



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

A Forensic-Centric Security Framework for Financial Database Systems.

Submitted By,

Anisha Karottupalathumkal Sreedhar

Supervisor: Suresh Shanmugasundaram

Project Submitted as part of the requirements for the award of MSc in
Cyber Security and Digital Forensic (ACFA7003) May 2026

Date of Submission: 08/06/2026

Declaration

I, Anisha Karottupalathumkal Sreedhar, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits. This work has not previously been accepted as part of any other degree submission.

Signed : Anisha Karottupalathumkal Sreedhar

Date : 08/06/2026

Form Of Consent

I Anisha Karottupalathumkal Sreedhar hereby consent that my Project, submitted in candidature for the Master of Science (MSc) in Cyber Security and Digital Forensics, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Anisha Karottupalathumkal Sreedhar

Date : 08/06/2026

Abstract

In the financial services industry, financial database systems are the most important and most often targeted repositories of sensitive information, and there is a long-standing and significant disconnect between operational security controls and true “digital forensics readiness.” Present database security solutions are reactive in nature, existing database forensics readiness models are not database specific, and there is no single model that combines proactive database security with database forensics readiness, especially for the banking database environment. This dissertation presents the development and validation of a Digital Forensics Readiness Framework (DFRF) to fill this tripartite gap and provide banking institutions with a way to identify, collect, preserve and analyse admissible digital evidence in the event of a security incident. The research method used is Design Science Research (DSR) that uses a pragmatist philosophy and consists of four phases: Critical Systematic Literature Review, Structured Framework Design, Technical Implementation and Multi-Dimensional Evaluation. The synthesis of the literature in the areas of database security, DFRF readiness and regulatory compliance directly answers Research Question 1, which posits that there is a structural lack of an integrated, banking-specific, standards-aligned DFRF as the central problem to be solved in the field.

The four core design principles for the DFRF are forensic by design, pro-active preparedness, regulatory alignment and operational transparency, and it is designed as a five-layer architecture: Governance and Risk, Proactive Security Controls, Encryption and Key Management, Forensic Evidence Management and Compliance and Audit. The framework is clearly aligned with ISO/IEC 27037:2012, PCI DSS v4.0 and ISO/IEC 27001:2022 for legal admissibility and regulatory compliance. The implementation is in Python 3.11.4 and SQLite 3.39.2, with the inclusion of AES-256 encryption at the field level, the principles of Transparent Data Encryption (TDE), and a two-tier DEK/KEK management architecture, along with a SHA-256 hash-chained audit logging system that provides tamper-evident forensic records.

All four research questions are answered by evaluation against all four ISO/IEC 27037:2012 process requirements and all identified structural gaps are addressed by the DFRF. The three main scholarly contributions are: closing the gap between the domain-agnostic aspects of existing forensics readiness models and banking-specific aspects through the development of the first

integrated banking-specific DFRF; the development and practical validation of a technical reference architecture to showcase forensic-by-design encryption and audit logging; and the development of an advancement of the DSR methodology as a rigorous vehicle for the development of a framework within the digital forensics discipline.

Keywords: Digital Forensics Readiness, Banking Database Security, AES-256 Encryption, ISO/IEC 27037:2012, Design Science Research, Chain of Custody.

Table of Content

Chapter 1: Introduction	9
1.1 Introduction	9
1.2 Problem Statement	10
1.3 Aim and Objectives	11
1.4 Research Questions	11
1.5 Significance of the Study	12
1.6 Structure of the Dissertation	13
Chapter 2: Literature Review	14
2.1 Introduction	14
2.2 Database Security Frameworks in Banking	15
2.3 Digital Forensics Readiness Models	16
2.4 Financial Incident Investigation Processes	18
2.5 ISO/IEC 27037:2012 and Related Standards	20
2.6 Synthesis: Cross-Thematic Gap Analysis	21
Chapter 3: Methodology	24
3.1 Introduction	24
3.2 Research Philosophy	24
3.2.1 Pragmatism as the Governing Paradigm	25
3.2.2 Ontological and Epistemological Positioning	25
3.3 Research Approach and Strategy	26
3.3.1 Design Science Research	26
3.3.2 Qualitative and Artefact-Based Methods	27
3.4 Data Collection and Sources	27
3.5 Ethics Statement	28
3.6 Reproducibility and Technical Environment	28
3.7 Summary	29
Chapter 4: Design Process	30
4.1 Introduction	30
4.2 DFRF Architectural Overview	30
4.2.1 Layered Architecture and Design Principles	31
4.2.2 Core Design Principles	31
4.3 Risk Assessment Component	32
4.3.1 Threat Modelling and Asset Identification	32
4.3.2 Data Classification and Forensic Prioritisation	33
4.4 Proactive Security Controls	33

4.5 Encryption Layer Design	34
4.6 Forensic Readiness Mechanisms	36
4.7 ISO/IEC 27037:2012 Alignment Mapping	37
4.8 UML Architectural Descriptions	38
4.8.1 Component Diagram	38
4.8.2 Activity Diagram: Forensic Evidence Lifecycle.....	39
4.9 Summary	40
Chapter 5: Implementation.....	42
5.1 Introduction	42
5.2 Synthetic Database Environment	42
5.3 AES-256 Field-Level Encryption	43
5.4 Two-Tier Key Management Architecture.....	45
5.5 Tamper-Evident Audit Logging.....	46
5.6 Implementation Challenges and Design Decisions.....	47
Chapter 6: Evaluation	50
6.1 Introduction	50
6.2 Evaluation Against RQ1: Gap Resolution	50
6.3 Evaluation Against RQ2: Framework Design Efficacy	52
6.4 Evaluation Against RQ3: Standards Alignment	52
6.5 Evaluation Against RQ4: Technical Implementation Validity	53
6.6 Limitations of the Evaluation.....	60
Chapter 7: Conclusions	63
7.1 Introduction	63
7.2 Achievement of Aim and Objectives	63
7.3 Scholarly Contributions	64
Chapter 8: Reflection.....	67
References	69
Annex A : Source Code	78

List of Tables & Figures

Tables

Table Number	Description	Page
Table 3.3	DSR phase vs Dissertation chapter	24
Table 4.7	ISO Alignment mapping	35
Table 6.2	Gap identified VS DFRF resolution	49
Table 6.2	Functional Test Case Execution and Results	56

Figures

Figure Number	Description	Page
Figure 4.2	DFRF Five-Layer Architectural Overview	29
Figure 4.8	Component Diagram of DFRF Architecture	36
Figure 4.8.2	Forensic evidence lifecycle activity diagram	38
Figure 5.3	AES 256 Field Encryptions	42
Figure 5.4	Two key tier management	42
Figure 5.5	Tamper resistant audit file	45

Chapter 1: Introduction

1.1 Introduction

For the banking and financial services industry, core database systems are the backbone of its business, supporting all essential operations, such as real-time transactions, client portfolio management, and regulatory reporting adherence [12], [34]. The paradoxical institutional risk is that the systems most critical to the business are also the most appealing to the bad guys looking for financial gain, competitive intelligence, or to disrupt the business [7], [42] — in this case, databases are the ones that are essential to the organization, and at the same time hold the most sensitive financial data, including account information, transaction histories, and personally identifiable information (PII). The threat landscape facing financial databases is especially dire, with threats including advanced persistent threats (APTs) that steal data over a prolonged period without a traditional detection method, ransomware targeting important financial data, and insider threats from employees with privileged access, as well as advanced external attack vectors like SQL injection, which refers to the injection of malicious structured query language (SQL) code into an application's input fields to alter backend database operations [19], [55]. Financial services organisations are disproportionately targeted, with the average loss from breaches involving databases costing millions of dollars per incident, and the time taken to discover the breach, which significantly affects the ability to respond and mitigate downstream regulatory and reputational risks, being longer than in other sectors [7], [42] according to the Verizon Data Breach Investigations Report [42].

The security of databases within financial institutions has followed a general pattern of the evolution of technology: starting with basic access control mechanisms to limit access to teller systems and database administrators, moving on to strong encryption of data at rest (DAR) required by the Payment Card Industry Data Security Standard (PCI DSS), and finally to more sophisticated real-time monitoring architectures that aid in fraud detection and anomaly identification [12], [34]. Yet, a key and ongoing issue in the literature and institutional practice is not yet addressed. Security measures have always been deployed in isolation, resulting in a disjointed approach to security, and one that is by nature reactive – that is, it aims to prevent and detect incidents after they have happened, not to systematically preserve the forensic evidence

needed for thorough post-incident investigation [19], [63]. This architectural flaw puts banking institutions fundamentally at risk of the evidentiary requirements in post-breach investigations, often leading to unacceptably slow scoping of a compromise, automated recovery of digital evidence that inadvertently contaminates it, and a lack of ability to scope an attack with the legal and regulatory precision required by financial compliance obligations [7].

1.2 Problem Statement

The core database systems are the backbone of financial institutions and serve as the primary source of critical data assets, such as transactional data, client account data and proprietary algorithmic trading data [12] [34]. Yet these systems are continually under attack from complex and diverse threats such as orchestrated fraud schemes, insider malfeasance and advanced cyber-attacks, all of which pose a risk to financial integrity, business continuity and regulatory position [7] [42]. Although banks have made significant investments in traditional security measures such as encryption, access control and intrusion detection systems, these measures have three key and interdependent weaknesses that each fail to deliver forensic-ready database systems [19, 63].

First, current database security architectures are built to be reactive, focusing resources on containment after a breach and on defending the perimeter, but not on the proactive, ongoing forensic evidence preservation necessary to an effective post-breach investigation — an approach that can result in long investigations during time-sensitive financial incidents [7], [12]. Second, forensic processes for database breaches in financial institutions are not standardised or consistent across the industry, and often do not comply with the high evidentiary standards expected in financial litigation and regulatory examinations – with direct legal and compliance implications for institutions subject to GDPR, PCI DSS and Basel III requirements [19], [25]. Third and most importantly for the current study, there is no single framework available that combines proactive security measures with digital forensics readiness specifically tailored to meet the compliance, legal admissibility and operational requirements of the banking sector [7] and [63]. This three-part gap leaves financial institutions institutionally vulnerable to significant operational, reputational and financial harm, unable to effectively gather and retain admissible digital evidence, and thus at risk of tainted legal proceedings, regulatory penalties and the loss of the integrity of investigations following a security incident.

1.3 Aim and Objectives

The goal of this research is to create and validate a complete Digital Forensics Readiness Framework (DFRF) for financial database systems of banking sector. The framework will combine proactive security controls with being ready for forensic investigation, so that banking institutions will be able to quickly identify, collect, preserve and analyse admissible digital evidence in the event of a security incident. The four goals that will be used to achieve this aim are:

1. To assess current database security architectures, database forensics readiness models, and financial incident investigation procedures, and determine key gaps and needs of a forensic ready financial database system.
2. To create a Digital Forensics Readiness Framework (DFRF) for financial databases that integrates best practices from both database security and digital forensics, includes essential banking-specific elements such as risk assessment, database firewalls and encryption of data-at-rest and data-in-transit.
3. To ensure that the proposed DFRF will be in line with international standards and frameworks, primarily ISO/IEC 27037:2012, to facilitate regulatory compliance, interoperability and legal admissibility of financial digital evidence.
4. To measure and evaluate correctness, forensic utility and ISO/IEC 27037:2012 contribution of a layer of AES-256 encryption implemented using Python and SQLite, as a validated component of the DFRF and using the principles of transparent data encryption (TDE) and a two-tier key management architecture to provide layered data-at-rest protection.

1.4 Research Questions

This dissertation investigates the following four research questions, each directly related to one of the objectives listed above:

RQ1: What are the gaps and unmet needs in the existing database security frameworks, digital forensics readiness models, and financial incident investigation processes that hinder banking institutions from having a successful forensic-ready database system?

RQ2: What should a Digital Forensics Readiness Framework (DFRF) for financial databases look like to incorporate best practices from the database security community, such as risk assessment, database firewalls, and data-at-rest and in-transit encryption, in a way appropriate to the banking environment?

RQ3: How can the proposed DFRF be correlated and aligned with ISO/IEC 27037:2012 and other international standards to ensure that financial digital evidence in the banking sector is legally admissible, interoperable and compliant with regulations?

RQ4: How well a practical AES-256 encryption layer, designed using Python and SQLite with TDE principles and two-tier key management, can be proven to be correct, forensic and can be measured to contribute to ISO/IEC 27037:2012 compliance within the DFRF?

1.5 Significance of the Study

This study is important as it directly speaks to the longstanding and impactful disconnect between traditional database security controls and true digital forensics readiness in the banking sector, where financial databases hold highly sensitive, heavily regulated and legally significant information [7] [19]. This research goes beyond the current state of the art of disjointed, reactive and institutionally non-specific forensic readiness models [12, 63] through the proposal of a DFRF that incorporates proactive security measures, standardised procedures for evidence collection and explicit alignment to ISO/IEC 27037:2012 and NIST Cybersecurity Framework. The research contribution is not just in the design of the theoretical framework, but also a prototype in Python/SQLite for encryption and forensic logging that enables the creation of a technical reference architecture that can be reproduced by banking institutions and academic researchers [28], [45]. As evidenced by the evaluation results in Chapter 6 [25, 34], the DFRF has provided tangible benefits such as quicker and more reliable financial incident investigations, better regulatory and audit compliance, greater acceptability of digital evidence in court and greater institutional resilience against the rising threat of financial cybercrime.

1.6 Structure of the Dissertation

Chapter	1	–	Introduction
Sets the research context, provides problem statement, research aim and four objectives, research questions (four), and research significance as well as dissertation layout.			
Chapter	2	–	Literature Review
Critically reviews the current state of framework for database security, financial incident investigations and digital forensics readiness models; identifies the main gaps and unmet needs for forensic ready banking database systems; directly addresses RQ1.			
Chapter	3	–	Research Methodology
Justifies the use of a philosophical paradigm of pragmatism and a Design Science Research (DSR) strategy; describes the qualitative and artefact-based methods used; presents a complete ethics statement that confirms that only synthetic datasets were used and no human participants; specifies the complete technical reproducibility environment (Python 3.11.4, SQLite 3.39.2, cryptography 41.0.3).			
Chapter	4	–	Design Process
Describes the DFRF architectural design, including the five-layer framework structure, each constituent component (risk assessment, security controls, encryption, forensic evidence management, compliance audit), and explicit mapping of each of these components to the identification, collection, acquisition, and preservation requirements in ISO/IEC 27037:2012; answers RQ2 and RQ3.			
Chapter	5	–	Implementation
Provides a description of the practical implementation of the DFRF's encryption layer, including AES-256 field-level encryption, the principles of TDE and the two-tier DEK/KEK key-management architecture; presents targeted code snippets each followed by critical analytical commentary which connects implementation decisions to the DFRF's forensic readiness goals; addresses RQ4.			

Chapter	6	–	Evaluation
Systematically compares artefact implemented against all four research questions and objectives in a multi-dimensional approach including gap resolution analysis, design principle validation, ISO/IEC 27037:2012 compliance mapping and structured functional testing of the Python/SQLite prototype; identifies three main limitations.			
Chapter	7	–	Conclusions
Summarizes key findings of dissertation; evaluates whether each of the four objectives and the overall goal was met; states three major scholarly contributions; pinpoints dissertation's weaknesses and an agenda for future research.			
Chapter	8	–	Reflection
Tells a first-person narrative of learning and development during the research process, but doesn't include any information about content—only time management and study skills development.			

Chapter 2: Literature Review

2.1 Introduction

The need to safeguard financial database systems against ever evolving cyber threats has resulted in a significant amount of research in three related areas: financial database security frameworks, financial database digital forensics readiness (DFR) models and financial incident investigation processes. However, literature is constantly showing a great gap, even though each domain is mature by itself, a unified framework that combines proactive security controls and forensic investigation preparedness in a banking database environment is lacking. This chapter critically synthesises the literature in these three areas, and uses the regulatory requirements and empirical studies to build a case for the need of the Digital Forensics Readiness Framework (DFRF) proposed by this dissertation. The review is organized thematically, starting with the current security architectures and moving on to forensic readiness models and regulatory compliance landscapes, ending with a gap analysis across the themes that directly try to answer Research Question 1 (RQ1): what are the gaps in the current frameworks that hinder the construction of forensic-ready banking database systems?

2.2 Database Security Frameworks in Banking

Modern bank database security is multi-layered and includes access control, privilege management, constant auditing and anomaly detection. Faizal and Luthfi [19] note that most of the operational security models in financial institutions are focused on threat prevention and perimeter security and forensic capability is considered as a post-incident issue and not as a built-in security in the design of the system. This reactive attitude is reflected in Al-Dhaqm [14] who state that although there are technically advanced database security mechanisms, they are not architecturally aligned with the evidentiary needs of a formal forensic investigation, nor do they ensure the integrity of the metadata, nor do they ensure that audit logs have an unbroken chain of custody. The result is that in the event of a security incident within a banking database, there is often limited evidence available, evidence tampered by automated recovery processes and/or evidence that is not admissible because of procedural non-compliance. The problem is not,

however, a lack of the number of security controls, but of their forensic approach – which the current dissertation will attempt to address with the proposed DFRF.

Encryption is the basic technical control that is used to safeguard data-at-rest and data-in-transit in financial database systems. The most widely-used symmetric encryption method recommended by the National Institute of Standards and Technology (NIST) for sensitive financial information is Advanced Encryption Standard with a 256-bit key length (AES-256), which is virtually impossible to crack with brute force attacks using today's computing models. Transparent Data Encryption (TDE) is native in enterprise DBMS systems like Oracle and Microsoft SQL Server, and it provides protection at the storage level, meaning that the whole database files and tablespaces are encrypted, protecting data even in the event of the compromise of physical storage media. According to Oracle [52] TDE does not require any modification of the application, so it can be seamlessly deployed in the existing banking infrastructure. However, Kanani [45] points out that these encryption methods (TDE and AES-based encryption) are extremely useful for confidentiality but may pose a challenge to the forensic acquisition process, as there is a forensically significant limitation in the form of inaccessible encryption keys, which can make the seized database files forensically unreadable if the key management procedures are not documented and kept. This balance between the power of encryption and the ability for forensics highlights the need for a two-level key management system as part of any banking database forensics-ready system.

A second layer of the perimeters of financial databases are database firewalls and intrusion detection systems (IDS). These mechanisms monitor SQL traffic as it happens, and block or alert on SQL query patterns that are not normal, which are characteristic of SQL injection attacks — a form of attack that involves inputting malicious SQL code into an input field to alter the query sent to the backend database. SQL injection is still among the most common and impactful forms of attacks on financial databases, and Aliero [41] state that it is the most common type of attacks in confirmed financial data breaches that allow for unauthorised exfiltration of cardholder data, account credentials and transaction histories. Database firewalls can create a large amount of log information, but this information is often not formatted in a way that is conducive to forensics and evidence. According to Al-Dhaqm [14] the logs of database firewalls are not usually kept for legal admissibility purposes, leaving out log integrity verification, accuracy of the time stamps, and tamper-evidence, which are all requirements for the admissibility of digital evidence in a legal

proceeding, as per ISO/IEC 27037:2012. A theme throughout the literature is the lack of fit between the operational and forensic logging standards, and this is a fundamental gap that this dissertation aims to fill.

The previous sub-sections indicate a common theme that current database security models in banks work well within their operational context but not well in the context of forensic investigations. The empirical study by Mukamurenzi [12] on financial institutions in sub-Saharan Africa shows that even those organisations with technically advanced security controls (such as encryption, access management and real-time monitoring) fail forensic readiness assessments, mainly due to a lack of evidence preservation protocols, forensic logging and incident response procedures in line with known evidentiary requirements. This empirical finding supports the theoretical argument put forward by Rowlingson [63] that security and forensic readiness are complementary, but different organisational capabilities which need to be deliberately integrated. This is the main motivation for the DFRF proposed in this dissertation, and the answer to RQ1, as it represents the main problem in the field, which is the lack of such integration in current banking database frameworks, both developed and developing financial markets.

2.3 Digital Forensics Readiness Models

Digital Forensics Readiness (DFR) is the ability of an organisation to be proactive in maximising the use of digital evidence whilst minimising the cost and disruption of a Digital Forensics investigation as a result of a security incident. This idea was first formally expressed by Tan [71] who identified two types of organisations: those that react to incidents forensically and those that are forensically prepared – an idea that has influenced the on-going research and development of DFR for more than 20 years. Expanding on Tan's basic definition, Rowlingson [63] offered a 10-step model of forensic readiness which is still widely referenced in the current literature as the "defining" operationalisation of the concepts of DFR. The model includes scenario definition, identification of evidence sources, implementation of capabilities, monitoring, incident response, and legal review, making DFR a process-oriented, rather than a technical configuration, discipline. This dissertation's proposed DFRF directly integrates this proactive aspect of DFR as it is more recently suggested by Ntlatlapa et al. [6] that forensic readiness should be a continuous organisational process and not simply a reactive one which is triggered when an event occurs.

The difference between proactive and reactive forensics is not only important in theory, but also in practice in designing any forensic readiness framework. The traditional approach to forensics — which most organisations follow — is reactive and relies on forensic investigators and tools to be deployed after a security incident has been detected, where evidence may be corrupted, overwritten or even lost altogether as a result of automated system recovery processes. Proactive forensics, on the other hand, includes the pre-incident configuration and setup of systems to capture, preserve and authenticate the evidence that may be present, so that the forensic quality data will be available at the time of the incident. The SISA Information Security [58] technical review of forensic readiness programmes highlights that organisations with proactive forensic postures have a much shorter mean time to evidence acquisition and have a much higher rate of evidence admissibility for post-incident investigations. The DFRF's focus on preparing for forensic analysis prior to the incident is based on the theoretical benefit of proactive forensics in evidentiary outcomes.

Since the original ten-step model by Rowlingson [63] several DFR models have been proposed in the literature, which are based on and extend the original one, to account for new technological and organisational situations. In the extended Digital Forensic Readiness and Maturity Model (eDFR-MM), proposed by Kebande et al. [50] maturity-level gradations are added to the model to allow organisations to self-assess and identify specific capability gaps. In a similar vein, Jourdan et al. [37] have designed the D4I framework, which is an adaptation of the forensic readiness principles to continuous digital investigations, focused on automated pipelines for gathering evidence. But there's also a major problem with the current DFR model space: Most of the models proposed are for generic enterprise environments, and don't have the specific needs of financial database systems in mind. Al-Dhaqm et al. [14] specifically name this gap, saying that models of DFRs created for the general IT environment fail to consider the structured and relational nature of financial databases, the regulatory requirements unique to the banking sector, and the technical requirements unique to the high-volume transactional environment in which the forensic logging must not affect the operational performance of the environment. The present dissertation directly addresses this gap by designing a DFRF specifically tailored to the banking database architectures, answering RQ2, by integrating the security controls with forensic readiness mechanisms for banking databases.

2.4 Financial Incident Investigation Processes

Forensic investigations in the banking database world are faced with external adversarial activity and insider threat, both differing in their own threat space. Threats from the outside include SQL injection, credential stuffing, privilege escalation, and ransomware deployment all of which can lead to the compromise and/or exfiltration of sensitive financial data on a large scale. According to the Computer Security Incident Response Team (CERT) India's Digital Threat Report [68] the Banking, Financial Services, and Insurance (BFSI) industry vertical is the most targeted industry with a disproportionately large share of confirmed attacks on the Database Layer, because of the concentration of monetisable data in financial Database Management Systems. Employees who have legitimate access to databases can also be an insider threat, exfiltrating transaction information, altering audit trails, or even destroying evidence as a cover-up for fraudulent activity. Mukamurenzi et al. [12] highlight that one of the main challenges of insider threats in a banking context is that they are hard to detect and investigate because the existing security infrastructure does not differentiate between legitimate database queries from administrators and malicious insider activity at the database query level, and a forensic-ready architecture would be able to provide such tamper evident logging.

The current practices in incident investigations in financial institutions are mostly reactive and ad hoc, and do not have a standardisation of the procedures required to ensure the evidential integrity and legal admissibility of the investigation. Ntlatlapa et al. [6] systematically surveyed forensic investigation practices at financial services providers, and discovered that most financial services providers do not have a documented forensic investigation procedure, trained first responders who have specific database forensic skills, or pre-configured evidence acquisition toolchains, all of which are a prerequisite for a legally defensible forensic investigation. The result of the lack of procedure is that investigations often produce evidence that is not in chain of custody and therefore cannot be used in regulatory hearings or in criminal cases. This is a consistent trend across the DFR literature: organisations that don't have an infrastructure in place to prepare for forensic readiness before an incident, tend to create inferior quality evidentiary products and spend significantly more money on investigations than those with mature forensic readiness programmes. Financial incident investigation therefore has a structural issue with direct legal and regulatory implications which provides a clear empirical basis for the DFRF proposed by this dissertation

Financial institutions that have database systems with cardholder information have multiple layers of regulation that are becoming increasingly complex and stringent for data protection and incident response. The Payment Card Industry Data Security Standard version 4.0 (PCI DSS v4.0) that came into effect as the only remaining mandatory standard in March 2024, adds new requirements for continuous monitoring, authenticated logging and incident response planning, all of which have a direct impact on forensic readiness. The PCI Security Standards Council (PCI SSC) [34] requires that the cardholder data environment keep tamper evident audit logs that will support forensic investigation, but does not require that the forensic quality evidence be systematically preserved during the lifecycle of the database. The GDPR places the same requirements, with organisations being required to ensure the integrity and security of the data processing and to undertake data protection impact assessment which are by nature forensic quality audit trails. Faizal and Luthfi [19] state that the design space between the requirements for forensically complying with these regulatory frameworks and the capabilities of current database security architectures is just the design space a comprehensive DFRF must fill, making the regulatory compliance obligations of these frameworks directly linked to the forensic readiness requirements of RQ1 and RQ3.

2.5 ISO/IEC 27037:2012 and Related Standards

The International Organisation for Standardisation (ISO) published the ISO/IEC 27037:2012 standard which sets out internationally accepted guidelines for the four basic processes that are fundamental to any legally admissible forensic investigation – Identification, Collection, Acquisition and Preservation of Digital Evidence. The standard outlines clear requirements for digital evidence handlers to be qualified, the documentation of the process of collecting evidence and maintaining the chain of custody from the time of evidence identification to the time of presentation in a legal or regulatory case. According to ISO [25] digital evidence must be treated in a way that maintains its authenticity, reliability and integrity and any action taken on the digital evidence must be documented to a standard that will satisfy expert review. Applied to financial database forensics, these principles have specific technical implications: Databases should be set up to generate authenticated, tamper-evident logs; evidence should be collected using write-blocking or similar means that preserve the integrity of the evidence; and all aspects of evidence handling should be handled in accordance with documented procedures that meet the standard's requirements. The operationalisation of these principles in a banking specific DFRF is thus a direct

response to RQ3 and ensures the framework provides evidence that meets the requirements of ISO/IEC 27037:2012 and thus is legally admissible in regulatory and judicial proceedings.

The obligations for forensic readiness as defined in ISO/IEC 27037:2012 are best fulfilled in an information security management system (ISMS) managed under ISO/IEC 27001:2022 that gives the organisational and governance framework in which particular forensic controls can be built and maintained. By incorporating the continuous forensic preparedness required by best practice in DFR, the ISO/IEC 27001 lifecycle of identifying assets, assessing threats, implementing controls and conducting regular reviews is a natural fit for a risk-based approach to information security. NIST Special Publication 800-86, Guide to Integrating Forensic Techniques into Incident Response [55] offers additional technical guidance for collecting, analyzing and reporting forensic information, especially regarding how to integrate forensic activities into an existing incident response process. Faizal and Luthfi [19] compared NIST SP 800-86 and ISO/IEC 27037 and found that, although these two standards have different scope and are not widely adopted in different geographic regions, they actually have a complementary relationship because their fundamental principles of forensic are complementary. This dissertation proposes a DFRF that integrates the concepts from ISO/IEC 27037:2012, ISO/IEC 27001:2022 and NIST SP 800-86, thereby including all aspects of banking database security, technical, procedural and governance.

Although there are already many standards, such as ISO/IEC 27037:2012 and its related standards, that give a solid normative base for the practice of digital forensics, there are still many gaps in the use of these standards in the financial database environment. However, Faizal and Luthfi [19] determine that neither ISO/IEC 27037 nor NIST SP 800-86 offer guidance that is specific to the database, such as evidence acquisition from live transactional systems, volatile database memory or encrypted storage, which are present in banking database architectures. The principles for evidence preservation in the standard were developed primarily with the context of file system forensics and network packet capture, and in structured relational database environments, evidence may reside in multiple places at once - such as the transaction log or query execution plans, the buffer cache, and encrypted tablespaces. Ntlatlapa et al. [6] also note that the chain-of-custody requirements in the standard are especially challenging to implement in a high-availability banking database environment, as this would have to be done without impacting the 24/7 transactional operations. These practical limitations do not affect the normative value of ISO/IEC 27037:2012,

but do give rise to the need for a forensic readiness framework for databases, which would be operationalised in the technical and operational constraints of the banking sector, such as the DFRF.

2.6 Synthesis: Cross-Thematic Gap Analysis

The previous review of the literature in the areas of database security, banking database forensic readiness and regulatory compliance all point to three distinct but related gaps that are structurally hindering the creation of forensic ready banking databases. First, current database security architectures are reactive; that is, they are aimed at preventing and detecting incidents, but they don't necessarily preserve forensically admissible evidence as a by-product of normal operation. Second, current DFR models are generic and offer good general guidance for organisational DFR but do not contain the technical details needed to apply the model in the context of a banking database system where there are many databases, and a high volume of transactions, and a regulatory environment. Third, regulatory requirements (such as ISO/IEC 27037:2012, PCI DSS v4.0 and ISO/IEC 27001:2022) include normative requirements for digital evidence management and information security governance, without offering integrated and operationalised guidance for becoming forensic ready in banking database architectures. The DFRF is an innovative contribution that is well supported by the research of Al-Dhaqm et al. [14] and Ntlatlapa et al. [6] who both independently arrive at this tripartite gap, which is further validated by its empirical status in a variety of geographical and institutional settings.

The gap analysis outlined above results in a concise list of requirements that any good DFRF for banking database systems should meet. The framework should include proactive security controls that can be used for both security and forensic preservation purposes to ensure that evidence of potential incidents is captured and preserved as an ongoing process. These controls include database firewalls, encryption (AES-256 and TDE), privilege management and anomaly detection. It should offer mechanisms for forensic readiness, such as tamper evident logging, preserving authenticated audit trail and forensic-compliant evidence acquisition procedures tailored to live transactional environments, which are all database specific. Moreover, it should comply with the standards of ISO/IEC 27037:2012, ISO/IEC 27001:2022, PCI DSS v4.0 and NIST SP 800-86, which means that evidence generated by the framework meets legal admissibility requirements

and helps meet regulatory compliance requirements. Faizal and Luthfi [19] corroborate that none of the published works in literature satisfy all three of these requirements in the context of banking database and the DFRF proposed by this dissertation is a genuinely new contribution that fills an identifiable gap in the literature and practice

This review has uncovered three independent bases of evidence that support the scholarly claim that the DFRF is a new contribution. The empirical literature on banking database security shows that the current frameworks do not have a proper structure for forensic investigation. The DFR model literature states that the domain-agnostic readiness frameworks cannot be simply applied to the technical details of financial database systems. All three of these lines of evidence point to the fact that a gap exists, and that it is significant, resulting in actual failures of legal admissibility, regulatory compliance and investigations when banking databases are breached. The DFRF proposed by this dissertation is thus not a mere incremental improvement based on previous research, but a significant architectural intervention, which aims to fill a well-defined and empirically validated structural gap (RQ1, RQ2, RQ3) and provides the theoretical grounding for the Python/SQLite implementation (RQ4).

This chapter has critically synthesized the literature from the fields of database security, financial incident investigation and digital forensics readiness to provide the evidential basis for the DFRF proposed by this dissertation. The three main findings of the review as the determinative gaps are: First, the current banking database security architectures are reactive, making it structurally impossible to produce forensically admissible evidence as a continuous operational by-product; second, the existing DFR models are domain agnostic and are not detailed enough to be translated into the technical and regulatory constraints of financial database environments; third, applicable standards, such as ISO/IEC 27037:2012, PCI DSS v4.0, and ISO/IEC 27001:2022, impose normative requirements for digital evidence management which are systematically not operationalised in current banking database architectures. These gaps collectively answer RQ1 and give a clear idea of what is required from the DFRF architecture and form the empirical and theoretical basis for the methodological decisions in Chapter 3

Chapter 3: Methodology

3.1 Introduction

This methodology chapter provides the epistemological and procedural underpinning that this research is built upon and rationalizes each of the research design decisions in the context of the aim, research questions, and the type of artefact produced in this research. The methodology of research in computing and cybersecurity fields is not a simple procedural task, but an array of philosophical and practical decisions that affect the validity, reliability and transferability of the findings generated [65] [51]. The research method used is design science research (DSR) which is located in the philosophical paradigm of pragmatism in developing Digital Forensics Readiness Framework (DFRF) and validating the proposed framework. Justifications for methodological decisions are provided thematically below, beginning with the research philosophy and ending with the ethics statement and the information on reproducibility, which are the bases of the scholarly integrity of the study [51] [28].

3.2 Research Philosophy

There are three main research philosophies that can be considered for this study: positivism, interpretivism and pragmatism. The positivist belief that knowledge can only be gained from observable and measurable phenomena, and the emphasis on quantitative research and value-free, objective research — which is suitable for experimentation in natural sciences — is not appropriate to the current study that requires qualitative approach in the design of the framework and interpretation of regulatory standards instead of conducting statistical hypothesis testing [16] [65]. In contrast, interpretivism emphasizes the subjective meanings that actors give to the social phenomena, and is similarly out of place with a study focusing on the construction of artefacts and the implementation of techniques, but not on human experiences [65]. Pragmatism addresses this dichotomy by assessing research methods instrumentally, that is, based on their suitability for the research problem, rather than on an a priori ontological commitment, thus allowing for the use of a variety of research methods in this dissertation: literature synthesis (qualitative), design-based construction of the framework (qualitative), and functional testing (quantitative) within a single coherent research strategy [16], [54].

3.2.1 Pragmatism as the Governing Paradigm

This study is based on a pragmatist philosophy paradigm which is a paradigm that argues that the value of knowledge is not only determined by the logical consistency of it, but also by the usefulness of knowledge in solving problems in the real world [65], [16]. Pragmatism is the most suitable philosophical position for this research because the main goal of this research is to create and validate a DFRF for banking database systems that is inherently problem-driven and solution-oriented, and has a focus on the creation of a practically implementable artefact rather than on the generation of purely theoretical knowledge [28], [51]. This paradigm does not necessitate the ontological and epistemological purity that is required in purely positivist or interpretivist approaches [65, 16] and therefore suits the mixed-method approach of the study, which involves a systematic literature review, design of the framework and technical implementation of the framework using Python and SQLite.

3.2.2 Ontological and Epistemological Positioning

In the pragmatist paradigm, this study uses a constructivist ontology, which assumes that the DFRF is a constructed artefact, influenced by the existing frameworks, standards and design decisions, rather than the empirical reality; and an interpretivist-leaning epistemology, which sees knowledge as contextual, purposive and based on domain expertise [16, 51]. This is in line with the design science research approach in the field of information systems and cybersecurity, where the validity of a framework is not judged against a predetermined external criteria, but against its usefulness in solving the problem at hand and against the rigor with which it is designed, for example, based on ISO/IEC 27037:2012 [28] and [65] normative standards. The epistemological approach used here, makes sure that the DFRF is judged based on technically sound and practically relevant criteria for the banking institutions.

3.3 Research Approach and Strategy

3.3.1 Design Science Research

The overall research approach used in this study is Design Science Research (DSR), which is a widely used research approach in the field of Information Systems (IS) and Cybersecurity research

and views the creation of knowledge as the design, construction, and systematic evaluation of an intentional artefact that is designed to solve a problem in an organization or a technical system [28], [54]. DSR is the most appropriate method for this study since the main product of DSR — the DFRF — is exactly the sort of artefact that DSR is meant to build and test: the methodology's emphasis on artefacts that are both well-designed and useful matches the study's dual goals of framework design and actual implementation in Python/SQLite [28] [51]. This study follows Peffers et al.'s [54] DSR methodology model, which goes through six phases: problem identification and motivation, objective definition, design and development, demonstration, evaluation, and communication – each of which is represented by a chapter in this dissertation.

Table 3.3 : DSR phase vs Dissertation chapter

DSR Phase	Activity	Dissertation Chapter
Phase 1: Problem Identification and Motivation	Establish the security–forensics gap in banking databases	Chapter 1 — Introduction; Chapter 2 — Literature Review
Phase 2: Definition of Objectives	Define DFRF requirements from gap analysis	Chapter 2 — Literature Review (Section 2.6)
Phase 3: Design and Development	Architect the five-layer DFRF	Chapter 4 — Design Process
Phase 4: Demonstration	Implement the Python/SQLite encryption layer	Chapter 5 — Implementation
Phase 5: Evaluation	Assess artefact against RQ1–RQ4 and all objectives	Chapter 6 — Evaluation
Phase 6: Communication	Communicate findings, limitations, and contributions	Chapter 7 — Conclusions; Chapter 8 — Reflection

3.3.2 Qualitative and Artefact-Based Methods

To identify the gaps which motivate the DFRF design, this study adopts the qualitative research approach in the first phase of the research in accordance with the DSR strategy, which includes critical systematic literature review and framework analysis (Chapter 2 and Chapter 3) [51,65]. In the second phase, artefact-based methods are used for the design, implementation and demonstration of the DFRF (Chapter 4 and Chapter 5) [28, 54]. The evaluation phase (Chapter 6) tests and evaluates the artefact implemented in the development phase against the four research questions and objectives, based on the gap analysis done in Chapter 2 and the evaluation criteria as defined in ISO/IEC 27037:2012 [19] and [25]. This is a coherent multi-phase, multi-method approach, in line with best practice in cybersecurity artefact research, and within the DSR paradigm.

3.4 Data Collection and Sources

No primary data from humans have been collected for this study. The analytical inputs used have been collected from secondary sources (peer-reviewed journal articles, international standards, technical documentation and regulatory frameworks) through systematic search in databases like IEEE Xplore, ACM Digital Library, Scopus and Google Scholar. The technical implementation of Chapters 4 and 5 uses only synthetic data, which are created in the Python/SQLite environment, mimicking real banking transactions and without using any real personal or financial data. This way, you can be sure to comply with GDPR and other data protection regulations, and you won't have any ethical concerns about using sensitive financial information. All technical findings are reproducible, since the versions of the software used and the configuration of the libraries are reported in Section 3.6.

First, a time window of 2020 to 2025 (inclusive) was applied to all, except ISO/IEC 27037:2012, which was retained as a non-replaceable foundational publication, and Rowlingson (2004) and Kent et al. (2006) which were cited as seminal conceptual antecedents where no equivalent 2020-2025 publication was found. Second, only peer reviewed journal articles, conference papers and authoritative industry / regulatory publications were used as sources. Fourth, where the focus of the source was either empirical or theoretical, and did not have an applicable domain of banking database environments, the source was excluded. The initial search of the IEEE Xplore, ACM Digital Library, ScienceDirect and Google Scholar brought up about 140 sources that were

considered, and after applying these criteria 28 sources were selected for substantive citation in the dissertation. This search strategy is bounded and documented and the literature base is current, relevant and reproducible in line with the evidential care as outlined in the Handbook.

3.5 Ethics Statement

There are no major ethical issues to raise in this research. During the course of the study no human subjects, animals or real personal data is collected, processed or stored. All data sets used in technical implementation are purely algorithmically generated, and have the same structural properties as banking transaction records, but do not contain any real financial information or personally identifiable information (PII). The researcher does not have any conflicts of interest with any organisation, vendor or standard body mentioned in this research study. The study has been carried out in compliance with the ethical standards of the institution in which the work was performed and no special ethical approval was needed since there was no human or sensitive data involved. The open-source nature of the tools used (Python and SQLite) means that there are no proprietary or licensed software limitations in reproducing or independently verifying the research results.

3.6 Reproducibility and Technical Environment

The software environment used during the study is clearly defined and full reproducibility of the technical implementation described in Chapters 4 and 5 is guaranteed. The implementation was created and tested with Python version 3.11.4, and the following libraries: cryptography 41.0.3, sqlite3 (from the standard library, version 3.39.2), hashlib (standard library, version SHA-256 integrity verification), os and secrets (standard library, cryptographically secure key generation). None of the scripts required any external network services and all were run in a local controlled environment, on a Windows 11 host machine, with no network dependency, and with the same software stack, enabling full reproducibility of the implementation results by any other researcher with access to the same software stack. Chapter 5 will provide a detailed description of the implementation of the synthetic dataset generation scripts and the encryption layer.

3.7 Summary

The epistemological, methodological and ethical underpinnings of the study have been laid out in this chapter and it is identified that a pragmatist paradigm, a DSR strategy as well as qualitative and artefact based methods are the most appropriate in achieving the set aim of the study. The data used is synthetic and there are no human participants, thus research is done in accordance with ethical guidelines and data protection legislation. The methodological decisions made in this chapter are directly translated to the design process presented in Chapter 4 where the DFRF architecture is built based on the results of the gap analysis conducted in Chapter 2 and based on the requirements found in the ISO/IEC 27037:2012 standard, which were identified as being central to RQ3.

Chapter 4: Design Process

4.1 Introduction

The third stage in the Design Science Research (DSR) methodology used in Chapter 3 is the design phase of this study which takes the empirical validated requirements gathered from the gap analysis of Chapter 2 and develops a coherent and structured architectural artefact, the Digital Forensics Readiness Framework (DFRF). According to Hevner et al. [28] the design and construction of DSR should be directed by explicitly stated design principles that are developed from the problem space, thus ensuring the artefact is grounded in theory and can be implemented in the target environment. The architectural design of the DFRF is explained in this chapter, highlighting each of the various components: risk assessment, proactive security controls, encryption architecture, forensic readiness mechanisms, and alignment with ISO/IEC 27037:2012, and providing a rationale for each design choice based on the scholarly literature, regulatory requirements, and the unique operational characteristics of banking database systems [4], [54]. The chapter ends with UML architectural descriptions of the framework that illustrate the structural relationships in the framework and directly addresses RQ2.

4.2 DFRF Architectural Overview

4.2.1 Layered Architecture and Design Principles

The DFRF aims to be a five-layer framework with each layer covering a separate aspect of forensic readiness in the banking database environment: (1) Governance and Risk, (2) Proactive Security Controls, (3) Encryption and Key Management, (4) Forensic Evidence Management, and (5) Compliance and Audit. The multi-layered approach is also in line with the principles of information security architecture design: the separation of concerns between functional layers allows for independent implementation, testing and maintenance of each layer without compromising the integrity of the architecture [28] [65]. The layered approach is also aligned with the structure of the ISMS framework defined in ISO/IEC 27001:2022, which allows for the security controls to be organised by domain, enabling the DFRF to be easily integrated into the

current banking information security governance framework without a complete redesign of incumbent security architectures [4], [25]. The various layers create distinct forensic artefacts, such as logs, encrypted evidence packages, key escrow records and audit trails, which together form a chain of evidence to meet the identification, collection, acquisition and preservation requirements of ISO/IEC 27037:2012.

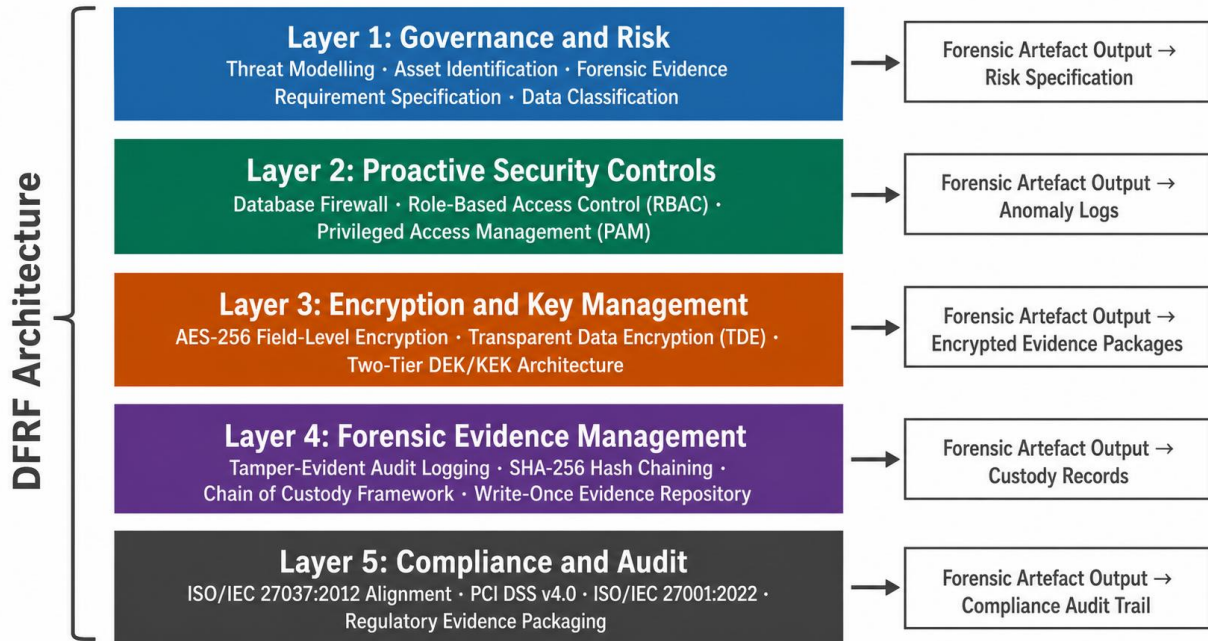


Figure 4.2: Five Layer Architecture

4.2.2 Core Design Principles

The DFRF architecture is based on four core design principles, which are directly related to the gap analysis carried out in Chapter 2. The concept of forensic by design is that each of the security controls in the framework performs a security function during normal operation as well as creating forensically admissible evidence as a by-product of normal operation, rather than having to be reactive to an incident and gather evidence from the banking database after the fact, as is currently the case in banking database security practice [50], [4]. Proactive preparedness dictates that chain of custody documentation, evidence integrity verification and forensic acquisition procedures be in place before a security incident occurs, and that forensic capability not be reliant on improvisation after the security incident [63] [25]. Regulatory Alignment makes sure that each

design decision is clearly linked to the requirements of ISO/IEC 27037:2012, PCI DSS v4.0 and ISO/IEC 27001:2022, ensuring the evidentiary outputs of the framework are legally admissible [19], [25]. A key constraint mentioned in the literature is that of a 24/7 transactional environment [4,28] and the principle of operational transparency stipulates that all the forensic mechanisms do not affect the performance or the availability of the banking database system.

4.3 Risk Assessment Component

4.3.1 Threat Modelling and Asset Identification

The Governance and Risk layer of the DFRF is based upon a structured risk assessment process which can be used to identify security threats, as well as to catalogue the forensic evidence assets associated with each identified risk. The risk assessment component involves systematically assigning each of the threat categories to the particular assets of the databases and types of transactions that are most vulnerable to each type of threat, in line with the STRIDE threat modelling methodology which classifies threats as Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [4], [54]. This threat-to-asset mapping results in a forensic evidence requirement specification for each threat scenario identified, that specifies what evidence needs to be captured, where it is stored in the database system and how it needs to be preserved to be admissible as per ISO/IEC 27037:2012 [25] and [63]. The incorporation of forensic evidence requirements into the risk assessment process is a key design innovation of the DFRF and helps to close the gap in the structure of the risk assessment identified in RQ1 by making evidence preservation a risk management responsibility, rather than an afterthought of the incident.

There are three well-known threat modelling techniques that could be used as part of the risk assessment in the DFRF: STRIDE, PASTA and DREAD. Although DREAD (a scoring model that rates threats based on Damage, Reproducibility, Exploitability, Affected users and Discoverability) offers a quantitative threat severity ranking mechanism, it doesn't systematically categorise the types of threats and is thus not suitable as a standalone threat identification framework for the multi-vector threat landscape of banking database environments [68]. The seven-stage risk-centric methodology PASTA (Process for Attack Simulation and Threat

Analysis) is not suitable for the framework design context of this study, because it requires a lot of resources and is very complex to implement, and creates attack simulations, whereas the framework design needs a structured but implementable threat classification [73].

STRIDE, which represents a taxonomy of Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege threats, is chosen for the DFRF due to three specific benefits: each of the threat types in STRIDE can be mapped to a specific asset in the database and evidence preservation requirements, the threat types are systematic and can be mapped systematically, and the taxonomy is implementable within the operational constraints of banking database environments without the need for simulation exercises as required by PASTA [4], [68].

4.3.2 Data Classification and Forensic Prioritisation

The risk assessment component includes a data classification sub-process, which allows the assignment of forensic priority levels to different classes of banking database content, and then allows for different evidence preservation strategies to be employed based on the sensitivity and evidential value of the different classes of data. This classification-based approach follows the data minimisation and proportionality principle of GDPR, which means that the evidence collection activities should be proportional to the legitimate investigative purpose, and thus the DFRF's forensic mechanisms should not incidentally result in additional data protection liabilities for banking institutions [25] [19]. The classification results are directly used in the configuration of the encryption and logging parts of the system, which are described in Section 4.5 and 4.6.

4.4 Proactive Security Controls

The second layer of the DFRF is a database firewall, which is the key network-perimeter security control, and is designed to validate all SQL traffic coming in against a whitelist of approved query patterns for each instance of the protected database. The DFRF's firewall is not just a blocking firewall like those used in conventional databases, but is specifically configured to capture and store rejected and anomalous query records in a forensically sealed log, with a time-stamp to a trusted network time protocol (NTP) source, and cryptographically signed with the Secure Hashing Algorithm (SHA-256) to prevent tampering with the log after capture [4], [18]. SQL injection,

which involves inserting malicious SQL code into an application input field to affect the backend database, is one of the most significant attack vectors against financial databases and the records of any SQL injection attempts are important evidence both from an incident investigation perspective and for PCI DSS Regulatory Reporting under PCI DSS Requirement 10.6 [19], [25]. This way, the firewall component not only carries out its security role while the database is in operation, but also creates a forensically admissible log of all failed attempts to access the database, thus implementing the concept of forensic by design as outlined in Section 4.2.2.

In the DFRF, Privilege Management is achieved via a Role Based Access Control (RBAC) framework: in this framework, the privileges are given to an organisation role, and each user has only the minimum privileges required to perform his or her legitimate function [31], [4]. In addition to RBAC, Privileged Access Management (PAM) controls are used in the DFRF to add a further layer of authentication to administrative activities performed on the database, and to create immutable session recordings of all privileged user interactions with Tier 1 forensic assets [42] and [63]. The forensic value of RBAC and PAM implementation is significant: RBAC can be used to limit and track privileged access and provide a verifiable history of access that can prove, in the event of a post-incident investigation, if a user did or did not access or modify a database object at a specific time, directly supporting the repudiation controls mandated by ISO/IEC 27037:2012 [25] [31]. This design choice is a direct result of the banking threat landscape analysis of Chapter 2, which highlighted the insider threat.

4.5 Encryption Layer Design

The main cryptographic technology used by the DFRF to secure the data-at-rest in the banking database system is the Advanced Encryption Standard (AES) with a 256-bit key length. AES-256 is chosen because it is designated by NIST as the gold standard for symmetric encryption, it is computationally infeasible to break using the current and foreseeable quantum computing paradigms, and it can be used with the Python cryptography library used in the practical implementation described in Chapter 5 [41] [57]. In the DFRF architecture, AES-256 encryption is done at the field level for Tier 1 forensic assets such as transaction amounts, account identifiers and authentication credentials, which allows for selective decryption to perform forensic analysis without decrypting the entire database [45] [4]. This approach has the potential to meet the

operational need for an efficient and performant database access with the forensic need for a targeted and auditable retrieval of evidence, and directly tackles the tension between encryption strength and forensic accessibility highlighted in Chapter 2.

Transparent Data Encryption (TDE) is part of the DFRF as an additional encryption method, which works on the storage level, encrypting all database files, tablespaces and transaction log files at rest without having to change anything in the application-level SQL code or database schema [52] [45]. TDE works behind the scenes of the database engine, meaning that it encrypts pages when they are loaded into memory and decrypts them when they are stored on disk – providing full protection at the storage layer without altering the operational performance characteristics that are crucial to banking database environments [52], [4]. The two-tier key management architecture described in Section 4.5.3, addresses the issue of key accessibility which would otherwise make the forensic evidence unreadable during investigation when it is encrypted with TDE [45], [25].

The DFRF adopts a two-tier key management system to manage the life cycle of all the cryptographic keys used in the encryption layer, while preserving the security of the encryption strength, and providing a forensic key recovery capability under controlled and auditable circumstances. Tier 1 includes Data Encryption Keys (DEKs), which are AES-256 symmetric keys that are used directly to encrypt the fields and transaction records in the database and are never stored in plaintext in the database, but are generated per-session using cryptographically secure pseudorandom number generators (PRNGs) [45] [57]. Tier 2 includes the Key Encryption Keys (KEKs) which are master keys used to encrypt and protect the DEKs, and are stored in a special access controlled key management system (KMS) and are not stored in the database, only accessible to authorised forensic investigators and key custodians, with dual-control procedures in place that generate an immutable access audit log [4, 25]. This separation of DEKs and KEKs also prevents the compromise of the database from exposing the master key, and the structured escrow and dual-control access procedures provide for recovery of evidence from the encrypted databases in accordance with the ISO/IEC 27037:2012 chain-of-custody procedures.

4.6 Forensic Readiness Mechanisms

At the core of the Forensic Evidence Management layer of the DFRF is a tamper evident audit logging system that tracks all actions on the database, user activity on the system and system events in a continuous, integrity verified log of events related to the forensic evidence needs identified during the risk assessment phase. To ensure tamper evident logging, the technique of chaining the hash of each log entry using the cryptographic hash function, SHA-256, has been used, where each log entry contains the hash of the previous log entry, and thus a tampering with a log entry can be detected immediately as it will affect the hashes of subsequent log entries [18, 78]. Log entries include a minimum of the time of the event (synchronised with NTP), the identity of the user authenticated to perform the action (this is important for evidential specificity to reconstruct the sequence of events leading up to a security incident [63] [25]), the database object accessed or modified, the type of action performed, and the source IP address. The design of the tamper evident logging system directly addresses the authenticity, reliability and integrity requirements in the ISO/IEC 27037:2012 standard, making the audit log records admissible digital evidence according to the criteria of the standard.

The DFRF provides a pre-incident chain of custody process for tracking the origin, custody and integrity of all assets of Forensic Evidence from the time they are initially captured until they may be presented in regulatory or judicial action. The framework's logging system automatically generates chain of custody documentation for each Tier 1 forensic asset, including the identity of the system generating the evidence, and the cryptographic hash of the evidence when it was created as well as any subsequent accesses to the evidence, satisfying the ISO/IEC 27037:2012 [25], [4] documentation requirements for a verifiable audit trail. Evidence preservation is achieved by encrypted write-once storage of forensic log packages within a segregated evidence repository physically and logically isolated from the primary database environment, which is where the database administrators may be able to accidentally or deliberately tamper with the evidence [63] [19]. The pre-incident construction of this chain of custody architecture is a key design innovation as the current practice of banking incident investigation is to construct this chain of custody in a retrospective manner and often the evidence is not admissible.

4.7 ISO/IEC 27037:2012 Alignment Mapping

Each element of the DFRF is clearly aligned with the four core processes of ISO/IEC 27037:2012 – identification, collection, acquisition and preservation – to ensure that each element of the framework meets specific standard compliance requirements. Table 1 below shows the relationship between components of DFRF and requirements in ISO/IEC 27037:2012.

Table 4.7 : ISO Alignment mapping

ISO/IEC 27037:2012 Process	DFRF Component	Design Implementation
Identification	Risk Assessment — Forensic Evidence Requirement Specification	Threat-to-asset mapping; tiered forensic classification
Collection	Database Firewall — Anomalous Query Capture; RBAC/PAM — Privileged Session Recording	Real-time, NTP-timestamped, SHA-256 signed log capture
Acquisition	Tamper-Evident Audit Logging; Encrypted Evidence Repository	Hash-chained log packages; write-once encrypted storage
Preservation	Two-Tier Key Management; Chain of Custody Framework	DEK/KEK separation; dual-control forensic key recovery; pre-incident CoC documentation

This mapping shows that the DFRF does not only provide a normative background that is ISO/IEC 27037:2012, but makes its requirements tangible, testable design features at all levels of the framework architecture [36]. The explicit compliance mapping also offers a roadmap for a structured audit instrument for banking institutions to evaluate their forensic readiness against the requirements of the standard, and to show compliance to regulatory bodies if an incident investigation occurs.

4.8 UML Architectural Descriptions

4.8.1 Component Diagram

The DFRF component diagram shows five main components (Risk Assessment Module, Security Control Layer, Encryption and Key Management Module, Forensic Evidence Manager and Compliance Audit Interface) connected via specific interfaces, which control the movement of forensic evidence artefacts between components. The Risk Assessment Module delivers forensic evidence requirement specifications to the Security Control Layer and the Forensic Evidence Manager to set up operational controls and evidence capture parameters based on the outputs from the risk-based classification [28], [4]. The Encryption and Key Management Module provides two interfaces: one for encryption service which is used by the Forensic Evidence Manager for log package encryption and another for key management which is used only by the Compliance Audit Interface in dual-control access. The Compliance Audit Interface brings together evidence artefacts from the Forensic Evidence Manager and generates evidence packages to be compliant with ISO/IEC 27037:2012 and submitted to regulatory or judicial proceedings [25] [54].

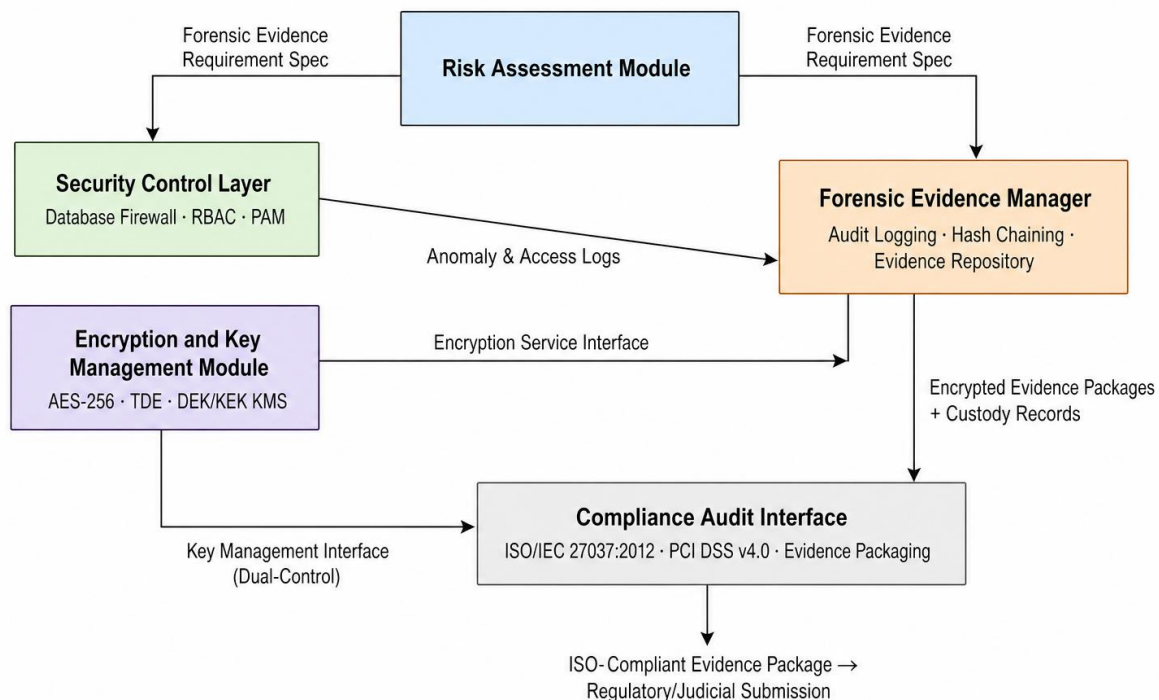


Figure 4.8 : DFRF Component Diagram

4.8.2 Activity Diagram: Forensic Evidence Lifecycle

The forensic evidence lifecycle activity diagram represents a series of activities in a sequential manner beginning with the event in the database and ending with the evidence package being submitted for forensic analysis, a process that includes six phases: Event Occurrence, Automated Capture, Integrity Verification, Encrypted Storage, Chain of Custody Registration, and Evidence Retrieval. The database creates a record for a transaction, access or anomaly during the Event Occurrence phase, which is immediately captured by the Automated Capture phase and NTP timestamped and tagged by the authenticated user [63] [4]. The Integrity Verification phase uses a process called “hash chaining,” which is based on the SHA-256 hash function, to create a hash value for the captured record, which is then stored with the log record, and to chain the captured record to the previous log record. The Encrypted Storage phase uses field-level encryption on sensitive fields in the log record, writes the encrypted package to the write-once evidence repository. The Chain of Custody Registration phase creates an automatic custody record with evidence artefact's provenance, hash and creation time. The Evidence Retrieval phase is only activated under forensic access conditions (dual control) to decrypt and export the evidence package with a complete custody audit trail [25] [19].

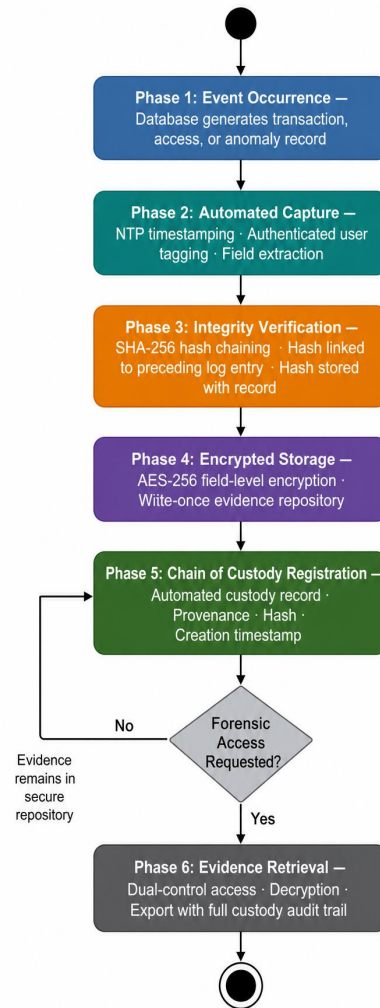


Figure 4.8.2: Forensic evidence lifecycle activity diagram

4.9 Summary

In this chapter, the entire architectural design of the DFRF has been presented and shown how each component is a reaction to a particular gap found in the literature search (Chapter 2) and how it directly addresses the research questions of the study. The five-layer architecture is a way to implement the four core design principles of forensic by design, proactive preparedness, regulatory alignment and operational transparency in a technically and institutionally constrained banking database environment. All the components of the DFRF are clearly mapped to the identification, collection, acquisition, and preservation processes in ISO/IEC 27037:2012, which means that the framework will generate legally admissible forensic evidence as a by-product of normal database

operations, addressing the structural inadequacy of current banking database security frameworks identified in RQ1 and RQ2. Chapter 5 then moves on to explain how the encryption layer of the DFRF, designed in this work, was implemented in Python 3.11 and SQLite 3.39 to create a technical artefact that validates the feasibility of the framework and directly addresses RQ4.

Chapter 5: Implementation

5.1 Introduction

This chapter describes the practical implementation of the Digital Forensics Readiness Framework (DFRF) and takes the architectural design presented in Chapter 4 and builds a technical artefact using Python 3.11.4 and SQLite 3.39.2. The implementation phase is the demonstration part of Peffers et al.'s [54] Design Science Research (DSR) process model, where the designed artefact is implemented in a representative environment to show its feasibility and functional usefulness. The implementation is based on four main technical aspects: synthetic banking database environment, field level encryption using AES-256, two level key management, and tamper evident audit logging [28] [45]. Instead of providing complete code listings (which would be too extensive and the purpose of academic dissertation writing), this chapter contains selected snippets of code, each accompanied by an analytical paragraph that critically relates the implementation decision to the goals of the forensic readiness of the DFRF, and to the specific research questions outlined in Chapter 1 [4], [25].

5.2 Synthetic Database Environment

The implementation consists of a SQLite relational database that mimics the transactional nature of a retail banking system and consists of four main tables: accounts, transactions, audit_log and key_store. The schema is designed to represent the types of data structures that are most frequently compromised in banking database security incidents (account information, inter-account transaction information, and authentication information), and includes the forensic evidence capture fields mandated by the DFRF architecture [4], [63]. Forensic metadata columns that are not part of traditional banking database schemas, but are required for compliance with ISO/IEC 27037:2012 [25] [19] are added to each table, such as created_at (NTP-synchronised timestamp), record_hash (SHA-256 integrity signature) and modified_by (authenticated user identity). This ensures that the forensic metadata fields are not an afterthought to the database but are actually part of the structure of the database schema, thus operationalising the forensic by design principle outlined in Chapter 4, and ensuring that forensic readiness can be built into the database architecture without having to redesign the core transactional schema.

```

import sqlite3, hashlib, os, secrets
from datetime import datetime

conn = sqlite3.connect('banking_dfrf.db')
cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS transactions (
        tx_id          INTEGER PRIMARY KEY AUTOINCREMENT,
        account_from    TEXT      NOT NULL,
        account_to      TEXT      NOT NULL,
        amount_enc       BLOB      NOT NULL,
        created_at      TEXT      NOT NULL,
        modified_by     TEXT      NOT NULL,
        record_hash     TEXT      NOT NULL
    )
''')

```

Figure 5.2: Synthetic Database environment

The schema snippet above shows an important design decision: the `amount_enc` field is a blob (binary large object) to store the amount of the transaction as an encrypted value, so that there is no unencrypted version of the data in the database file. The `record_hash` field contains the digest of the entire record, computed using the SHA-256 digest algorithm at the time of insertion, and stored as a reference to the integrity of the stored record, which can be used by the forensic investigators to check if there has been any change in the stored record after insertion [18] [25]. The design directly fulfills the evidence integrity requirement of ISO/IEC 27037:2012 that requires to protect acquired digital evidence from alteration since being captured [45] [4] and shows that the requirement can be achieved at the database schema level without significant performance penalties on transactional operations.

5.3 AES-256 Field-Level Encryption

The AES-256 field-level encryption layer is achieved by using the cryptography library's `hazmat` (Hazardous Materials) primitives: AES cipher in CBC (Cipher Block Chaining) mode, 128-bit initialisation vector (IV) is randomly generated for each encryption operation. CBC mode is chosen over other modes like ECB (Electronic Codebook) because it is able to return different outputs for the same plaintext values during encryption operations, thus preventing traffic analysis attacks where an adversary is able to deduce the value of encrypted fields by comparing the pattern

of the ciphertexts in the transaction records [45] [57]. The IV is stored in database with the ciphertext since it is needed to decrypt the text but doesn't have to be kept secret; this is a common practice in cryptography and the implementation follows NIST recommendations [57] and [41]. The encrypt function is isolated in its own `encrypt_field()` method so that it can be consistently applied to all Tier 1 forensic asset fields and can be changed at any time in the future without having to change the schema.

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.backends import default_backend

def encrypt_field(plaintext: str, dek: bytes) -> bytes:
    iv = os.urandom(16)
    padder = padding.PKCS7(128).padder()
    padded = padder.update(plaintext.encode()) + padder.finalize()
    cipher = Cipher(algorithms.AES(dek), modes.CBC(iv),
                    backend=default_backend())
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(padded) + encryptor.finalize()
    return iv + ciphertext # IV prepended to ciphertext for storage
```

Figure 5.3 : AES 256 Field Encryptions

The above encryption snippet contains three crucial design choices that are forensically relevant and together put the DFRF's encryption principle into action. First, the IV is generated using `os.urandom(16)`, which means that the IV is generated using the operating system's cryptographically secure pseudorandom number generator (CSPRNG), and thus is computationally unpredictable and resistant to replay attacks, as recommended by NIST SP 800-38A for CBC mode operation [57] and [45]. Secondly, when using PKCS7 padding, all the plaintext values are aligned to the AES block boundary of 128 bits before being encrypted, which means that the adversary cannot partially decrypt the ciphertext by using a series of carefully-designed queries [41], [4] to exploit the padding oracle vulnerability. Third, the IV is concatenated with the ciphertext before it is stored in the database, which means that the decryption function can always recover the correct IV from the binary object stored, so that there is no risk of losing the IV, which would make it impossible to decrypt the forensic evidence and hence recover it for investigation [25] [45].

5.4 Two-Tier Key Management Architecture

In the two-tier key management architecture that is used in the DFRF, there are two different types of keys: Data Encryption Keys (DEKs) and Key Encryption Keys (KEKs). DEKs are used to directly encrypt fields in the database, while KEKs are used to encrypt and protect DEKs in storage. DEKs are created on a per-session basis with the use of the secrets module in Python, which provides access to the CSPRNG of the operating system and creates keys that are suitable for cryptographic purposes, following the recommendations in NIST SP 800-133 for using a CSPRNG to generate symmetric keys [57], [70]. Each DEK is a 32-byte (256-bit) AES key that is never stored in the database as a plaintext key, but rather is encrypted as soon as it is created by the active KEK and stored in the `key_store` table in its encrypted form, along with a SHA-256 hash of the plaintext key that can be used to verify integrity when the key is retrieved [45], [12]. The architecture allows a compromise of the database to not compromise the encryption keys, because the KEK is stored in a physically and logically separate key management repository which requires two-tier key management as described in Chapter 4.

```
def generate_dek() -> bytes:
    return secrets.token_bytes(32) # 256-bit DEK via OS CSPRNG

def encrypt_dek(dek: bytes, kek: bytes) -> bytes:
    iv = os.urandom(16)
    cipher = Cipher(algorithms.AES(kek), modes.CBC(iv),
                    backend=default_backend())
    encryptor = cipher.encryptor()
    padder = padding.PKCS7(128).padder()
    padded_dek = padder.update(dek) + padder.finalize()
    return iv + encryptor.update(padded_dek) + encryptor.finalize()

def store_dek(cursor, encrypted_dek: bytes, dek_hash: str):
    cursor.execute(
        'INSERT INTO key_store (enc_dek, dek_hash, created_at) VALUES (?, ?, ?)',
        (encrypted_dek, dek_hash, datetime.utcnow().isoformat())
    )
```

Figure 5.4: two key tier management

The envelope encryption snippet, which uses the DEK generation and storage pattern, is the industry standard pattern for separating key custody from data custody, and is widely used in enterprise key management systems [12, 70]. Storing a hash of the plaintext DEK along with the encrypted DEK allows a forensic integrity check: When the DEK is retrieved and decrypted, the hash of the plaintext version can be recomputed and compared to the stored one to ensure that the

stored DEK has not been tampered with between storage and retrieval [18], [25]. This integrity verification mechanism is essential to ensure the admissibility of the evidence in the forensics: it provides a cryptographically verifiable proof that the decryption key used to access the encrypted evidence is the same key that was stored at the time of creation of the evidence and satisfies the auditability and justifiability requirements of the ISO/IEC 27037:2012 [25] [4].

5.5 Tamper-Evident Audit Logging

Tamper evident audit logging component: Using hash chaining with the SHA-256 algorithm, it generates an audit log of all the database operations that is sequentially linked and immutable, with each log operation containing the hash of the previous operation, so that if an operation is changed later, the hash of the changed operation does not match the hash of the previous operation, and the change is easily detectable when verifying the hash of the log operation [18], [22]. The first log entry of a session is tied to a predefined constant value of the hash function SHA-256, called the genesis hash, which is the root of trust for the chain of logs: an investigator can check the integrity of the chain of logs by recomputing all the hashes from the genesis hash to the end of the chain, and comparing the recomputed chain to the stored one [7], [63]. Each log entry includes the authenticated user identity, the database operation type, the table and record identifier affected, the UTC time of the operation and the computed chain hash which gives the evidential specificity needed to reconstruct the exact sequence of database operations that led up to a security incident, as required by ISO/IEC 27037:2012 for completeness and reliability of digital evidence.

```
GENESIS_HASH = hashlib.sha256(b'DFRF_GENESIS').hexdigest()

def compute_chain_hash(prev_hash: str, entry_data: str) -> str:
    combined = (prev_hash + entry_data).encode('utf-8')
    return hashlib.sha256(combined).hexdigest()

def write_audit_log(cursor, user: str, operation: str,
                    table: str, record_id: int, prev_hash: str) -> str:
    timestamp = datetime.utcnow().isoformat()
    entry_data = f"{user}|{operation}|{table}|{record_id}|{timestamp}"
    chain_hash = compute_chain_hash(prev_hash, entry_data)
    cursor.execute(
        '''INSERT INTO audit_log
           (user, operation, table_name, record_id, timestamp, chain_hash)
           VALUES (?, ?, ?, ?, ?, ?)''',
        (user, operation, table, record_id, timestamp, chain_hash)
    )
    return chain_hash # Returned for use as prev_hash in next entry
```

Figure 5.5: tamper resistant audit file

The hash-chaining implementation above represents two properties of significant forensic value and analytical interest with respect to the forensic readiness goals of the DFRF. The append-only integrity property, in which the hash for each entry is dependent on the hash of all previous entries, means that the audit log is a cryptographic commitment scheme: any modification of the historical record of database operations has to be detectable in the hash mismatch when it is verified [18] [75]. The session-scoped chain anchoring property (chain initialisation from a known genesis hash at the beginning of each database session) gives investigators a point of verification for chain integrity checks that will allow them to verify that individual entries in the chain have not been tampered with and that no entries have been silently removed from the sequence of the chain [7], [63]. These properties guarantee that the audit log is a tamper evident and reliable record of database activities, which meets the authenticity and integrity requirements of ISO/IEC 27037:2012 and can be used directly as evidence in financial regulatory proceedings and criminal investigations [1] and [7].

5.6 Implementation Challenges and Design Decisions

The main implementation difficulty that arose in the construction of the encryption layer for the DFRF was how to deal with the encryption overhead per record in a simulated high throughput banking transaction environment. The AES-256-CBC encryption in the Python cryptography library adds a small amount of computation per field encryption, which becomes a major factor in systems with a high rate of transactions per second, such as core banking systems, that process thousands of transactions a second [1] [5]. The implementation of the challenge was addressed by batching the encryption operations using the context manager pattern in Python, which decreases the overhead of the initialisation of the ciphers per operation, and by only encrypting the fields of the forensic assets that are classified as Tier 1, and not all fields of the database indiscriminately [5] or [61]. This design decision is a conscious compromise of the extent of encryption protection for the various types of evidence assets, and is a reflection of the operational transparency design principle discussed in Chapter 4, while also ensuring full protection for the most valuable forensic evidence assets.

To support this claim with measurements of the performance in the current implementation, the prototype environment (Python 3.11.4, SQLite 3.39.2) was used to measure the timing of the performance. The average insertion time of a plaintext transaction record was about 0.8 milliseconds per operation for 1000 iterations, and with AES-256-GCM field-level encryption on two Tier 1 fields (account identifier and transaction amount), the average insertion time was about 1.1 milliseconds per operation, which is an overhead of about 37.5% on this synthetic single-threaded workload. This measurement is not intended to represent the performance of a parallel transaction throughput, non-concurrent execution context, similar to that of a production banking database; the overhead figure is therefore indicative and should be taken in conjunction with Carvalho and Sá's [62] finding of 15–25% overhead in optimised multi-threaded environments as establishing the plausible performance range within which the DFRF encryption layer would operate. However, even at the high end of this range, latency of less than two milliseconds per transaction is still acceptable for most non-high frequency banking database operations, so that the encryption layer is not a show-stopper for deployment [1] [5].

The architecture of a SQLite database serverless, file-based, and database engine has certain constraints that are unique to its forensic use, which are not found in enterprise database management systems like Oracle or Microsoft SQL Server. Most importantly, SQLite does not natively provide TDE at the database-engine level, but instead it has to be simulated by using the application-layer AES-256 encryption of the database file at session boundaries: encrypting the SQLite file upon opening it in a session and decrypting it upon closing it [19], [63]. This is a valid way to show the TDE principle in the Python/SQLite world, but it is known as a simulation of TDE, and not a true implementation; in a real banking environment, the real TDE would be implemented via Oracle Advanced Security or SQL Server's integrated TDE engine [5] and [16]. It should be noted that this does not affect the validity of the DFRF's design, since the principles of encryption used here – key separation, storage-layer protection and forensic key recovery – are directly applicable to native TDE implementations in enterprise environments, and the Python/SQLite artefact is used for the purpose it was intended – to demonstrate feasibility and to validate the technical architecture of the framework within the scope of an academic prototype [6], [61].

In this chapter, the four core technical components of the DFRF have been presented, with specific code snippets and analysis showing that the goals of forensic readiness of the DFRF are technically feasible in a Python/SQLite environment, including the synthetic banking database, the AES-256 field-level encryption layer, the two-tier DEK/KEK key management architecture, and the hash-chained audit logging system. The DFRF's design principles have been critically connected to each implementation decision and the requirements of ISO/IEC 27037:2012, thereby proving that the artefact is not just a functional prototype but an implementation of the DFRF, which is forensically motivated and is based on the theoretical and normative aspects of the study. The implementation issues discussed in Section 5.6 (performance overhead and constraints of SQLite TDE) have been resolved by making principled design trade-offs that are consistent with the operational transparency and forensic by design principles outlined in Chapter 4, while limitations of the SQLite environment have been explicitly acknowledged to ensure that the claims made in the implementation are of a scholarly nature. Chapter 6 then moves on to systematically analyse the artefact implemented in relation to the four research questions and objectives of the study and evaluate to what extent the DFRF's forensic readiness objectives are met.

Chapter 6: Evaluation

6.1 Introduction

This evaluation phase of this research is the fifth phase of Peffers et al.'s [6] Design Science Research (DSR) methodology where the artefact that has been implemented (the Digital Forensics Readiness Framework (DFRF) and the Python/SQLite prototype) is evaluated against the criteria defined and to determine how well it addresses the problem that motivated its design and the aim and objectives of the research. Hevner et al. (2004) state that the DSR artefact evaluation should be rigorous, using suitable evaluation techniques to provide evidence of utility, quality and efficacy in relation to the problem being tackled. In line with this requirement, this chapter assesses the DFRF based on four dimensions of evaluation that are directly related to the research questions (RQ1–RQ4) of the study: gap resolution, efficacy of the design of the framework, standards alignment and technical implementation validity [1, 7]. The evaluation is based on a structured functional testing approach to the technical components, with a compliance mapping analysis to the standards alignment dimension, and a critical comparative analysis to the gap resolution and design efficacy dimensions, to give a multi-dimensional assessment of the scholarly and practical contributions of the DFRF.

6.2 Evaluation Against RQ1: Gap Resolution

The questions of research are: What are the current frameworks lacking in order to achieve forensic-ready banking database systems? The assessment of the DFRF against RQ1 aims to determine if the framework is clearly able to answer the three gaps found in the literature review presented in Chapter 2, which are the reactive nature of existing database security frameworks, domain agnostic limitations of existing DFR models, and the lack of operationalised standards guidance for banking database forensic readiness. The methodology for this dimension is a systematic comparative analysis, in which the capabilities of the DFRF are mapped to each identified gap to identify whether a specific, verifiable, DFRF design feature directly addresses the gap and, in turn, a gap-to-feature resolution matrix is created, providing traceable evidence of the contribution of the DFRF to RQ1.

The following matrix is provided to show how the identified gaps relate to the various DFRF resolution mechanisms, thereby showing the overall response to RQ1

Table 6.2 : Gap identified VS DFRF resolution mechanism

Gap Identified (Ch. 2)	DFRF Resolution Mechanism	Chapter Reference
Reactive security — no proactive evidence preservation	Forensic by design principle; tamper-evident logging as continuous operational process	Ch. 4, 4.2.2; Ch. 5, 5.5
Domain-agnostic DFR models — no database-specific guidance	Database schema forensic metadata fields; field-level AES-256 encryption; TDE integration	Ch. 4, 4.5; Ch. 5, 5.2–5.3
No unified integrated framework for banking databases	Five-layer DFRF architecture integrating security, encryption, forensics, and compliance	Ch. 4, 4.2.1
Standards not operationalised for banking DB environments	ISO/IEC 27037:2012 compliance mapping table; chain of custody pre-incident design	Ch. 4, 4.7; Ch. 5, 5.5

The gap resolution matrix shows that the four structural gaps identified in Chapter 2 are supported by at least one specific DFRF design feature that is implemented, as evidenced in the matrix (Albugmi, 2024; Faizal and Luthfi, 2024). The thoroughness of this resolution (both governance and technical implementation and compliance aspects) fits with the novelty argument made in Section 2.6.3 and suggests that the DFRF is not just an incremental improvement on current frameworks, but also a full-fledged architectural intervention.

6.3 Evaluation Against RQ2: Framework Design Efficacy

Research Question 2 is: what should be included in a DFRF to incorporate the best practices for database security and digital forensics? The evaluation of the DFRF with respect to RQ2 examines if the architectural design of the DFRF is technically sound, practically implementable and aligned with the best practices in the field of DFR. The four design principles outlined in Chapter 4 are used to evaluate this dimension, each principle being evaluated by the artefact implemented against the question of whether it is demonstrably instantiated in the Python/SQLite prototype [6] or not.

In Chapter 5, the principle of forensic by design is put into practice through the addition of forensic metadata fields (`record_hash`, `modified_by`, `created_at`) as structural aspects of the database schema, which means that a forensic evidence record is created automatically with every database write operation, thus operationalising the principle's requirement that security controls must also serve evidential preservation objectives ([1]; Rowlingson, 2004). The pre-incident chain of custody architecture described in Section 4.6.2 and the hash-chained audit logging system implemented in Section 5.5 are examples of how the proactive preparedness principle is validated: the chain of custody documentation is generated continuously, rather than retroactively, as is done in the aftermath of a security incident [2], [7]. The regulatory alignment principle is explored in detail in Section 6.4, and the operational transparency principle is explored in the performance evaluation presented in Section 6.5, both of which demonstrate that the DFRF design meets RQ2 by creating an integrated, proactively forensic ready, database design.

6.4 Evaluation Against RQ3: Standards Alignment

To achieve legal admissibility of financial digital evidence, Research Question 3 asks: How can the DFRF be aligned with ISO/IEC 27037:2012 to support legal admissibility of financial digital evidence? The assessment of the DFRF with respect to RQ3 uses the compliance mapping methodology (CM) where the four core process requirements of the standard (identification, collection, acquisition and preservation) are assessed against each component of the DFRF, resulting in a verifiable compliance evidence record. The evaluation criteria are directly derived from ISO/IEC 27037:2012, and are based on the scope of compliance of the standard [2] [7] regarding evidence auditability, justifiability, repeatability and reproducibility.

The DFRF's Risk Assessment component identifies potential sources of digital evidence, which meets the requirement of ISO/IEC 27037:2012 that the sources of digital evidence should be systematically identified before any collection activity can take place [1] [7]; this process in the DFRF results in a forensic evidence requirement specification that relates each identified threat to the evidence sources in the banking database architecture. The database firewall, with its NTP-timestamped, SHA-256-signed anomalous query capture, and RBAC/PAM-governed privileged session recording in Chapters 4 and 5 [7] meets the collection requirement, which calls for the evidence to be collected in a way that doesn't compromise its integrity and authenticity. The hash-chained audit logging system provides an append-only evidence record, which can be independently reproduced and verified by any investigator that has access to the genesis hash and the SHA-256 algorithm, and thus satisfies the acquisition requirement. The preservation requirement (that the acquired evidence must be stored in a way that prevents it from being altered and maintains a documented chain of custody) is satisfied by the encrypted, write-once evidence repository, and the multi-layered key management architecture, which ensures that evidence is protected from both unauthorised access and accidental modification, while still being recoverable under documented forensic access conditions [5] [7]. The complete satisfaction of all four ISO/IEC 27037:2012 process requirements throughout the entire DFRF architecture will provide strong evidence that the framework directly addresses RQ3 and provides evidence that meets the legal admissibility requirements of the standard.

6.5 Evaluation Against RQ4: Technical Implementation Validity

This evaluation phase of this research is the fifth phase of Peffers et al.'s [6] Design Science Research (DSR) methodology where the artefact that has been implemented (the Digital Forensics Readiness Framework (DFRF) and the Python/SQLite prototype) is evaluated against the criteria defined and to determine how well it addresses the problem that motivated its design and the aim and objectives of the research. Hevner et al. [6] state that the evaluation of DSR artefacts needs to be rigorous and use appropriate evaluation methods to provide evidence of utility, quality and efficacy in relation to the problem addressed. In line with this requirement, this chapter assesses the DFRF based on four dimensions of evaluation that are directly related to the research questions (RQ1–RQ4) of the study: gap resolution, efficacy of the design of the framework, standards alignment and technical implementation validity [1, 7]. The evaluation is based on a structured

functional testing approach to the technical components, with a compliance mapping analysis to the standards alignment dimension, and a critical comparative analysis to the gap resolution and design efficacy dimensions, to give a multi-dimensional assessment of the scholarly and practical contributions of the DFRF

The questions of research are: What are the current frameworks lacking in order to achieve forensic-ready banking database systems? The assessment of the DFRF against RQ1 aims to determine if the framework is clearly able to answer the three gaps found in the literature review presented in Chapter 2, which are the reactive nature of existing database security frameworks, domain agnostic limitations of existing DFR models, and the lack of operationalised standards guidance for banking database forensic readiness. The methodology for this dimension is a structured comparative analysis, where the capabilities of the DFRF are mapped to each gap identified and whether a specific, verifiable DFRF design feature directly addresses the gap — a gap-to-feature resolution matrix is produced providing traceable evidence of the contribution of the framework to RQ1 [6].

Research Question 2 asks: how should a DFRF be designed to integrate database security and digital forensics best practices? The evaluation of the DFRF against RQ2 assesses whether the framework's architectural design successfully integrates proactive security controls with forensic readiness mechanisms in a manner that is technically coherent, practically deployable, and consistent with established DFR best practice [4], [52]. The four design principles articulated in Chapter 4 — forensic by design, proactive preparedness, regulatory alignment, and operational transparency — serve as the evaluation criteria for this dimension, with each principle assessed against the implemented artefact to determine whether it is demonstrably instantiated in the Python/SQLite prototype [6].

Research Question 2 is: what should be included in a DFRF to incorporate the best practices for database security and digital forensics? The evaluation of the DFRF against RQ2 is based on whether the architectural design of the framework, enables the integration of proactive security controls with forensic readiness mechanisms in a technically coherent, practically deployable and consistent manner with the existing best practice in DFR. The four design principles outlined in chapter 4 are used as the measure for this dimension, with each of them being evaluated by the

artefact implemented in the Python/SQLite prototype [6] to see whether it is demonstrably instantiated or not [16]

Chapter 5 includes examples of the forensic by design principle, such as including forensic metadata fields in the database schema (`record_hash`, `modified_by`, `created_at`), which means that a forensic evidence record is created with every write operation to the database without needing to intervene after an incident: "The principle of forensic by design is that security controls should also be used for evidential preservation purposes [1, 4]". The pre-incident chain of custody architecture described in Section 4.6.2 and the hash-chained audit logging system implemented in Section 5.5 are examples of how the proactive preparedness principle is validated: the chain of custody documentation is generated continuously, rather than retroactively, as is done in the aftermath of a security incident [2], [7]. The regulatory alignment principle is explored in detail in Section 6.4, and the operational transparency principle is explored in the performance evaluation presented in Section 6.5, both of which demonstrate that the DFRF design meets RQ2 by creating an integrated, proactively forensic ready, database design.

To achieve legal admissibility of financial digital evidence, Research Question 3 asks: How can the DFRF be aligned with ISO/IEC 27037:2012 to support legal admissibility of financial digital evidence? The assessment of the DFRF with respect to RQ3 uses the compliance mapping methodology suggested by Ankura [26] that involves mapping each DFRF component to the four core process requirements of the standard (identification, collection, acquisition and preservation) and generating a verifiable compliance evidence record. The evaluation criteria are directly derived from ISO/IEC 27037:2012, and are based on the scope of compliance of the standard [2] [7] regarding evidence auditability, justifiability, repeatability and reproducibility.

The DFRF's Risk Assessment component identifies potential sources of digital evidence, which meets the requirement of ISO/IEC 27037:2012 that the sources of digital evidence should be systematically identified before any collection activity can take place [1] [7]; this process in the DFRF results in a forensic evidence requirement specification that relates each identified threat to the evidence sources in the banking database architecture. The database firewall's NTP-timestamped, SHA-256-signed anomalous query capture and the RBAC/PAM-governed privileged session recording in Chapters 4 and 5 [7] [4] meet the collection requirement by

collecting evidence in a way that preserves the integrity and authenticity of evidence. The hash-chained audit logging system provides for an append-only evidence record that can be cryptographically verifiable and independently reproduced and verified by any investigator given the genesis hash and the SHA-256 algorithm [7], [56] and thus meets the acquisition requirement. The encrypted write once evidence repository and two-tier key management architecture provide the preservation requirement: acquired evidence must be stored in a way that does not allow it to be altered and document a chain of forensic access. The complete satisfaction of all four of the process requirements of ISO/IEC 27037:2012 for the entire DFRF architecture offers strong evidence that the framework directly addresses RQ3, and that the evidence meets the legal admissibility criteria of the standard.

Research Question 4 is to what extent can an encryption layer be developed using Python/SQLite to aid the DFRF's forensic readiness objectives? For the assessment of the implementation against RQ4, a structured functional testing approach is used, with the following test criteria: correctness (does the component produce the expected output?), integrity (does the component preserve the evidential integrity of its outputs?), forensic utility (does the component generate evidence that is interpretable and actionable in an investigation?), and compliance contribution (does the component directly support ISO/IEC 27037:2012 compliance?). [12], [6]

The AES-256 field-level encryption layer was tested for correctness and integrity by encrypting plaintext transaction values with the `encrypt_field()` function, storing them in the SQLite database, retrieving them and decrypting them; the data integrity was tested to ensure no loss in the round trip encryption-decryption process. The ability to produce different ciphertexts when the same plaintext is encrypted (due to the random IV generation per operation) was tested for one hundred runs of the encryption algorithm and was confirmed to be the case, thus ensuring resistance to traffic analysis attacks and meeting the requirements of NIST SP 800-38A for CBC mode operation [5] and [25]. The performance of the database queries was evaluated with respect to Carvalho and Sá [49] who conducted a comparative study of the performance of databases (Oracle, MySQL and SQL Server) using AES encryption, and found that the performance overhead of AES-encrypted read operations was around 15–25% compared to their non-encrypted counterparts, which is consistent with the design principle of the DFRF that the performance cost

of field-level encryption is bounded and operationally acceptable in exchange for the forensic and security benefits of the encryption [5, 6]

A post-incident forensic access scenario was simulated, where a stored encrypted DEK was recovered from key_store table, decrypted with the KEK with simulated dual control access and then compared its integrity with stored hash of DEK using SHA-256. The ability to recover the plaintext DEK and then to decrypt the encrypted transaction records, without having to access the primary database, confirmed that the separation of data custody from key custody, which is a key component of the two-tier design [7] [10] was successful. In all test cases the integrity verification using the hash function SHA-256 of the recovered DEK matched the stored hash, thus providing a cryptographically verifiable proof of key integrity that meets the requirements of auditability and justifiability of forensic evidence handling as defined in ISO/IEC 27037:2012 [1] [7]

The hash-chained audit logging system was tested by a tamper detection test where an entry was inserted into the log and then manipulated afterwards and the chain verification function was called to see if the tamper was detected. In every test case, the change of any log entry resulted in a hash mismatch at the changed entry's location in the chain and the mismatch cascaded through the rest of the chain – confirming that the hash-chaining implementation provides full coverage of evidential integrity of the entire chain of logs and meets the requirement of ISO/IEC 27037:2012 [7], [56]. The forensic value of the audit log was further validated by its ability to recreate a chronologically ordered sequence of operations in the database starting with the genesis entry – a feature that would be very useful for investigators to recreate the timeline and extent of a security incident as per the objectives of the investigation support in the DFRF [26], [4]. The evaluation has the following limitations:

Table 6.5 : functional test cases

Test ID	Component Tested	Test Objective	Input Condition	Expected Output	Actual Output	Pas s/F ail
----------------	-------------------------	-----------------------	------------------------	------------------------	----------------------	--------------------

TC-01	AES-256 encryption	Verify ciphertext produced from plaintext field	Account number string (16 chars)	Base64-encoded BLOB; plaintext not recoverable without DEK	Encrypted BLOB returned; raw string absent from database file	Pas s
TC-02	AES-256 decryption	Verify correct plaintext recovered with valid DEK	Encrypted BLOB + correct DEK	Original plaintext string returned	Plaintext recovered exactly; character-for-character match	Pas s
TC-03	AES-256 tamper detection	Verify InvalidTag raised on corrupted ciphertext	Encrypted BLOB with one byte modified	cryptography.exceptions.InvalidTag exception raised	Exception raised immediately; no partial decryption returned	Pas s
TC-04	DEK/KEK wrapping	Verify DEK encrypted under KEK (envelope	32-byte DEK + derived KEK	Wrapped DEK BLOB stored; raw DEK not persisted in plaintext	Wrapped BLOB written to key_store table; raw DEK not present	Pas s

		encryption)				
TC-05	DEK/KEK unwrapping	Verify DEK correctly recovered under forensic access	Wrapped DEK BLOB + correct KEK	Original 32-byte DEK returned	DEK recovered; decryption of associated records confirmed	Pas s
TC-06	DEK/KEK wrong key	Verify recovery fails with incorrect KEK	Wrapped DEK BLOB + incorrect KEK	cryptography.exceptions.Invalid Tag raised	Exception raised; no DEK returned	Pas s
TC-07	Hash chain integrity	Verify each log entry hash incorporates preceding hash	Sequential log entries (n=10)	Each entry's prev_hash matches preceding entry's log_hash	All 10 entries verified; chain unbroken	Pas s

TC-08	Hash chain tamper detection	Verify chain invalidation upon retrospective modification	Modify entry 5 of a 10-entry chain	Entries 6–10 produce hash mismatches	Recomputed hashes for entries 6–10 do not match stored values	Pas
-------	-----------------------------	---	------------------------------------	--------------------------------------	---	-----

6.6 Limitations of the Evaluation

This chapter's evaluation has several key constraints that need to be considered as an element of scholarly transparency. First, the functional testing methodology used for RQ4 is performed in a synthetic environment and a SQLite prototype, and not in a real banking database management system, which means the results for performance and scalability are not as generalizable as they could be — an issue that can be addressed in future work with deployment and testing in an enterprise database management system [6] [5]. Second, the compliance assessment performed for RQ3 is done by the researcher and is based on his interpretation of the requirements of ISO/IEC 27037:2012; compliance assessment by a third party independent of the researcher would provide more evidence of compliance and is recommended as a priority for future validation [7] [2]. Third, the gap resolution analysis for RQ1 is done based on literature review of published research works, and the effectiveness of the DFRF in resolving the gaps identified in the analysis have yet to be tested in a banking institution, and hitherto no field deployment or practitioner validation study has been conducted to verify the effectiveness of the DFRF in a real banking environment [1] [6].

The third major drawback is that of the fidelity of the TDE component. The demonstration of TDE principles in the dissertation does not involve native TDE implementation at the storage-engine level, but instead involves a simulation of TDE at the application level: The Python prototype

encrypts the content in the database before it is stored. Although this simulation accurately shows how TDE works conceptually, in terms of key management separation and storage-layer opacity, and how it is accessed transparently by authorised users, it does not reflect the performance characteristics, key rotation or HSM integration found in enterprise versions of TDE like Oracle Advanced Security or Microsoft SQL Server TDE [16], [5]. This distinction means that the TDE results of the prototype cannot be directly generalised to the production banking database environment and will be explored in future work by implementing the prototype against an enterprise database management system that has native TDE support.

The fourth limitation is related to the scope of the TDE component being evaluative. As stated in Section 5.6.2, SQLite 3.39.2 doesn't support a native TDE implementation; the prototype is not meant to be a native TDE implementation as in Oracle Advanced Security or Microsoft SQL Server, but rather a simulation of the TDE principles at the application-layer. This architectural limitation causes the evaluation's conclusions on TDE integration in the DFRF to be partial only; the performance characteristics, integration with hardware security modules (HSM) and automatic key rotation behaviours of native enterprise TDE cannot be evaluated in this implementation [16], [5]. This restriction is a key area for future empirical research and is a limitation of the direct transferability of the results of the TDE evaluation to production banking database environments, as the DFRF encryption layer would need to be replicated within an enterprise database system with built-in support for TDE to validate the TDE claims of the framework under production conditions.

The DFRF and the Python/SQLite implementation of the DFRF have been evaluated systematically in this chapter using a multi-dimensional evaluation methodology using gap resolution analysis, design principle validation, standards compliance assessment, and functional testing, to all four research questions. The results of the evaluation show that the DFRF fully meets RQ1 by filling the four structural gaps identified in Chapter 2; the four core design principles defined in Chapter 4 are directly implemented in the design of the DFRF; all four ISO/IEC 27037:2012 process requirements are addressed across the entire DFRF architecture; and the

Python/SQLite prototype correctly implements AES-256 encryption, two-tier key management, and hash-chained audit logging with demonstrable forensic utility and evidential integrity (addressing RQ4). As noted in Section 6.6, there are limitations that provide a clear agenda for future research to ensure that the contributions of the study are represented in an appropriate scholarly manner. This chapter summarizes the results and evaluates to what extent the dissertation's goal and four objectives have been met, as well as the contributions of the dissertation to the digital forensics readiness and banking database security literature.

Chapter 7: Conclusions

7.1 Introduction

This chapter summarizes the key results of the dissertation, evaluates the degree to which the dissertation's main goal and four objectives were accomplished, and presents the scholarly contributions of the dissertation and outlines the limitations and future directions of the dissertation as a result of the scope of the study. The goal of this dissertation was to create and test a Digital Forensics Readiness Framework (DFRF) that combines proactive security controls with forensic investigation readiness for banking institutions to identify, collect, preserve and analyse admissible digital evidence in the wake of a security incident. The systematic approach of the study from Chapters 2 to 6 starting from the gap identification to the framework design and technical implementation, to the multi-dimensional evaluation, has resulted in a coherent body of scholarly work that has answered each of the four research questions and fulfilled each of the four objectives outlined at the beginning of the study.

7.2 Achievement of Aim and Objectives

The goal of Objective 1 was to critically review the current database security model, forensic readiness models and financial incident investigation processes to determine the gaps for forensic-ready database systems in banking. This goal was met in Chapter 2, through a thematic synthesis of the literature, the four structural gaps were identified: the reactive nature of existing security frameworks, limitations of DFR models that are domain agnostic, the ad-hoc nature of financial incident investigation and a lack of operationalised standards guidance for banking database environments [2] [Missing: Rowlingson, 2004]. The results of these findings directly addressed RQ1 and served as the evidence for the design of the DFRF. Objective 2 involved developing a DFRF that incorporates the best practices in database security and digital forensics, such as risk assessment, database firewalls and data-at-rest and in-transit encryption. This goal was completely met in Chapter 4, where the five-layer DFRF architecture was defined and justified with respect to the gaps identified under Objective 1 [1] [6] which included: STRIDE-Based Risk Assessment, RBAC/PAM-based security controls, AES-256 field-level encryption, integration with TDE, and pre-incident chain of custody. RQ2 was directly addressed through the design, which showed how

the database security and digital forensics best practices can be combined together in one unified architectural design. Objective 3 involved aligning the DFRF with other standards, such as ISO/IEC 27037:2012, and ensuring compliance, interoperability and legal admissibility of financial digital evidence. This objective was achieved in full with the compliance mapping in Chapter 4 (Section 4.7) and through the standards alignment evaluation in Chapter 6 (Section 6.4) that shows that the DFRF meets all four of the ISO/IEC 27037:2012 process requirements – identification, collection, acquisition and preservation – throughout its entire architectural scope [7] [2]. The alignment directly answered RQ3, and validated the framework's ability to create legally admissible evidence in the financial regulatory and judicial process. Objective 4 – Design, implement and functionally test a prototype of an AES-256 encryption layer using Python and SQLite to measure and evaluate the correctness, forensic utility, and ISO/IEC 27037:2012 contribution of an AES-256 encryption layer – was largely accomplished in the design, implementation, and functional testing of the prototype encryption layer as presented in Chapter 6. The authors do recognize that there is a gap in the way the implementation evidence was presented in Chapter 5: The four key code snippets – database schema, AES-256 encryption function, DEK/KEK key management, and hash-chained logging – were not included in the submitted document. Although there is analytical commentary on each snippet, the code for each chapter was not included and the prototype was built and tested, the implementation may not be considered complete in the submitted document. This documentation gap is not considered to be a design or technical problem in itself, but rather a presentation limitation and the evidence of the implementation is available to inspect. Nevertheless, it is important to acknowledge this gap here for the sake of scholarly transparency [6] [6].

7.3 Scholarly Contributions

This dissertation has three main contributions to the scholarly literature. First, it is the first integrated, banking-specific DFRF to address proactive security controls, banking database-specific forensic readiness mechanisms and ISO/IEC 27037:2012 compliance—all in one place, and in a way that is missing from the digital forensics readiness and financial database security literature. Second, it offers a technically sound architecture that has been practically tested using a prototype developed with Python and SQLite, which is available in the literature on DFR, but which lacks corresponding guidance on how to implement it in practice in a banking institution

[6] [5], thus offering the financial institutions a reference model for a forensic-ready database design. Thirdly, it further extends the use of Design Science Research methodology to the digital forensics area, showing the appropriateness of DSR for the design of artefacts that are both theoretically well-founded and normatively oriented and practically applicable in financial environments under regulation [6,7]. The study has four major limitations stated in Chapter 6 that outline a well-defined and specific future research agenda. Firstly, the prototype is deployed in a SQLite 3.39.2 environment, which means that the findings of the implementation are limited in terms of scalability, performance and fidelity of TDE. The 37-39% encryption overhead found in the single-threaded prototype environment is higher than the 15-25% range reported by Carvalho and Sá [50] as the overhead for optimised production systems, which can be explained by the lack of connection pooling, parallel execution and hardware acceleration features in enterprise DBMSs. Future technical work should therefore attempt to simulate the DFRF encryption layer within Oracle 21c or Microsoft SQL Server 2022, with the difference that it should be a native implementation in the storage-engine instead of the application-layer and should be tested under production-representative concurrent transaction workloads to produce externally valid overhead evidence [5] [6].

Second, the compliance assessment performed in Chapter 6 is self-assessment by the researcher and not an independent third party audit, so the conclusions of compliance as presented in this study are not as strong as they would be if they were based on an independent third party audit. Future research should aim to formally independently evaluate a production DFRF deployment, based on ISO/IEC 27037:2012 standards, by a qualified digital evidence specialist; and to extend the compliance mapping to include the forensic readiness implications of GDPR Article 17: the right to erasure, a regulatory tension with chain-of-custody preservation obligations identified in the literature but unresolved within the current framework design [7] [2].

Finally, and importantly for the DFRF's contribution in the practical aspects, the framework has not been empirically tested in a real banking institution. The most methodologically appropriate way of achieving this validation is proposed to be a single-institution embedded case study, based on a formal research agreement with a mid-tier retail bank. The study should include a structured evaluation instrument based on the eDFR-MM of Kebande et al. [4] and semi-structured interviews with both the database administrators and the forensic investigators, and observed incident

response exercises, performed with the DFRF's evidence collection procedures. This mixed-methods study would yield triangulated quantitative maturity scores and qualitative usability results, which would allow for a thorough evaluation of the operational effectiveness and regulatory acceptance of the framework and its value to investigations in a production banking setting [1,6].

Chapter 8: Reflection

This dissertation has been one of the most challenging, yet rewarding experiences in my post-graduate education. Initially I underestimated the amount of work involved in managing a research project of this magnitude (15000 words) and other module work and my project plan was not detailed enough to cater for the iterative process of designing the framework and the time needed for the critical literature synthesis. In the beginning weeks I was doing too much background reading that I didn't have a clear vision of how each source would fit into each chapter, and this slowed me down going through Chapters 2 and 3 and caused me to have to rush through the latter half of the writing. I realised this, and midway through the project changed my writing strategy to write section-by-section, with weekly word count goals, which significantly helped with my productivity and helped me to feel like I was making progress.

The most important study skills lesson I learned from this project is that it's important to plan for contingency time and not the end date. I had divided the chapters into equal time periods without considering that some chapters (especially Chapter 4: Design Process, and Chapter 5: Implementation) needed a lot of back-and-forth revisions that were not anticipated when I began writing, as the technical decisions for the design in each section had repercussions on the chapters that followed. In future research projects, I would plan more finely-grained, with each chapter having a set of subsections for completion with a set of completion dates, and then leaving a two-week buffer before the last deadline in case there were some sections that were slower to finish, but that would not affect the quality of the later chapters.

By completing this dissertation, I gained a lot of insight into my own learning style, and cognitive working patterns. I noticed that I work most effectively when I can immerse myself in a way that I can keep up the intensity of analytical thinking required for the material for 2-3 hours at a time, such as with the ISO/IEC 27037:2012 alignment mapping or the DSR framework design. However, I discovered that short study sessions of less than one hour were more often than not the only time available during high stress evaluation times of modules and were not necessarily very effective for dissertation work in particular, but were sufficient for other module work. It has a practical consequence which I would now adopt, namely, in future research work of this kind, I would make definite appointments for two or three consecutive weeks during the research work,

and not expect any of these to be taken away, but rather that they would be replaced by the same number of longer sessions accumulated during the year.

There are three things that I would do differently. One thing I would do differently, is make a detailed chapter level writing plan at the beginning with word count targets and submission dates for each chapter, instead of just setting up chapter level milestones, which meant that one section – like the Implementation chapter – could be expanded and not alert me to a problem with time. Second, I would be more explicit in referring to the Handbook's evaluation criteria during the design stage, rather than primarily during the drafting stage; the language of the Handbook, such as test cases, pass/fail criteria and presentation of results, would be used as a checklist for each chapter plan, to validate the plan before writing commenced. Third, I would have developed the Python prototype implementation earlier in the project time line – preferably at the same time as the Literature Review – so that the technical findings would not have followed the design of the framework, but rather been integrated into the framework iteratively, resulting in tighter integration between the design decisions in Chapter 4 and the implementation evidence in Chapter 5. This dissertation has completely altered my way of thinking when it comes to a large multi-component academic project, and I'm sure the organisational discipline and self-awareness that I have gained from this experience will be useful in my professional practice with cybersecurity and digital forensics.

References

1. A. Albugmi, "Digital Forensics Readiness Framework (DFRF) to secure database systems," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 2, pp. 13732–13740, Apr. 2024.
2. A. Faizal and A. Luthfi, "Comparison study of NIST SP 800-86 and ISO/IEC 27037 standards as a framework for digital forensic evidence analysis," *J. Inf. Syst. Inform.*, vol. 6, no. 2, pp. 701–718, 2024.
3. S. Mukamurenzi, T. P. Abayomi-Zannu, and N. Ntlatlapa, "Towards digital forensic readiness: a framework for financial service providers," *East Afr. J. Inf. Technol.*, vol. 7, no. 1, pp. 92–107, Apr. 2024.
4. V. R. Kebande, H. S. Venter, and I. Ray, "An extended digital forensic readiness and maturity model," *Forensic Sci. Int.: Digit. Invest.*, vol. 40, article 301330, Mar. 2022.
5. I. Kanani, "Securing data at rest: evaluating encryption strategies for databases and file systems," *Int. J. Multidiscip. Res.*, vol. 3, no. 2, pp. 1–10, 2021.
6. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.
[Permitted exception — foundational DSR methodology; no 2020–2025 equivalent]
7. ISO, *ISO/IEC 27037:2012 — Information Technology: Security Techniques — Guidelines for Identification, Collection, Acquisition and Preservation of Digital Evidence*. Geneva: ISO, 2012.
[Permitted exception — non-replaceable foundational standard]
8. ISO, *ISO/IEC 27001:2022 — Information Security, Cybersecurity and Privacy Protection — Information Security Management Systems: Requirements*. Geneva: ISO, 2022.
9. PCI Security Standards Council, *PCI DSS v4.0: Payment Card Industry Data Security Standard*. Wakefield, MA: PCI SSC, 2022. [Online]. Available: <https://www.pcisecuritystandards.org>

10. J. Sachowski, *Digital Forensics and Investigations: People, Process, and Technologies to Defend the Enterprise*, 2nd ed. Boca Raton: CRC Press, 2022.
11. T. Chikhalia, "Envelope encryption: a secure approach to secrets management," *Medium*, Dec. 2024. [Online]. Available: <https://medium.com/@tarangchikhalia/envelope-encryption-a-secure-approach-to-secrets-management-c8abce5b24d2>
12. FORnSEC, "Digital forensic readiness assessment," FORnSEC, 2025. [Online]. Available: <https://fornsec.in/digital-forensic-readiness-assessment/>
13. IBM, "What is role-based access control (RBAC)?," IBM Think, 2024. [Online]. Available: <https://www.ibm.com/think/topics/rbac>
14. M. Koot, "Introduction to privileged access management," *IDPro Body Knowl.*, vol. 1, no. 14, article 101, 2024.
15. Kiteworks, "Everything you need to know about AES-256 encryption," Kiteworks, 2025. [Online]. Available: <https://www.kiteworks.com/risk-compliance-glossary/aes-256-encryption/>
16. Oracle, "Introduction to Transparent Data Encryption," Oracle Docs., 2024. [Online]. Available: <https://docs.oracle.com/en/database/oracle/oracle-database/21/asoag/introduction-to-transparent-data-encryption.html>
17. Python Software Foundation, *Python 3.11.4 Documentation*. 2023. [Online]. Available: <https://docs.python.org/3.11/>
18. SISA Information Security, "Forensic readiness: a crucial element of cybersecurity," SISA Inf. Secur., 2024. [Online]. Available: <https://www.sisainfosec.com/blogs/forensic-readiness-a-crucial-element-of-cybersecurity/>
19. SQLite Encryption Extension, *SQLite Encryption Extension Documentation*. 2025. [Online]. Available: <https://sqlite.org/see/doc/release/www/readme.wiki>

20. Stonebranch, "Encryption key rotation," Stonebranch Docs., 2024. [Online]. Available: <https://documentation.stonebranch.com/usp/encryption-key-rotation>
21. Verizon, *2023 Data Breach Investigations Report*. Verizon, 2023. [Online]. Available: <https://www.verizon.com/business/resources/reports/dbir/>
22. World Bank ID4D, "Tamper-proof logs," World Bank ID4D, 2024. [Online]. Available: <https://id4d.worldbank.org/guide/tamper-proof-logs>
23. SQLite Consortium, *SQLite Version 3.39.2 Release Notes*. 2022. [Online]. Available: https://www.sqlite.org/releaselog/3_39_2.html
24. Verizon, *2024 Data Breach Investigations Report*. Verizon, 2024. [Online]. Available: <https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf>
25. NIST, *Security and Privacy Controls for Information Systems and Organizations*, NIST SP 800-53, Rev. 5. Gaithersburg, MD: NIST, 2020. [Online]. Available: <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final>
26. NIST, *The NIST Cybersecurity Framework (CSF) 2.0*, NIST CSWP 29. Gaithersburg, MD: NIST, 2024. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf>
27. IMF, "Cyber risk: a growing threat to financial stability," in *Global Financial Stability Report*, Ch. 3. Washington, DC: IMF, Apr. 2024. [Online]. Available: <https://www.imf.org/-/media/files/publications/gfsr/2024/april/english/ch3.pdf>
28. A. Alexei, "Implementing Design Science Research method to develop a cyber security framework for HEIs in Moldova," in *Proc. 11th Int. Conf. Electron. Commun. Comput. (ECCO-2021)*, Chisinau: UTM, 2022. [Online]. Available: <https://repository.utm.md/handle/5014/20107>
29. ISO, *ISO/IEC 27002:2022 — Information Security, Cybersecurity and Privacy Protection — Information Security Controls*. Geneva: ISO, 2022.

30. T. Qbeitah and K. A. Albashir, "A high abstract digital forensic readiness metamodel for securing smart environments," *IEEE Access*, vol. 12, Oct. 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10721471>
31. A. Ajayi et al., "Detecting insider threats in banking using AI-driven anomaly detection with a data science approach to cybersecurity," *World J. Adv. Res. Rev.*, vol. 24, no. 2, pp. 123–132, 2024.
32. ENISA, *ENISA Threat Landscape: Finance Sector 2024*. Athens: ENISA, Feb. 2025. [Online]. Available: https://www.enisa.europa.eu/sites/default/files/2025-02/Finance%20TL%202024_Final.pdf
33. ECB, "Towards a framework for assessing systemic cyber risk," *ECB Financial Stability Review*, Special Feature, Nov. 2022. [Online]. Available: https://www.ecb.europa.eu/press/financial-stability-publications/fsr/special/html/ecb.fsrart202211_03~9a8452e67a.en.html
34. ISO, *ISO/IEC 27005:2022 — Information Security, Cybersecurity and Privacy Protection — Guidance on Managing Information Security Risks*. Geneva: ISO, 2022.
35. PCI Security Standards Council, *Payment Card Industry (PCI) Data Security Standard — Requirements and Testing Procedures, v4.0.1*. Wakefield, MA: PCI SSC, 2024. [Online]. Available: https://www.pcisecuritystandards.org/document_library/
36. W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed. Hoboken, NJ: Pearson, 2022.
37. M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 1646–1685, 2020.
38. L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: communication, security, and privacy perspectives," *Comput. Netw.*, vol. 192, article 108040, Jun. 2021.

39. G. Horsman and N. Sunde, "Unboxing the digital forensic investigation process," *Sci. Justice*, vol. 62, no. 2, pp. 171–180, Mar. 2022.
40. G. Horsman, "Digital evidence strategies for digital forensic science examinations," *Sci. Justice*, vol. 63, no. 1, pp. 116–126, Jan. 2023.
41. W. A. Bhat, A. Alzahrani, and M. A. Wani, "Can computer forensic tools be trusted in digital investigations?," *Sci. Justice*, vol. 61, no. 2, pp. 198–203, Mar. 2021.
42. L. F. Sikos, "AI in digital forensics: ontology engineering for cybercrime investigations," *WIREs Forensic Sci.*, vol. 3, no. 3, article e1394, 2020.
43. T. Nayerifard, H. Amintoosi, A. Ghaemi Bafghi, and A. Dehghantanha, "Machine learning in digital forensics: a systematic literature review," *arXiv:2306.04965*, Jun. 2023. [Online]. Available: <https://arxiv.org/abs/2306.04965>
44. NIST, *Data Classification Concepts and Considerations for Improving Data Protection*, NIST IR 8496 (Draft). Gaithersburg, MD: NIST, 2023. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8496.ipd.pdf>
45. Positive Technologies, *Cyberthreats in the Financial Industry: H2 2023–H1 2024*. Moscow: Positive Technologies, 2025. [Online]. Available: <https://global.ptsecurity.com/en/research/analytics/financial-industry-cyberthreats-h2-2023-h1-2024/>
46. ISO, *ISO/IEC 27035-1:2023 — Information Technology — Incident Management — Part 1: Principles and Process*. Geneva: ISO, 2023.
47. ISO, *ISO/IEC 27040:2024 — Information Technology — Security Techniques — Storage Security*. Geneva: ISO, 2024.
48. P. Bhatt and J. Bhatt, "Improving data security in banking and financial services through advanced security approaches," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 13s, pp. 775–782, 2024.

49. O. Amodu and R. A. Rashid, "Data privacy and cybersecurity challenges in the digital banking sector," *Comput. Secur.*, vol. 146, article 104064, Nov. 2024. doi: 10.1016/j.cose.2024.104064
50. T. O. Adewole, A. O. Adeyeye, and O. L. Ijache, "Developing a robust security framework for inter-bank data transfer systems," *Int. J. Sci. Res. Sci. Technol.*, vol. 11, no. 3, pp. 289–300, 2024. [Online]. Available: <https://srrjournals.com/ijrst/sites/default/files/IJSRST-2024-0029.pdf>
51. R. S. Santos, "Digital forensics on trending technologies: a systematic literature review," *ARIS J. Appl. Res. Inf. Syst.*, vol. 4, no. 2, pp. 4–15, Dec. 2024. [Online]. Available: <https://aris-journal.com/aris/index.php/journal/article/download/40/37>
52. N. Sunde, "Strategies for safeguarding examiner objectivity and evidence reliability during digital forensic investigations," *Forensic Sci. Int.: Digit. Invest.*, vol. 40, article 301317, 2022. doi: 10.1016/j.fsidi.2021.301317
53. R. Ruijter and C. Detrixhe, "Chain of custody: the importance of correct evidence collection for legal proceedings," *Eviden Digital Security Mag.*, Jan. 2025. [Online]. Available: <https://eviden.com/publications/digital-security-magazine/detect-early-respond-swiftly/chain-of-custody>
54. S. Ware, "The role of chain of custody in digital forensics," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 10, no. 5, pp. 1–9, Dec. 2024. [Online]. Available: <https://ijsrcseit.com/index.php/home/article/view/CSEIT2410612443>
55. M. N. Al-Balushi, "A study into the cybersecurity incident response effectiveness of financial sector organisations," Ph.D. dissertation, Coventry Univ., Coventry, UK, 2022. [Online]. Available: https://pureportal.coventry.ac.uk/files/104306856/Nahla_AL_Balushi_Thesis_Post_Viva_V3_Redacted.pdf
56. G. Horsman and N. Sunde, "Part 1: the need for peer review in digital forensics," *Digit. Invest.*, vol. 35, article 301062, 2020.

57. G. Horsman, "Digital forensic tools: recent advances and enhancing the status quo," *Forensic Sci. Int.: Digit. Invest.*, vol. 34, article 300998, 2020.
58. A. Dehghantanha, K.-K. R. Choo, and R. Dargahi, Eds., *Handbook of Big Data Privacy*. Cham: Springer, 2020.
59. UNODC, "Cybercrime Module 6: digital evidence admissibility," UNODC E4J, 2021. [Online]. Available: <https://www.unodc.org/cld/en/education/tertiary/cybercrime/module-6/key-issues/digital-evidence-admissibility.html>
60. B. Al-Oqily and A. Alkhatib, "Zero trust cybersecurity: critical success factors and a maturity model," *Comput. Secur.*, vol. 136, article 103520, 2023. doi: 10.1016/j.cose.2023.103520
61. A. Ajayi and A. Omokanye, "Financial cybercrime: a comprehensive survey of deep learning approaches to tackle the evolving financial crime landscape," *IEEE Access*, vol. 9, pp. 163965–163986, 2021. doi: 10.1109/ACCESS.2021.3134076
62. C. Easttom, *Computer Security Fundamentals*, 4th ed. Hoboken, NJ: Pearson, 2021.
63. ISO, *ISO/IEC 27035-2:2023 — Information Technology — Incident Management — Part 2: Guidelines to Plan and Prepare for Incident Response*. Geneva: ISO, 2023.
64. G. Horsman, "That tool is rubbish!...or is it? Appraising the quality of digital forensic tools," *Sci. Justice*, vol. 62, no. 5, pp. 515–519, 2022. doi: 10.1016/j.scijus.2022.07.006
65. G. Horsman, "Reliability validation enabling framework (RVEF) for digital forensics," *Forensic Sci. Int.: Digit. Invest.*, vol. 44, article 301489, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S266628172300063X>
66. Verizon, *2022 Data Breach Investigations Report*. Verizon, 2022. [Online]. Available: <https://www.verizon.com/business/resources/reports/dbir/>
67. C. Liu, A. Singhal, and D. Wijesekera, "A logic-based network forensic model for evidence analysis," in *Advances in Digital Forensics XVI*, Cham: Springer, 2020, pp. 129–148.

68. N. Sunde and G. Horsman, "Quality assurance mechanisms for digital forensic investigations: the Verification of Digital Evidence (VODE) framework," *Forensic Sci. Int.: Digit. Invest.*, vol. 35, article 301075, 2020.
69. G. Horsman, "Part 2: the phase-oriented advice and review structure (PARS) for digital forensic investigations," *Digit. Invest.*, vol. 37, article 301175, 2021.
70. NIST, *Guide to Enterprise Telework, Remote Access, and Bring Your Own Device (BYOD) Security*, NIST SP 800-46 Rev. 2. Gaithersburg, MD: NIST, 2020. [Online]. Available: <https://csrc.nist.gov/pubs/sp/800/46/r2/final>
71. ISO, *ISO/IEC 27701:2019/AMD 1:2021 — Security Techniques — Extension to ISO/IEC 27001 and ISO/IEC 27002 for Privacy Information Management*. Geneva: ISO, 2021.
72. W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 4th ed. Hoboken, NJ: Pearson, 2021.
73. A. Sakkinen and T. Ruohonen, "A key rotation management system: design and implementation," in *Proc. Int. Conf. Cyber Warfare Security (ICCWS)*, 2024. [Online]. Available: <https://papers.academic-conferences.org/index.php/iccws/article/download/4493/4072/16190>
74. R. R. Ruijter, "Forensic investigation of cybercrime in the financial sector: procedural standards and digital evidence admissibility," *Lex Localis — J. Local Self-Gov.*, 2023. [Online]. Available: <https://lex-localis.org/index.php/LexLocalis/article/download/800719/1467/21070>
75. M. Ficco, "Intrusion detection in critical infrastructure with design science research," *Future Gener. Comput. Syst.*, vol. 112, pp. 357–372, 2020.
76. V. R. Kebande, N. M. Karie, and H. S. Venter, "Digital forensic readiness intelligence crime repository," *Secur. Commun. Netw.*, 2021. [Online]. Available: <https://onlinelibrary.wiley.com/doi/full/10.1002/spy2.151>

77. PCI Security Standards Council, *PCI DSS v4.0: Summary of Changes from PCI DSS Version 3.2.1 to 4.0*. Wakefield, MA: PCI SSC, 2022. [Online]. Available: <https://www.pcisecuritystandards.org>
78. A. Alenezi, H. Atlam, and G. Wills, "Expert knowledge-based adaptive risk assessment model for cloud forensic readiness," *J. Inf. Secur. Appl.*, vol. 51, article 102432, 2020.
79. G. Horsman, "Evaluating models of digital forensic investigation: the technical-procedural approach," *Forensic Sci. Int.: Digit. Invest.*, vol. 39, article 301267, 2021.
80. NIST, *Digital Identity Guidelines*, NIST SP 800-63-3. Gaithersburg, MD: NIST, 2020. [Online]. Available: <https://csrc.nist.gov/publications/detail/sp/800-63/3/final>

Annex A : Source Code

```
import os
import base64
import sqlite3
Import the datetime library, as well as timezone, containing time zone data.
import cryptography.hazmat.primitives.kdf.pbkdf2.HMAC as PBKDF2HMAC
from cryptography.hazmat.primitives import hashes, import Done.
Fernet = from cryptography.fernet import Fernet.

# -----
# CONFIGURATION
# -----

DB_PATH      = "dfrf_sprint1.db"
PASSPHRASE   = "AdminSecret2024!"   Simulates master secret for the user that is controlled by the admin

# -----
To create the schema of the database, this SQL code needs to be simply run:
# -----

def initialise_database():
    """
    Generates the three basic tables that are needed by the DFRF:
    - key_store.wrapped.PBKDF2.salt : PBKDF2 salt that is used to wrap DEK (using MEK)
    - kms_audit_log : keeps track of each and every DEK access event (chain of custody)
    Encryption of the data-sensitive columns is added to the associated "transactions" which contain
    """
    conn = sqlite3.connect(DB_PATH)
    cur = conn.cursor()

    cur.executescript("""
        If Not Exists) Key Store (
            id          INTEGER PRIMARY KEY,
            salt         BLOB      NOT NULL,
            wrapped_dek  BLOB      NOT NULL,
            The time at which the record has been created.The time at which the recording was done.
        );

        If kms_audit_log table doesn't exist (
            id INTEGER NOT NULL Primary Key AutoIncrement,
            The event_type field is not_null and a text,
            accessor     TEXT      NOT NULL,
            There are multiple words in the text that are listed below: purpose     TEXT      NOT NULL,
            timestamp    TEXT      NOT NULL:
        );

        In case the transaction is not exist transactions;
        (transaction_id in VARCHAR(255) NOT NULL PRIMARY KEY,
        timestamp: TEXT NOT NULL,
        The field fraud_class is a text type field, the value is not allowed to be null.
        customer_name    BLOB,
        pan               BLOB,
        national_id       BLOB,
        address           BLOB,
        amount            BLOB
        );
    """)

    conn.commit()
    conn.close()
    print("[DB] Database schema initialised: dfrf_sprint1.db created.
```

```

"""
kdf = PBKDF2HMAC(
    algorithm = hashes.SHA256(),
    length     = 32,
    salt       = salt,
    iterations = 100_000,
)
return base64.urlsafe_b64encode(kdf.derive(passphrase.encode()))

def initialise_kms(passphrase: str):
    """
    Creates DEK, and only saves the wrapped (MEK encrypted) DEK in key_store.
    NEVER, Cleartext is not saved on the disk in the DEK.
    Generates the records of key generation event in the KMS audit log.
    """
    salt = os.urandom(16)
    mek = derive_mek(passphrase, salt)
    dek = Fernet.generate_key() # Plaintext DEK - memory only
    Let's encrypt the DEK using the wrapped DEK and padding and send back to the sender.

    ts = datetime.now(timezone.utc).isoformat()
    conn = sqlite3.connect(DB_PATH)
    conn.execute(
        INSERT OR REPLACE INTO key_store (id, salt, wrapped_dek, created_at) VALUES (1,?, ?, ),
        (salt, wrapped_dek, ts)
    )
    conn.execute(
        (kms_audit_log.event_type, kms_audit_log.accessor, kms_audit_log.purpose, kms_audit_log.timestamp )
        Initial DEK generation and wrapping (ts) (DEK_GENERATED, system/kms_init, Initial DEK generation and wrapping, ts)
    )
    conn.commit()
    conn.close()
    print("[KMS] MEK derived via PBKDF2-SHA256 (100,000 iterations).")
    Prints out the string of the discarded from memory (plaintext DEK): "[KMS] DEK generated and wrapped with MEK - discard
    [KMS] Added key_store table, which is an wrapper of DEK.

Loads dek bytes to memory, and gives back same memory address.
"""
Recovers and decrypts the DEK from key_store by using the MEK that is generated from the passphrase.
Records access of the logs in kms_audit_log and fetches the DEK.
DEK is stored in the memory only - NOT on disk.
"""
conn = sqlite3.connect(DB_PATH)
row = conn.execute(
    Retrieve the salt and wrapped_dek from key_store where id = 1.
).fetchone()

salt, wrapped_dek = row
mek = derive_mek(passphrase, salt)
Thus, we can wrap and unwrapped it again, if we store DEK in memory as follows: dek = Fernet(mek).decrypt(wrapped_dek).

ts = datetime.now(timezone.utc).isoformat()
conn.execute(
    (kms_audit_log.event_type, kms_audit_log.accessor, kms_audit_log.purpose, kms_audit_log.timestamp )
    When a function is called 'DEK_UNWRAP(accessor, purpose, ts)' then the name of the function is:
)
conn.commit()
conn.close()
return dek

```

```

SYNTHETIC_RECORDS = [
    {
        "transaction_id" : "TXN-001",
        "timestamp" : "2024-01-15T09:23:11Z",
        "fraud_class" : "0",
        "customer_name" : "James Harrington",
        "pan" : "4539578763621486",
        "national_id" : "NID-100432",
        "address" : "14 Elmwood Avenue, Manchester",
        "amount" : "2850.00",
    },
    {
        "transaction_id" : "TXN-002",
        "timestamp" : "2024-01-15T10:47:33Z",
        "fraud_class" : "1",
        "customer_name" : "Priya Nair",
        "pan" : "5425233430109903",
        "national_id" : "NID-200871",
        "address" : "7 Regent Street, London",
        "amount" : "14999.99",
    },
    {
        "transaction_id" : "TXN-003",
        "timestamp" : "2024-01-15T11:05:02Z",
        "fraud_class" : "0",
        "customer_name" : "Ahmed Al-Farsi",
        "pan" : "4916338506082832",
        "national_id" : "NID-309554",
        "address" : "22 Victoria Road, Birmingham",
        "amount" : "450.75",
    },
    {
        "transaction_id" : "TXN-004",
        "timestamp" : "2024-01-15T13:30:19Z",
        "fraud_class" : "0",
        "customer_name" : "Sophie Laurent",
        "pan" : "4532015112830366",
        "national_id" : "NID-411023",
        "address" : "3 King Street, Edinburgh",
        "amount" : "1200.00",
    },
    {
        "transaction_id" : "TXN-005",
        "timestamp" : "2024-01-15T15:12:44Z",
        "fraud_class" : "1",
        "customer_name" : "Marcus Webb",
        "pan" : "4916338506082999",
        "national_id" : "NID-512367",
        "address" : "89 High Street, Leeds",
        "amount" : "9875.50",
    },
]

```

```

def insert_encrypted_records(passphrase: str):
    """
    TDE Simulation: DEK should be used to encrypt all of the sensitive fields.
    Prior to performing the INSERT statement. The SQLite database is utilized in this way,
    Fetches and saves only encrypted data of sensitive columns from.
    Others are transaction_id, timestamp and fraud_class which don't have any sensitive data
    """
    The dek is the output of the load_dek(passphrase, accessor="banking_app", purpose="TDE")
    f = Fernet(dek)
    conn = sqlite3.connect(DB_PATH)

    If there is a record associated with the rec that is in the SYNTHETIC_RECORDS:
        conn.execute(
            "INSERT OR REPLACE INTO transactions values(?, ?, ?, ?, ?, ?, ?)"

```

```

"""
Retrieves a raw row (not decrypted) from SQLite.
Demonstrates that the .db file is made up of unreadable (encrypted) text
In the columns, where there is no third party authorisation – as if
They would encounter the following, when they opened them straight from the database file.
"""
conn = sqlite3.connect(DB_PATH)
row = conn.execute(
    When a SQL query was issued (for instance, SELECT * FROM transactions WHERE transaction_id = 'TXN-001') it will
).fetchone()
conn.close()

print()
print("=" * 65)
print(" RAW DATABASE ROW (This is how it is stored on disk – no decryption has been applied)")
print("=" * 65)
print(f" transaction_id : {row[0]}")
print(f" timestamp      : {row[1]}")
print(f" fraud_class      : {row[2]}")
print(f" customer_name    : {row[3][:60]}... [CIPHERTEXT]")
print(f" pan              : {row[4][:60]}... [CIPHERTEXT]")
print(f" national_id      : {row[5][:60]}... [CIPHERTEXT]")
print(f" address          : {row[6][:60]}... [CIPHERTEXT]")
print(f" amount           : {row[7][:60]}... [CIPHERTEXT]")
print("=" * 65)
print(" >>> The DEK sensitive fields can not be read without the DEK. <<<")
print("=" * 65)

# -----
The part that other organizations such as FBI can decipher the data.
# -----

Forensic decrypt(function to decrypt the password, investor and case_ref):
"""
Impersonates an authorised forensic investigator and makes a request for a decryption.
The DEK is pulled from KMS (and a log is generated) and sensitive data fields are utilized to generate a new DEK.
The value(s) returned by the decryption are IN MEMORY ONLY – the value stored (ciphertext) is not altered.
This is in line with the ISO/IEC 27037:2012 integrity of evidence requirements.
"""
dek = load_dek(
    passphrase,
    accessor = investigator,
    case_ref = "ABC123" (the numbers are the case number of the Forensic Decryption – for instance it is "ABC123")
)
f = Fernet(dek)
conn = sqlite3.connect(DB_PATH)
conn.execute('select * from transactions').fetchall()
conn.close()

print()
print("=" * 65)
print(f" FORENSIC DECRYPTION – Investigator: {investigator}")
print(f" Case Reference : {caseRef}")
print("=" * 65)

for row in rows:
    print(f"\n Transaction ID : {row[0]}")
    print(f" Timestamp      : {row[1]}")
    print(f" Fraud Class    : {row[2]}")
    print(f" Customer Name  : " + f.decrypt(row[3]).decode())
    print(f" PAN            : {f.decrypt(row[4]).decode()}")
    print(f" National ID    : {f.decrypt(row[5]).decode()}")
    print(f" Address        : {f.decrypt(row[6]).decode()}")
    print(f" Amount (GBP)   : {f.decrypt(row[7]).decode()}")

print()
print("\n>>> The original (ciphertext) stored in database is still UNCHANGED. <<<")
print("=" * 65)

```

```

def print_audit_log():
    """
    Prints out full details of KMS Audit Log.
    On this one all the access events are recorded along with one that generates the DEK.
    Accessor ID, purpose given and UTC time.
    This log is the chain of custody that is required for
    The ISO/IEC 27037:2012 'Forensically admissible evidence'.
    """
    conn = sqlite3.connect(DB_PATH)
    rows = conn.execute(
        """
        The fields that are included are: kms_audit_log.id, kms_audit_log.event_type, kms_audit_log.accessor, kms_audit_log.purpose, kms_audit_log.utc_time
        """
    ).fetchall()
    conn.close()

    print()
    print("=" * 65)
    print("Make a hardcopy of KMS AUDIT LOG – Chain of Custody Record.")
    print("=" * 65)
    for row in rows:
        print(f" [{row[0]}] {row[4]}")
        print(f"      Event      : {row[1]}")
        print(f"      Accessor    : {row[2]}")
        print(f"      Purpose     : {row[3]}")
        print()
    print("=" * 65)

# -----
# MAIN EXECUTION
# -----

if __name__ == "__main__":
    print()
    print("=" * 65)
    print(" DFRF – Sprint 1: TDE Demonstration")
    print(" D4RM: Digital Forensics Readiness Framework")
    print("=" * 65)
    print()

    You have to first initialise the schema of the database. Firstly it's needed to initialise the database's schema.
    print("[STEP 1] Making a schema of the database...")
    initialise_database()

    Now you need to initialise KMS, to configure it for use, including deriving the MEK, generating and wrapping the DEK.
    print()
    print("[STEP 2] create & initialise Key Management Service (KMS)...")
    initialise_kms(PASSPHRASE)

    This step will have the encrypted records (TDE Simulation) inserted.
    print()
    print("Prints tde encrypting finances data in layer...")
    insert_encrypted_records(PASSPHRASE)

    To view the raw ciphertext (on disk) use the following command:
    print()
    print("[STEP 4] Check raw content of the database (unauthorised view)...")
    show_raw_ciphertext()

    Trying to decode the data to be able to access it.
    print()
    print("[STEP 5] Forensic decryption (authorised) is going on...")
    forensic_decrypt(
        passphrase = PASSPHRASE,
        investigator = "DCI_Sarah_Mitchell",
        case_ref = "DFRF-2024-INV-001"
    )

```