DECISION-ORIENTED AUDITING OF ENCRYPTION AND KEY MANAGEMENT POLICIES BASED ON CONSISTENCY, STABILITY, AND RISK METRICS

J. R. Baratov, A. N. Ulashev, T. T. Aynakulov

Department of Computer Science and Programming, Jizzakh Branch, National University of Uzbekistan, Jizzakh, Uzbekistan

ABSTRACT

Auditing encryption and key management policies in modern web and server systems is complicated by architectural complexity and continuous configuration change. Existing approaches largely rely on static compliance checks or isolated metrics, providing limited support for actionable decision-making. This paper proposes a decision-oriented framework that bridges metric-based auditing and practical security governance. The framework relies on system-level abstractions of policy requirements and enforcement evidence, and maps consistency, conflict, stability, and risk metrics to discrete decision outcomes. A bounded and non-intrusive satisfaction function supports partial compliance, heterogeneous evidence, and conservative handling of missing data without accessing cryptographic key material. In addition, a risk-aware remediation prioritization algorithm ranks policy requirements by urgency and architectural impact. Scenario-based evaluation demonstrates improved interpretability of audit results and supports proactive, risk-aware remediation planning.

KEYWORDS

Encryption policy auditing, key management, decision-oriented security auditing, policy–enforcement consistency, configuration drift, risk-aware remediation, satisfaction functions, system-level security metrics

1. INTRODUCTION

The widespread adoption of encryption and centralized key management mechanisms has become a fundamental requirement for securing modern web and server systems. Contemporary software architectures rely on transport-layer encryption, encrypted storage, and dedicated key management services to protect sensitive data and meet regulatory and organizational security requirements. Consequently, policies governing cryptographic algorithms, protocol versions, key lengths, and key lifecycles have grown increasingly complex and system-specific [1, 2, 5].

Although comprehensive standards and guidelines are provided by organizations such as NIST, ISO/IEC, and ENISA, practical enforcement of encryption and key management policies remains error-prone [1–7]. Empirical studies consistently report widespread misconfigurations, use of deprecated cryptographic parameters, and inconsistent policy enforcement across real-world systems, including environments that nominally satisfy compliance requirements [12, 13]. These problems are intensified by continuous system evolution, frequent configuration changes, and the layered structure of modern software architectures.

Most existing approaches to encryption policy auditing rely on static compliance checks or checklist-based validation against predefined requirements [3–5]. While effective at detecting

isolated violations at a specific point in time, such methods provide limited insight into internal policy coherence, cross-layer enforcement mismatch, or temporal behavior under configuration drift. As a result, systems may appear compliant while exhibiting structurally inconsistent or unstable enforcement patterns that remain undetected by traditional audits.

To overcome the limitations of binary compliance assessment, recent research has emphasized quantitative security metrics and automated auditing techniques [16–19]. However, metric-based approaches often lack clear semantic interpretation: numerical values are reported without explicit mapping to operational decisions or remediation priorities. In addition, temporal aspects such as configuration drift, delayed key rotation, or gradual weakening of cryptographic settings are rarely modeled explicitly, leading either to overreaction to transient deviations or failure to detect systematic policy erosion [18, 21].

This paper addresses these challenges by proposing a decision-oriented framework for auditing encryption and key management policies in dynamic software systems. The proposed approach integrates system-level audit metrics with a formal decision model that maps quantitative observations to discrete, interpretable audit outcomes. Consistency, conflict, stability, and risk metrics are jointly analyzed to distinguish acceptable configurations, early warning conditions, configuration drift, and critical misconfigurations. A formally defined satisfaction function enables partial compliance assessment without accessing cryptographic key material, while a risk-aware remediation prioritization algorithm translates audit decisions into actionable guidance.

The main contributions of this work are summarized as follows:

1. a system-level abstraction of encryption and key management policies and their enforcement evidence suitable for continuous auditing [1, 2, 7];
2. a decision-oriented interpretation model that transforms audit metrics into discrete operational states, enabling clear and reproducible audit outcomes [16, 19];
3. a formal satisfaction function design supporting heterogeneous evidence, partial compliance, and conservative handling of missing data [18, 21];
4. a risk-aware remediation prioritization algorithm that ranks policy violations according to urgency, temporal instability, and architectural impact [12, 17];
5. a scenario-based evaluation demonstrating the advantages of decision-oriented auditing over metric-only interpretation.

The remainder of this paper is organized as follows. Section 2 reviews related work on encryption policy enforcement, security metrics, and automated auditing. Section 3 introduces the system model and formal abstractions. Section 4 defines system-level audit metrics, while Section 5 presents the decision-oriented audit model. Section 6 formalizes the satisfaction function design, and Section 7 introduces the remediation prioritization algorithm. Section 8 evaluates the proposed framework through representative scenarios, followed by discussion and conclusions in Sections 9 and 10.

2. RELATED WORKS

Research on encryption and key management policy auditing spans several complementary domains, including cryptographic standards, empirical studies of misconfiguration, security metrics, and automated compliance frameworks. This section reviews representative work in these areas and highlights the limitations that motivate the proposed decision-oriented approach.

2.1. Cryptographic Standards and Policy Guidance

International standards and recommendations issued by organizations such as NIST, ISO/IEC, ENISA, and IETF define accepted cryptographic algorithms, protocol versions, key lengths, and key lifecycle practices [1–7]. These documents establish authoritative policy baselines and play a critical role in regulatory compliance and organizational security governance.

However, standards primarily specify *what* should be enforced rather than *how* enforcement should be audited in complex, evolving systems. They provide limited guidance on quantifying partial compliance, analyzing internal policy coherence, or interpreting temporal changes in enforcement. Consequently, audits based solely on standards often reduce policy assessment to binary compliance outcomes.

2.2. Empirical Studies of Cryptographic Misconfiguration

Extensive empirical research demonstrates that cryptographic misconfiguration remains widespread in real-world systems. Large-scale measurement studies of TLS deployments and PKI infrastructures report persistent use of deprecated protocols, weak configurations, and inconsistent certificate management practices [12, 13]. Developer-focused studies further reveal frequent misuse of cryptographic APIs due to complexity and insufficient tooling support [14, 15].

While these studies provide strong evidence of the prevalence and impact of cryptographic policy violations, they are largely descriptive. They identify misconfigurations but do not propose systematic frameworks for continuous auditing, metric aggregation, or decision-oriented interpretation across heterogeneous system components.

2.3. Security Metrics and Risk Assessment

Security metrics have been widely proposed as a means of moving beyond qualitative or checklist-based security assessments [16–19]. Surveys and systematic reviews catalog numerous metrics for evaluating system security and risk exposure.

Despite their analytical value, most metrics are reported in isolation and lack explicit semantic interpretation. Numerical values are rarely mapped to operational decisions or remediation priorities, and temporal aspects such as configuration drift and gradual degradation are often treated implicitly. This limits the practical usefulness of metric-based approaches in continuous auditing scenarios.

2.4. Policy Compliance and Automated Auditing Frameworks

Research on automated compliance checking explores formal representations of security policies and their enforcement [20]. Process-based and model-driven approaches improve automation by verifying rule satisfaction against observed system behavior.

However, these approaches typically focus on point-in-time compliance and do not address partial satisfaction, cross-layer enforcement conflicts, or remediation prioritization under resource constraints. As a result, automated audits may produce extensive violation reports without clear guidance on urgency or operational impact.

2.5. Key Management Systems and Cloud Environments

The adoption of centralized key management services in cloud platforms introduces additional complexity into encryption policy enforcement. Vendor documentation describes configuration options and operational practices for cloud-based KMS solutions [22–24], but these sources are platform-specific and do not provide system-agnostic auditing models.

Moreover, cloud environments amplify temporal dynamics through frequent automated updates, making it difficult for existing auditing approaches to distinguish controlled policy evolution from unintended configuration drift.

2.6. Research Gap and Positioning of This Work

The reviewed literature reveals a gap between quantitative measurement and actionable interpretation in encryption and key management policy auditing. Standards define requirements without audit semantics; empirical studies expose problems without decision frameworks; metrics quantify properties without operational meaning; and automated audits detect violations without prioritization.

This work addresses this gap by introducing a decision-oriented auditing framework that integrates system-level metrics, formal satisfaction semantics, temporal analysis, and risk-aware remediation prioritization. By explicitly mapping audit metrics to discrete decision states and actionable guidance, the proposed approach supports consistent and interpretable security governance in dynamic software systems.

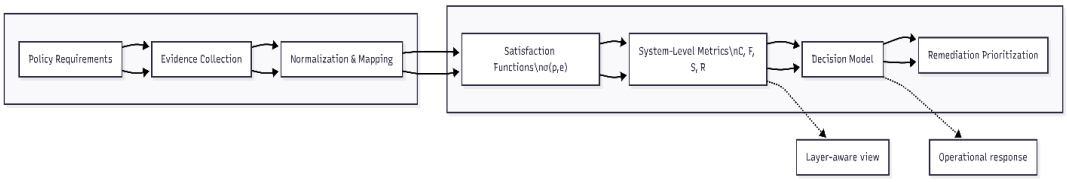

Fig. 1. Decision-oriented encryption and key management policy auditing framework

Fig. 1 Overview of the proposed decision-oriented framework for auditing encryption and key management policies. Policy requirements and heterogeneous enforcement evidence are normalized and evaluated using satisfaction functions, aggregated into system-level metrics, interpreted through a decision model, and translated into prioritized remediation actions.

3. SYSTEM MODEL AND POLICY ABSTRACTION

This section introduces the formal system model and abstraction layers used throughout the paper. The objective is to establish a precise representation of encryption and key management policies, their enforcement artifacts, and the temporal audit context, while remaining independent of specific platforms or implementations.

3.1. System Architecture Model

We consider a software system as a composition of interacting architectural layers that collectively enforce encryption and key management policies. Let

$$\mathscr{L} = \{\ell_1, \ell_2, \ldots, \ell_m\} \quad (1)$$

denote the finite set of architectural layers, such as application, transport, storage, and key management layers.

Each layer $\ell \in \mathscr{L}$exposes a set of configuration interfaces and runtime artifacts through which cryptographic enforcement is realized. This layered abstraction reflects modern system architectures and aligns with common security standards and deployment practices [1, 3, 5].

3.2. Policy Requirement Model

Encryption and key management policies are formalized as a set of atomic policy requirements. Let

$$\mathcal{P} = \{p_1, p_2, \dots, p_n\} \tag{2}$$

denote the set of policy requirements applicable to the system.

Each policy requirement $p_i \in \mathcal{P}$is represented as a tuple

$$p_i = \langle \ell(p_i), \tau(p_i), \kappa(p_i), w_i \rangle, \tag{3}$$

where $\ell(p_i) \in \mathscr{L}$denotes the architectural layer to which the requirement applies, $\tau(p_i)$specifies the requirement type (e.g., algorithm selection, key length, protocol version, key rotation interval), $\kappa(p_i)$defines the formal constraint associated with the requirement, and $w_i \in (0,1]$denotes its relative importance.

This abstraction enables policy requirements derived from standards such as NIST and ISO/IEC to be represented uniformly, regardless of their source or scope [1, 2, 5, 7].

3.3. Enforcement Artifact Model

Policy enforcement is observed through concrete system artifacts, including configuration files, runtime metadata, management APIs, and audit logs. Let

$$\mathscr{E} = \{e_1, e_2, \dots, e_k\} \tag{4}$$

denote the set of enforcement artifacts collected during an audit.

Each artifact $e_j \in \mathscr{E}$is associated with one or more policy requirements and is abstracted as a normalized attribute vector

$$e_j = \langle v_{j,1}, v_{j,2}, \dots, v_{j,m_j} \rangle. \tag{5}$$

Normalization ensures that heterogeneous evidence sources—such as TLS configuration parameters, key metadata from key management systems, or protocol negotiation results—can be compared against policy constraints in a uniform manner [4, 22–24].

3.4. Policy–Enforcement Mapping

The relationship between policy requirements and enforcement artifacts is modeled as a mapping

$$\mathscr{M} \subseteq \mathcal{P} \times \mathscr{E}, \tag{6}$$

where $(p_i, e_j) \in \mathscr{M}$indicates that artifact $e_j$provides enforcement evidence for requirement p_i.

This mapping is many-to-many: a single policy requirement may be enforced by multiple artifacts across different layers, and a single artifact may contribute evidence for multiple requirements. Such multiplicity is characteristic of real-world systems and is a common source of policy–enforcement inconsistency [12, 18].

3.5. Temporal Audit Model

To capture system dynamics, audits are performed at discrete time points. Let

$$\mathcal{T} = \{t_1, t_2, \dots\} \tag{7}$$

denote the ordered set of audit times.

At each time $t \in \mathcal{T}$, a snapshot of enforcement artifacts $\mathscr{E}_t \subseteq \mathscr{E}$is collected, producing a time-indexed policy–enforcement mapping $\mathscr{M}_t$. This temporal model enables analysis of configuration

drift, delayed key rotation, and gradual policy erosion, which are typically invisible to point-in-time audits [18, 21].

3.6. Scope and Assumptions

The proposed system model makes several deliberate assumptions. First, audit access is limited to configuration data and metadata; cryptographic key material is never accessed or inspected directly, in line with best practices and regulatory constraints [2, 7]. Second, policy requirements are assumed to be explicitly defined and externally available, for example through organizational security policies or regulatory baselines. Third, the model assumes that enforcement artifacts can be normalized into comparable representations, although the specific extraction mechanisms are system-dependent.

These assumptions ensure that the model remains broadly applicable while preserving audit safety and deployability.

3.7. Role of the Model in the Overall Framework

The abstractions introduced in this section form the foundation for all subsequent analysis. Policy requirements, enforcement artifacts, and their temporal relationships are used in Section 4 to define system-level audit metrics. These metrics are then interpreted through the decision model in Section 5, grounded by satisfaction functions in Section 6, and ultimately translated into prioritized remediation actions in Section 7.

4. SYSTEM-LEVEL AUDIT METRICS

This section defines the quantitative metrics used to evaluate encryption and key management policy enforcement at the system level. Building on the policy, enforcement, and temporal abstractions introduced in Section 3, the metrics capture structural alignment, mismatch, temporal behavior, and risk exposure in a unified and bounded form suitable for decision-oriented auditing.

4.1. Local Satisfaction and Requirement Aggregation

Let $p_i \in \mathcal{P}$ denote a policy requirement and $\mathscr{E}_t(p_i) \subseteq \mathscr{E}_t$ the set of enforcement artifacts associated with p_i at time t. Using the satisfaction function $\sigma(p_i, e_j) \in [0,1]$ (formalized later in Section 6), the local satisfaction of requirement p_i at time t is defined as

$$\sigma_t(p_i) = \frac{1}{|\mathscr{E}_t(p_i)|} \sum_{e_j \in \mathscr{E}_t(p_i)} \sigma(p_i, e_j), \ \sigma_t(p_i) \in [0,1] \quad (8)$$

where w_i denotes the relative importance of requirement p_i.
This aggregation preserves requirement-level weighting while preventing artifact multiplicity from disproportionately influencing satisfaction.

4.2. Global Consistency Metric

The global consistency metric quantifies the degree of structural alignment between policy requirements and observed enforcement across the entire system. At time t, consistency is defined as the normalized weighted average of local satisfactions:

$$C_t = \frac{\sum_{p_i \in \mathcal{P}} w_i \cdot \sigma_t(p_i)}{\sum_{p_i \in \mathcal{P}} w_i}, C_t \in [0,1]. \quad (9)$$

Higher values indicate stronger policy–enforcement alignment, with explicit support for partial compliance.

4.3. Conflict Metric

While consistency captures alignment, it does not explicitly quantify the extent of mismatch. To address this, the conflict metric is defined as the complement of consistency:

$$F_t = 1 - C_t, F_t \in [0,1]. \quad (10)$$

This formulation provides a direct measure of aggregated policy–enforcement conflict, enabling auditors to reason about the severity of mismatches without collapsing results into binary outcomes.

4.4. Temporal Stability Metric

Modern systems evolve continuously, making temporal analysis essential. The stability metric captures changes in enforcement between consecutive audit snapshots.

Let $\sigma_t(p_i)$and $\sigma_{t-1}(p_i)$denote the local satisfaction of requirement $p_i$at times $t$and $t-1$, respectively. Temporal stability is defined as

$$S_t = 1 - \frac{\sum_{p_i \in \mathcal{P}} w_i \cdot |\sigma_t(p_i) - \sigma_{t-1}(p_i)|}{\sum_{p_i \in \mathcal{P}} w_i}, S_t \in [0,1]. \quad (11)$$

Lower values indicate temporal instability, including configuration drift.

4.5. Risk Metric

To support decision-making and remediation prioritization, audit metrics must be interpretable in terms of risk. The system-level risk metric aggregates requirement-level exposure as a function of satisfaction and importance:

$$R_t = \frac{\sum_{p_i \in \mathcal{P}} w_i \cdot (1 - \sigma_t(p_i))}{\sum_{p_i \in \mathcal{P}} w_i}, R_t \in [0,1]. \quad (12)$$

This formulation reflects the intuition that highly important requirements with low satisfaction contribute disproportionately to overall risk [16, 19].

4.6. Layer-Aware Metric Decomposition

Given the layered system model introduced in Section 3.1, metrics can be decomposed by architectural layer. Let $\mathcal{P}_\ell \subseteq \mathcal{P}$denote the set of requirements associated with layer $\ell \in \mathscr{L}$. Layer-specific consistency is defined as

$$C_t^{(\ell)} = \frac{\sum_{p_i \in \mathcal{P}_\ell} w_i \cdot \sigma_t(p_i)}{\sum_{p_i \in \mathcal{P}_\ell} w_i}. \quad (13)$$

Layer-aware metrics enable localized diagnosis and support targeted remediation strategies, particularly in systems where enforcement responsibilities are distributed across teams or services [12, 22–24].

4.7. Metric Properties and Interpretability

All proposed metrics are bounded within the unit interval, ensuring comparability and numerical stability. Consistency and risk metrics are monotonic with respect to satisfaction, while stability explicitly captures temporal variation. Importantly, metric definitions are independent of specific cryptographic algorithms or platforms, relying solely on abstracted policy and enforcement representations.

These properties make the metrics suitable inputs for the decision model introduced in Section 5, where they are mapped to discrete audit outcomes and remediation strategies.

Table 1. Summary of system-level audit metrics, their interpretation, and usage within the proposed framework.

Metric	Range	Interpretation	Used in Sections
Consistency (C)	[0,1]	Degree of alignment between policy requirements and observed enforcement	4, 5
Conflict (F)	[0,1]	Aggregated policy–enforcement mismatch	4, 5
Stability (S)	[0,1]	Temporal consistency between consecutive audit snapshots	4, 5, 8
Risk (R)	[0,1]	Weighted exposure based on requirement importance and satisfaction	4, 7

All metrics are normalized to $[0, 1]$. C: consistency; F: conflict; S: temporal stability; R: risk. "Used in Sections" indicates the primary sections where each metric is formally defined or operationally applied.

5. DECISION-ORIENTED AUDITING MODEL

5.1. Decision Space and Audit Outcome States

Existing encryption and key management policy auditing approaches typically report numerical metrics without explicitly defining how such values should be interpreted in operational terms. Consequently, audit results often lack clear guidance on whether a system state is acceptable, requires attention, or demands immediate remediation. To address this limitation, this section introduces a decision-oriented audit model that formally maps system-level audit metrics to a finite and interpretable decision space.

Let an audit snapshot observed at time $t$be represented by the metric tuple

$$\mathrm{M}_t = \langle C_t, F_t, S_t, R_t \rangle, \tag{14}$$

where $C_t \in [0,1]$denotes global policy consistency, $F_t \in [0,1]$denotes aggregated policy–enforcement conflict, $S_t \in [0,1]$denotes temporal stability between consecutive audit snapshots, and $R_t \in [0,1]$denotes the normalized audit risk.

Decision Space Definition

The decision space is defined as a finite set of mutually exclusive audit outcome states:

$$\mathcal{D} = \{\text{Acceptable, Warning, Critical, Drift}\}. \tag{15}$$

Each element of $\mathcal{D}$represents a qualitatively distinct operational condition of the audited system. In particular, this classification explicitly distinguishes between static policy violations and temporally evolving misconfigurations, a distinction not captured by metric-only auditing approaches.

Metric-to-Decision Mapping Function

Let $\theta_C, \theta_F, \theta_S, \theta_R \in (0,1)$denote configurable decision thresholds. The mapping from audit metrics to decision outcomes is formalized by the decision function

$$\delta(\mathrm{M}_t): \mathrm{M}_t \to \mathcal{D}. \tag{16}$$

An audit snapshot is classified as Acceptable if the following condition holds:

$$C_t \geq \theta_C \wedge R_t \leq \theta_R \wedge S_t \geq \theta_S. \tag{17}$$

A Warning state is assigned when the system remains largely compliant but exhibits early signs of instability:

$$C_t \geq \theta_C \wedge R_t \leq \theta_R \wedge S_t < \theta_S. \quad (18)$$

A snapshot is classified as Critical whenever a significant policy–enforcement mismatch or elevated risk is detected:

$$C_t < \theta_C \vee R_t > \theta_R \vee F_t > \theta_F. \quad (19)$$

Finally, a Drift condition is identified when the inequality in Equation (18) persists across multiple consecutive audit snapshots, indicating sustained configuration evolution rather than transient deviation:

$$\exists\, k \geq 2 \text{s.t}\, S_{t-i} < \theta_S, i = 0, \ldots, k-1. \quad (20)$$

Interpretation and Operational Significance

The proposed decision-oriented formulation transforms abstract audit metrics into discrete, interpretable system states that directly support operational decision-making. By explicitly separating warning conditions from long-term drift, the model reduces false positives and enables more precise remediation planning.

Moreover, the bounded nature of all metrics and thresholds ensures reproducibility across audit environments while remaining independent of system-specific implementation details.

5.2. Threshold-Based Classification and Decision Rules

While the audit metrics introduced in Section 4 provide quantitative insight into policy–enforcement alignment, operational security auditing requires interpretable and actionable outcomes. To this end, the proposed framework employs a threshold-based decision interpretation that maps continuous metric values to a finite set of operational audit states.
An audit snapshot observed at time *t* is represented by the metric vector

$$M_t = \langle C_t, F_t, S_t, R_t \rangle, \quad (21)$$

where consistency, conflict, stability, and risk capture complementary aspects of policy enforcement. Rather than interpreting these values in isolation, they are jointly evaluated against configurable decision thresholds reflecting organizational risk tolerance and regulatory requirements.

Based on this interpretation, the audit outcome space is defined as a finite set of decision states: *Acceptable*, *Warning*, *Critical*, and *Drift*. An *Acceptable* state corresponds to high policy consistency, low risk, and stable enforcement over time. A *Warning* state indicates early signs of degradation, typically manifested as reduced temporal stability while overall compliance and risk remain within acceptable bounds. A *Critical* state is triggered whenever significant policy–enforcement mismatch or elevated risk is detected, requiring immediate remediation.

Temporal behavior plays a central role in distinguishing transient deviations from sustained degradation. Short-term instability may arise from benign configuration updates or maintenance activities and should not automatically trigger aggressive remediation. In contrast, persistent instability observed across consecutive audit snapshots is interpreted as configuration drift, signaling gradual and potentially unintentional erosion of policy enforcement. By incorporating temporal persistence into decision interpretation, the framework avoids overreacting to isolated fluctuations while ensuring that long-term degradation is detected in a timely manner.

Importantly, this decision-oriented interpretation explicitly separates metric computation from decision semantics. Metrics remain continuous and system-agnostic, while decision thresholds can be adjusted to reflect evolving organizational policies or external standards. This separation ensures reproducibility of audit results and enables consistent operational decision-making across heterogeneous systems.

Overall, the proposed threshold-based decision interpretation transforms raw audit metrics into discrete, semantically meaningful system states. This approach enables auditors and security operations teams to clearly distinguish acceptable configurations, early warning conditions, sustained configuration drift, and critical misconfigurations, thereby supporting timely and proportionate remediation actions.

6. SATISFACTION FUNCTION DESIGN FOR POLICY–ENFORCEMENT EVALUATION

Audit metrics and decision rules defined in previous sections rely on the accurate quantification of the degree to which observed enforcement artifacts satisfy formal policy requirements. This quantification is captured by the satisfaction function, which transforms heterogeneous and partially observable enforcement evidence into bounded numerical values suitable for aggregation and decision-making. This section formalizes the design principles, mathematical properties, and operator-level constructions of the satisfaction function $\sigma(p, e)$.

6.1. Role and Requirements of the Satisfaction Function

Let $p \in \mathcal{P}$denote a policy requirement and $e \in \mathscr{E}$denote an associated enforcement artifact. The satisfaction function

$$\sigma: \mathcal{P} \times \mathscr{E} \to [0,1] \tag{22}$$

quantifies the extent to which $e$satisfies p.

For use in system-level auditing and decision-oriented analysis, $\sigma(p, e)$must satisfy the following requirements:

First, boundedness: all outputs must lie within the closed interval $[0, 1]$, enabling aggregation across heterogeneous requirements.

Second, semantic monotonicity: stronger enforcement must not yield lower satisfaction values.

Third, partial satisfaction support: deviations from policy must be captured gradually rather than collapsed into binary outcomes.

Fourth, non-intrusiveness: satisfaction must be computable without accessing cryptographic key material, relying solely on configuration and metadata.

These requirements ensure that $\sigma(p, e)$serves as a reliable interface between low-level enforcement observations and high-level audit decisions.

6.2. Formal Satisfaction Semantics

Let a policy requirement $p$be represented as a tuple

$$p = \langle \text{type, constraint, scope} \rangle, \tag{23}$$

where *type* denotes the requirement category (e.g., key length, protocol version), *constraint* defines acceptable values, and *scope* specifies the architectural layer to which the requirement applies.

Similarly, let an enforcement artifact $e$be represented by the extracted attribute vector

$$e = \langle v_1, \ldots, v_m \rangle, \tag{24}$$

obtained through normalization of configuration files, runtime metadata, or management APIs.

The satisfaction value $\sigma(p, e)$is computed by comparing $e$against the constraint defined by p, using operators tailored to the requirement type.

6.3. Operator Classes for Common Encryption Policy Requirements

Encryption and key management policy requirements exhibit heterogeneous structural properties, including numeric thresholds, ordered discrete values, set-based constraints, and temporal

conditions. To ensure generality while preserving interpretability, the proposed satisfaction function employs a small set of operator classes, each tailored to a common category of cryptographic policy requirements.

For numeric threshold requirements, such as minimum key length or iteration count, satisfaction is computed as a normalized ratio between the observed value and the required minimum. This formulation yields full satisfaction when policy requirements are met or exceeded and penalizes under-enforcement proportionally:

$$\sigma(p, e) = \min\left(1, \frac{v_{\text{obs}}}{v_{\text{req}}}\right) \tag{25}$$

where v_{obs}denotes the observed value and v_{req}denotes the required minimum.

Ordered discrete requirements arise in domains where configurations can be ranked according to cryptographic strength, such as protocol versions or algorithm classes. In this case, satisfaction is defined as the ratio of the observed rank to the required rank, ensuring monotonicity with respect to cryptographic strength:

$$\sigma(p, e) = \frac{\text{rank}(v_{\text{obs}})}{\text{rank}(v_{\text{req}})}, 0 < \sigma \leq 1, \tag{26}$$

Set-based requirements are used to express constraints over collections of acceptable or required configurations, for example cipher suite selections. Partial compliance is captured using normalized set similarity, reflecting the degree of overlap between observed and required configurations:

$$\sigma(p, e) = \frac{|E_{\text{obs}} \cap E_{\text{req}}|}{|E_{\text{obs}} \cup E_{\text{req}}|}. \tag{27}$$

Temporal requirements, such as key rotation intervals or certificate validity periods, require explicit modeling of time-dependent deviation. Satisfaction for temporal constraints is therefore defined using an exponential decay function that penalizes increasing delay beyond the required threshold:

$$\sigma(p, e) = \exp\big(-\lambda \cdot \max(0, t_{\text{obs}} - t_{\text{req}})\big), \tag{28}$$

where the parameter λcontrols the severity of temporal penalties.

6.4. Handling Missing and Uncertain Evidence

In practical audit settings, enforcement evidence may be incomplete or partially observable. To avoid overestimating compliance, missing evidence is treated conservatively.

Let $e = \emptyset$denote missing or inaccessible enforcement data. The satisfaction function is then defined as

$$\sigma(p, \emptyset) = \sigma_{\min}, \sigma_{\min} \in (0,1), \tag{29}$$

where $\sigma_{\min}$is a configurable lower bound reflecting uncertainty rather than explicit violation.

This approach preserves audit continuity while discouraging unjustified compliance inflation.

6.5. Satisfaction Aggregation and Weighting

For a policy requirement $p_i$associated with multiple enforcement artifacts $e_j \in \mathscr{E}(p_i)$, local satisfaction is defined as

$$\sigma(p_i) = \frac{\sum_{e_j \in \mathscr{E}(p_i)} w_i \cdot \sigma(p_i, e_j)}{w_i \cdot |\mathscr{E}(p_i)|}, \tag{30}$$

where $w_i$denotes the relative importance of requirement p_i.

This aggregation preserves requirement-level weighting while avoiding disproportionate influence of artifact multiplicity.

6.6. Properties of the Satisfaction Function

The proposed satisfaction function design satisfies the following properties:
Boundedness follows directly from operator definitions.
Continuity holds for numeric and temporal operators, ensuring smooth response to gradual changes.
Monotonicity is preserved with respect to enforcement strength.
Non-intrusiveness is guaranteed by construction, as no cryptographic secrets are accessed.
These properties collectively ensure that satisfaction values are suitable inputs for the decision model defined in Section 5.

7. RISK-AWARE REMEDIATION PRIORITIZATION ALGORITHM

While the decision-oriented audit model identifies the operational state of a system, effective security governance requires transforming decisions into prioritized remediation actions. In complex web and server environments, remediation resources are limited, and not all policy violations or instabilities can be addressed simultaneously. This section introduces a risk-aware remediation prioritization algorithm that ranks audit findings based on their impact, urgency, and architectural relevance.

7.1. From Decision Outcomes to Remediation Objectives

Let an audit snapshot at time $t$yield a decision outcome

$$d_t = \delta(\mathrm{M}_t) \in \mathcal{D}, \tag{31}$$

as defined in Section 5. The objective of remediation prioritization is to determine an ordered set of policy requirements

$$\mathcal{P}_t^{\uparrow} = \langle p_{(1)}, p_{(2)}, \dots, p_{(n)} \rangle, \tag{32}$$

where the ordering reflects decreasing remediation priority.
Remediation is triggered for decision states *Warning*, *Critical*, and *Drift*, while *Acceptable* states require no immediate corrective action.

7.2. Requirement-Level Risk Decomposition

Global risk metrics obscure the contribution of individual policy requirements. To enable fine-grained remediation, global risk is decomposed into requirement-level components.
Let $R_t(p_i) \in [0,1]$denote the risk contribution associated with policy requirement $p_i$at time t, computed as

$$R_t(p_i) = w_i \cdot (1 - \sigma(p_i)), \tag{33}$$

where $w_i$denotes the relative importance of requirement p_i, and $\sigma(p_i)$is the aggregated satisfaction defined in Equation (30).
This formulation ensures that highly important requirements with low satisfaction are assigned higher remediation urgency.

7.3. Incorporating Temporal Instability and Conflict

Risk alone is insufficient to capture remediation urgency in dynamic systems. Temporal instability and structural conflict must also be considered.
Let $S_t(p_i)$denote the local stability of requirement p_i, and let $F_t(p_i)$denote its conflict contribution. A composite urgency score is defined as

$$U_t(p_i) = \alpha \cdot R_t(p_i) + \beta \cdot (1 - S_t(p_i)) + \gamma \cdot F_t(p_i), \tag{34}$$

where $\alpha, \beta, \gamma \geq 0$are tunable weighting coefficients satisfying $\alpha + \beta + \gamma = 1$.

This formulation balances immediate risk, temporal degradation, and structural mismatch in a unified prioritization metric.

7.4. Layer-Aware Prioritization Strategy

Encryption and key management policies span multiple architectural layers, including application, transport, storage, and key management services. Remediation actions targeting different layers exhibit varying operational costs and systemic impact.

Let $\ell(p_i) \in \mathscr{L}$denote the architectural layer associated with requirement p_i. A layer-adjusted priority score is defined as

$$\widetilde{U}_t(p_i) = U_t(p_i) \cdot \lambda_{\ell(p_i)}, \tag{35}$$

where $\lambda_\ell$is a layer-specific amplification factor reflecting remediation complexity or criticality.

This adjustment enables strategic remediation planning by emphasizing layers with higher systemic risk exposure.

7.5. Prioritization Algorithm

The remediation prioritization process is formalized by the following algorithmic workflow.

Input:

Audit snapshot M_t, requirement set $\mathcal{P}$, satisfaction values $\sigma(p_i)$, stability values $S_t(p_i)$, conflict values $F_t(p_i)$.

Output:

Ordered remediation list $\mathcal{P}_t^{\uparrow}$.

1. For each $p_i \in \mathcal{P}$, compute $R_t(p_i)$using Equation (33).
2. Compute urgency score $U_t(p_i)$using Equation (34).
3. Apply layer adjustment using Equation (35).
4. Sort all $p_i$in descending order of $\widetilde{U}_t(p_i)$.
5. Output the ordered list $\mathcal{P}_t^{\uparrow}$.

This algorithm is deterministic, interpretable, and compatible with automated remediation pipelines.

7.6. Decision-Dependent Remediation Policies

The aggressiveness of remediation actions depends on the decision outcome d_t.

For *Critical* states, immediate remediation is recommended for the top-ranked requirements. For *Drift* states, remediation may be scheduled or combined with enhanced monitoring. For *Warning* states, corrective actions may be deferred pending trend confirmation.

This decision-dependent strategy prevents overreaction to transient deviations while ensuring timely response to severe violations.

7.7. Discussion and Practical Implications

The proposed remediation prioritization algorithm transforms audit results into an actionable security management instrument. By integrating risk, stability, conflict, and architectural context, it supports informed decision-making under resource constraints.

Importantly, the algorithm avoids black-box optimization or learning-based heuristics, preserving transparency and auditability—properties essential in regulated environments.

8. CASE STUDY AND SCENARIO-BASED EVALUATION

This section evaluates the proposed decision-oriented auditing and remediation framework through controlled scenarios representative of modern web and server systems. The objective is to demonstrate how the proposed metrics, decision rules, and prioritization algorithm jointly enable accurate interpretation of audit results and effective remediation planning under both static and dynamic conditions.

8.1. Experimental Setup

The evaluation environment models a multi-layer software system comprising application, transport, storage, and key management components. Each layer enforces a subset of encryption and key management policy requirements defined over protocol versions, cryptographic parameters, and key lifecycle properties.

Audit snapshots are collected at discrete time points, producing metric vectors $\mathrm{M}_t$as defined in Equation (5.1-1). Satisfaction values $\sigma(p_i)$, stability metrics $S_t(p_i)$, and conflict indicators $F_t(p_i)$are computed using the methods introduced in Sections 6 and 5, respectively. Threshold values are fixed across scenarios to ensure comparability.

8.2. Scenario Definitions

Three evaluation scenarios are considered, each designed to highlight a distinct operational condition commonly encountered in practice.

Scenario A: Stable and Compliant Configuration
In this scenario, all policy requirements are satisfied or exceeded, and no significant configuration changes occur over time. Observed satisfaction values remain close to unity, and temporal stability is high across all requirements.
Formally, for all $t$and p_i,

$$\sigma(p_i) \approx 1, S_t(p_i) \geq \theta_S, \tag{36}$$

resulting in a global decision outcome of *Acceptable*. The remediation prioritization algorithm produces an empty or low-priority remediation list, confirming that no corrective action is required.

Scenario B: Gradual Configuration Drift
This scenario models unintentional degradation, such as delayed key rotation or incremental weakening of cryptographic parameters. Satisfaction values decrease gradually while remaining above compliance thresholds, and stability metrics consistently fall below θ_S.
For a subset of requirements $\mathcal{P}_d \subset \mathcal{P}$,

$$\sigma(p_i) \searrow, C_t \geq \theta_C, S_t < \theta_S, p_i \in \mathcal{P}_d. \tag{37}$$

The decision model correctly classifies this condition as *Drift*. The remediation prioritization algorithm ranks drifting requirements according to urgency scores defined in Equation (34), enabling proactive intervention before critical violations emerge.

Scenario C: Acute Misconfiguration Event
The third scenario introduces a sudden policy violation, such as the use of deprecated protocol versions or disabled key rotation. Satisfaction values drop sharply, and conflict and risk metrics exceed their respective thresholds.
Formally,

$$\exists\, p_j \in \mathcal{P} \text{ s.t. } \sigma(p_j) \ll 1 \vee R_t(p_j) > \theta_R. \tag{38}$$

The decision model assigns a *Critical* state, and the remediation algorithm prioritizes the offending requirement at the top of the remediation list. This outcome demonstrates the framework's ability to distinguish acute violations from gradual degradation.

8.3. Decision Outcomes, Remediation Prioritization, and Interpretation

Across all evaluated scenarios, including the illustrative case study, the decision-oriented audit model produced outcomes consistent with expected operational behavior. Stable configurations were classified as *Acceptable*, early instability as *Warning*, and acute policy violations as *Critical*. Decision transitions occurred only under sustained threshold violations, confirming the suitability of the proposed model for continuous auditing rather than point-in-time assessment.

The illustrative case study further demonstrates how decision outcomes emerge from the joint interpretation of consistency, stability, and risk metrics. In particular, the key rotation requirement at the key management layer was classified as *Drift* due to reduced temporal stability across audit snapshots, despite remaining within nominal consistency bounds. This highlights the ability of the decision model to distinguish gradual policy erosion from acute misconfiguration—a distinction not observable through static compliance checks.

Remediation prioritization results align with these decision outcomes. The proposed prioritization algorithm ranked policy requirements according to risk contribution, temporal instability, and architectural impact. Requirements associated with key management and other lower architectural layers consistently received higher priority due to their systemic influence on overall security posture. In contrast, requirements classified as *Acceptable* were deprioritized, while *Drift* conditions were elevated for proactive remediation before escalation into critical violations.

Compared to metric-only auditing approaches, the proposed framework provides clearer and more actionable outcomes. By explicitly mapping audit metrics to discrete decision states and prioritized remediation actions, the framework reduces interpretive ambiguity and improves the operational usefulness of audit results. Metric values that appear acceptable in isolation can thus be contextualized within temporal behavior and architectural impact, enabling more consistent and informed decision-making by auditors and security operations teams.

The evaluation focuses on controlled scenarios, including the compact illustrative case study, to isolate decision behavior and avoid confounding factors. While real-world systems may exhibit greater scale and complexity, the scenarios capture representative patterns of stability, configuration drift, and misconfiguration commonly observed in operational environments. Threats to validity include simplified system modeling and fixed decision thresholds; however, these limitations do not affect the qualitative conclusions regarding decision interpretability, temporal sensitivity, and remediation effectiveness.

9. DISCUSSION

This section discusses the implications, strengths, and limitations of the proposed decision-oriented auditing framework, and analyzes potential threats to validity. The discussion emphasizes how the framework advances current practice beyond metric-only auditing while maintaining transparency and auditability.

9.1. Practical Implications for Auditors and Security Operations

The proposed framework transforms encryption and key management audits from passive compliance assessments into actionable decision-support processes. By explicitly mapping audit metrics to discrete decision states and prioritized remediation actions, the framework reduces interpretive ambiguity and shortens response times in operational environments.

For auditors, the decision space introduced in Section 5 provides a clear semantic interpretation of audit outcomes, enabling consistent reporting and governance alignment. For security operations teams, the remediation prioritization algorithm in Section 7 offers a structured mechanism to allocate limited resources based on quantified risk, temporal instability, and architectural impact.
Importantly, the framework supports continuous auditing without requiring intrusive access to cryptographic secrets, making it suitable for regulated and production environments.

9.2. Advantages over Metric-Only and Compliance-Driven Approaches

Unlike traditional audits that report isolated metric values or binary compliance flags, the proposed approach integrates metrics into a coherent decision model. This integration enables differentiation between transient deviations, gradual drift, and acute misconfigurations—distinctions that are critical for effective remediation planning but are typically absent in checklist-based audits.

Furthermore, the explicit separation between metric computation and decision interpretation allows organizations to adjust decision sensitivity through threshold calibration without redefining metrics. This flexibility supports adaptation to evolving security standards and risk appetites.

9.3. Interpretability and Transparency

A key design goal of the framework is interpretability. All decision outcomes and remediation priorities are derived from deterministic rules and clearly defined thresholds. Unlike machine learning–based approaches, the framework avoids opaque decision logic, ensuring that audit results remain explainable to auditors, system owners, and regulatory bodies.

This transparency is particularly important in environments where audit findings must be justified, documented, and reproduced across independent assessments.

This work has limitations related to threshold calibration and the use of scenario-based evaluation. While the framework is designed to be system-agnostic, practical deployment requires system-specific normalization mechanisms. Large-scale empirical validation is left for future work.

9.4. Positioning within the Broader Research Landscape

Within the broader landscape of encryption and key management auditing research, this work occupies a middle ground between purely theoretical metric frameworks and purely engineering-focused audit tools. By combining formal metrics, decision semantics, and remediation logic, the framework addresses a critical gap between analysis and action.

The discussion highlights that the primary contribution of this work is not the introduction of new cryptographic metrics, but rather the systematic interpretation and operationalization of existing metrics within a decision-oriented audit process.

10. CONCLUSION AND FUTURE WORK

This paper presented a decision-oriented framework for auditing encryption and key management policies in dynamic web and server systems, addressing the gap between quantitative audit metrics and actionable security decisions. Unlike traditional approaches focused on static compliance checking or isolated metric reporting, the proposed framework systematically transforms audit results into interpretable decision states and prioritized remediation actions.

The core contribution of this work lies in the integration of system-level metrics, decision semantics, and remediation logic within a unified audit process. Consistency, conflict, stability, and risk metrics are jointly interpreted through a threshold-based decision model, enabling explicit differentiation between acceptable configurations, early warning conditions, configuration drift, and critical misconfigurations. This decision model is grounded in a formally defined satisfaction function that supports partial compliance, heterogeneous enforcement evidence, and non-intrusive evaluation without accessing cryptographic key material.

A key strength of the proposed approach is its interpretability and transparency. All audit outcomes and remediation priorities are derived from deterministic rules and configurable thresholds, avoiding opaque or data-driven black-box mechanisms. This property makes the framework suitable for regulated and production environments, where audit results must be explainable, reproducible, and aligned with organizational security policies.

Scenario-based evaluation demonstrated that the framework effectively distinguishes between transient deviations, gradual enforcement drift, and acute policy violations, while providing clear remediation guidance based on quantified risk and temporal behavior. Compared to metric-only auditing, the decision-oriented approach reduces ambiguity and improves the operational usefulness of audit results.

Future work will focus on large-scale empirical validation across heterogeneous production environments, adaptive calibration of decision thresholds, and integration with automated remediation and orchestration platforms. These extensions aim to further enhance the practical applicability and operational impact of decision-oriented auditing for encryption and key management policies.

REFERENCES

[1] Barker, E. (2019). Transitioning the use of cryptographic algorithms and key lengths (NIST SP 800-131A Rev. 2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-131Ar2

[2] Barker, E. (2020). Recommendation for key management – Part 1: General (NIST SP 800-57 Rev. 5). National Institute of Standards and Technology.https://doi.org/10.6028/NIST.SP.800-57pt1r5

[3] Ross, R., Pillitteri, V., Dempsey, K., Riddle, M., & Guissanie, G. (2020). Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5). NIST. https://doi.org/10.6028/NIST.SP.800-53r5

[4] McKay, K. A., Cooper, M. S., Gallagher, J. F., & Padilla, S. W. (2019). Guidelines for the selection, configuration, and use of TLS implementations (NIST SP 800-52 Rev. 2). NIST.https://doi.org/10.6028/NIST.SP.800-52r2

[5] ISO/IEC. (2022). ISO/IEC 27002: Information security, cybersecurity and privacy protection — Information security controls. ISO.https://www.iso.org/standard/75652.html

[6] ISO/IEC. (2022). ISO/IEC 27001: Information security management systems — Requirements. ISO.https://www.iso.org/standard/82875.html
[7] European Union Agency for Cybersecurity. (2022). Algorithms, key sizes and parameters. ENISA.
[8] Rescorla, E. (2018). The Transport Layer Security (TLS) protocol version 1.3 (RFC 8446). IETF.https://doi.org/10.17487/RFC8446
[9] Sheffer, Y., Holz, R., & Saint-Andre, P. (2022). Recommendations for secure use of TLS and DTLS (RFC 9325). IETF. https://doi.org/10.17487/RFC9325
[10] Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC Press.https://cacr.uwaterloo.ca/hac/
[11] Stallings, W. (2020). Cryptography and network security: Principles and practice (8th ed.). Pearson.https://www.pearson.com/en-us/subject-catalog/p/cryptography-and-network-security/P200000003295
[12] Holz, R., Braun, L., Kammenhuber, N., & Carle, G. (2011). The SSL landscape: A thorough analysis of the X.509 PKI. Proceedings of the ACM Conference on Computer and Communications Security, 1–14.https://doi.org/10.1145/2046707.2046744
[13] Durumeric, Z., Kasten, J., Bailey, M., & Halderman, J. A. (2017). The security impact of HTTPS interception. Network and Distributed System Security Symposium (NDSS).https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/security-impact-https-interception/
[14] Acar, Y., Backes, M., Fahl, S., et al. (2016). Developers' use of cryptography in practice. ACM Conference on Usable Security and Privacy (SOUPS), 1–15.
[15] Krüger, S., Späth, J., Acar, Y., et al. (2017). CogniCrypt: Supporting developers in using cryptography. IEEE Symposium on Security and Privacy, 1–16.
[16] Jaquith, A. (2007). Security metrics: Replacing fear, uncertainty, and doubt. Addison-Wesley.
[17] Pendleton, M., Garcia-Luna-Aceves, J. J., et al. (2016). A survey on systems security metrics. ACM Computing Surveys, 49(4), 1–35.https://doi.org/10.1145/3005714
[18] Tøndel, I. A., Line, M. B., & Jaatun, M. G. (2014). Information security risk assessment: A systematic review. Information and Software Technology, 56(8), 937–951.
[19] Sommestad, T., Karlzén, H., & Hallberg, J. (2013). The theory and practice of security risk assessment. Computers & Security, 32, 1–13.
[20] Pasquale, L., & Spoletini, P. (2012). Process-based compliance for security policies. Requirements Engineering, 17(2), 87–102.
[21] Sgandurra, D., & Lupu, E. (2024). Evolution of automated security assessment and compliance techniques. ACM Computing Surveys, 56(2), 1–36.
[22] OASIS. (2022). Key Management Interoperability Protocol (KMIP) Specification v2.0.https://docs.oasis-open.org/kmip/kmip-spec/v2.0/kmip-spec-v2.0.html
[23] Amazon Web Services. (2023). AWS Key Management Service Developer Guide.https://docs.aws.amazon.com/kms/
[24] Microsoft. (2023). Azure Key Vault security overview.https://learn.microsoft.com/en-us/azure/key-vault/general/secure-key-vault

AUTHORS

Baratov Jasur was born on April 18, 1991, in the Republic of Uzbekistan. He holds the academic title of Associate Professor and is currently affiliated with the Department of Computer Science and Programming at the Jizzakh Branch of the National University of Uzbekistan. His research interests focus on information security, cryptographic policy auditing, encryption and key management systems, security metrics, and decision-oriented security analysis for web and server-based software systems. He is actively involved in both academic research and higher education, with particular emphasis on system-level security modeling and practical security governance.

Ulashev Asror was born on September 7, 1988, in the Republic of Uzbekistan. He holds the academic title of Associate Professor and is currently affiliated with the Department of Computer Science and Programming at the Jizzakh Branch of the National University of Uzbekistan. His research interests include electronic education (e-learning), digital learning platforms, educational information systems,

and the application of information technologies in higher education.

Aynakulov Toxir was born on November 9, 1995, in the Republic of Uzbekistan. He is currently working as an Assistant at the Department of Computer Science and Programming, Jizzakh Branch of the National University of Uzbekistan. His research interests include software development, programming technologies, and the practical application of programming methods in educational and applied software systems.