormal shutdown	Medium	High	High	Implement crash-resilient logging protocol (SLiC-like) and periodic cloud anchoring of Merkle root
R3	**Key Compromise** – Adversary gains access to current key	Medium	High	High	Use forward security (key evolution); limit key exposure by frequent rotation and storing master keys in HSM
R4	**Performance Bottlenecks** – High log volume slows system	Medium	Medium	Medium	Benchmark against 10^3–10^6 logs; optimize storage with log compression and selective encryption
R5	**Insider Threat** – Malicious admin tries to manipulate logs	Medium	High	High	Enforce role-based access control (RBAC); require two-party verification for log deletion/export
R6	**Scalability Limitations** –	Low	Medium	Low-Medium	Design system with modular

	SME grows beyond current log handling capacity				architecture; allow horizontal scaling via database sharding
R7	**Privacy Breach** – Sensitive user data exposed in logs (GDPR risk)	Medium	High	High	Apply selective encryption for personal data fields; conduct privacy impact assessments
R8	**Compliance Failure** – Logs not accepted in regulatory audits	Low	High	Medium	Ensure logs comply with ISO 27001, GDPR, HIPAA, PCI-DSS standards; maintain audit-ready proofs
R9	**Cloud Dependency Risk** – Cloud anchoring unavailable (downtime)	Medium	Medium	Medium	Use multi-cloud strategy (AWS/GCP/Azure) or fallback to local immutable storage
R10	**Budget Overrun** – Unexpected costs for cloud services	Low	Medium	Low	Limit to free tiers/student credits; only scale cloud if pilot deployment requires

Appendix B: Ethical Considerations

The proposed cryptographically secured logging system involves the collection, storage, and verification of log data. Although logs are technical artifacts, they may contain sensitive information such as user IDs, IP addresses, or system activity records. Therefore, several ethical issues must be addressed:

Data Privacy and Minimalism

- Only essential log data will be collected to meet security and forensic needs.

- Personally identifiable information (PII) will be excluded or pseudonymized wherever possible.
- Logs containing sensitive fields will be encrypted to prevent unauthorized access.

User Consent and Transparency

- In deployment scenarios, SMEs will be informed about what data is logged and why it is necessary.
- End-users will be informed (where applicable) that their activity may be recorded for security and compliance purposes.

Compliance with Legal Frameworks

- The project aligns with GDPR, HIPAA, and other data protection regulations.
- Data retention policies will ensure logs are kept only as long as required for compliance or forensic purposes.
- Secure deletion procedures will be followed once retention periods expire.

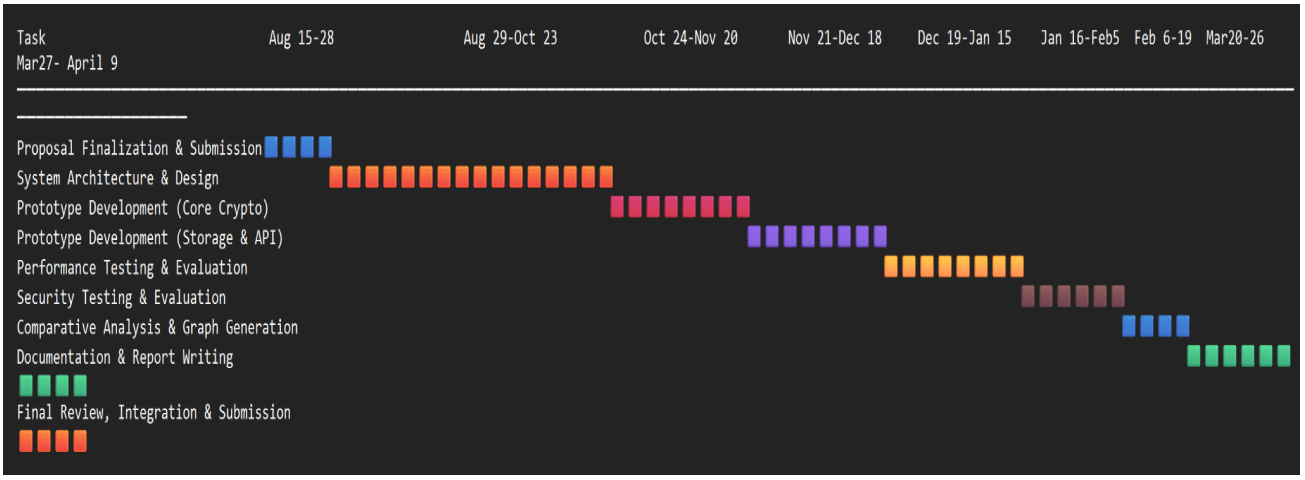
Security and Integrity of Research Data

- All test data used in experiments will be synthetic or anonymized, ensuring no exposure of real user data.
- System prototypes will be deployed in controlled lab or SME test environments with proper safeguards.

Responsible Disclosure and Use

- Any vulnerabilities discovered during development/testing will be responsibly disclosed and documented.
- The system will not be misused for surveillance or unethical monitoring of individuals.

Appendix C: Project Timeline (Gantt Chart)



Detailed 30-Week Timeline:

1. 🟦 Proposal Finalization & Submission (2 weeks)
    - August 15 - August 28, 2025
2. 🟧 System Architecture & Design (4 weeks)
    - August 29 - October 23, 2025
3. 🟥 Prototype Development (Core Crypto Modules) (4 weeks)
    - October 24 - November 20, 2025
4. 🟪 Prototype Development (Storage & API Integration) (4 weeks)
    - November 21 - December 18, 2025
5. 🟨 Performance Testing & Evaluation (4 weeks)
    - December 19, 2025 - January 15, 2026
6. 🟫 Security Testing & Evaluation (3 weeks)
    - January 16 - February 5, 2026
7. 🔵 Comparative Analysis & Graph Generation (2 weeks)
    - February 6 - February 19, 2026
8. 🟢 Documentation & Report Writing (5 weeks)
    - February 20 - March 26, 2026
9. 🟠 Final Review, Integration with ForLens & Submission (2 weeks)
    - March 27 - April 9, 2026