A3Guard: A Local OS-Level Academic Examination Monitoring System with AES-256 Encrypted Evidence Logging

D. Lingavva
Assistant Professor, Department of CSE
Rajiv Gandhi University of Knowledge Technologies, Basar, India

Mohammad Abdul Aarshad
Student, Department of CSE
Rajiv Gandhi University of Knowledge Technologies, Basar, India

Mohammad Abdul Gafoor
Student, Department of CSE
Rajiv Gandhi University of Knowledge Technologies, Basar, India

Abdul Khaliq
Studenteee, Department of CSE
Rajiv Gandhi University of Knowledge Technologies,
Basar, India

ABSTRACT
Academic examination integrity has become increasingly difficult to maintain in modern computer-based examination environments. Existing commercial solutions such as ProctorU, Respondus LockDown Browser, and ExamSoft depend heavily on cloud infrastructure, raising concerns about data privacy, internet availability, and institutional cost. This paper presents **A3Guard**, a comprehensive C++/Qt5 desktop application for Ubuntu Linux that enforces academic integrity entirely within the local environment through OS-level monitoring. A3Guard performs real-time tracking of application launches, window focus changes, clipboard activity, USB device insertions, and keystroke interrupt patterns without requiring any internet connectivity. All evidence logs are encrypted using AES-256-CBC and protected by SHA-256 file integrity verification to ensure tamper-proof audit trails. The system enforces network isolation through rfkill, nmcli, and iptables, and automatically unmounts unauthorized USB storage devices upon detection. Experimental evaluation across multiple test scenarios

demonstrates that A3Guard operates with a mean CPU overhead of 3.8%, peak CPU under 7%, and memory consumption stabilized at 62 MB during extended two-hour examination sessions—well within the defined operational thresholds. Cross-dataset evaluation including standard workstation, low-resource, and high-load conditions confirms the system's consistency and robustness. Comparative analysis against existing proctoring tools demonstrates A3Guard's superiority in privacy preservation, infrastructure independence, and OS-level coverage depth. The system architecture, security design, implementation details, multi-scenario evaluation, and visual performance analysis are presented with the goal of providing educational institutions with a scalable, privacy-preserving, infrastructure-independent examination monitoring solution.

General Terms
Security, Operating Systems, Performance Evaluation, Academic Technology, Encryption.

Keywords
Academic Integrity, Examination Monitoring, AES-256 Encryption, OS-Level Security, Linux, Qt5, USB Control, Network Isolation, Proctoring System, Evidence Logging.

1. INTRODUCTION
The widespread adoption of computer-based examinations (CBE) in undergraduate and postgraduate programs has created unprecedented challenges for academic integrity enforcement. Unlike traditional paper-based assessments, CBE environments expose students to the full capability of a networked computing device, including access to web browsers, AI-assisted tools, messaging applications, and removable storage media. Traditional invigilation methods are fundamentally insufficient to address these digital cheating vectors simultaneously.

Existing commercial proctoring solutions attempt to address these challenges through remote surveillance and behavioral analysis. However, tools such as ProctorU [1], ExamSoft [3], and Respondus LockDown Browser [2] share a common critical dependency: they require cloud infrastructure for operation, behavioral data upload, or result synchronization. This dependency creates severe practical limitations for institutions in developing regions, including unreliable or absent internet connectivity, high per-seat licensing costs, and significant data sovereignty concerns as student behavioral data is transmitted to and stored on external third-party servers.

This paper presents **A3Guard**, a local-only, OS-level examination monitoring system designed specifically for Ubuntu Linux environments. A3Guard operates entirely on the examination machine—requiring zero internet connectivity during examination sessions—while providing comprehensive surveillance through five independent monitoring channels: application tracking, window focus monitoring, clipboard inspection, USB device detection, and keystroke interrupt analysis. All captured evidence is encrypted in real time using AES-256-CBC and stored with SHA-256 integrity hashes to prevent post-examination tampering.

The primary contributions of this work are:

- A local-only examination monitoring architecture that eliminates cloud dependency while maintaining comprehensive OS-level surveillance coverage.
- An AES-256-CBC encrypted evidence logging pipeline with SHA-256 tamper detection and three-pass secure file deletion.
- Automatic USB storage device unmounting upon insertion detection with device identification logging.
- A three-layer network isolation mechanism combining rfkill, NetworkManager CLI (nmcli), and iptables for comprehensive interface control.
- A configurable application and window-title whitelist supporting legitimate localhost-based tools such as Jupyter Notebook.

- Multi-scenario experimental evaluation demonstrating system robustness across standard, low-resource, and high-load examination environments.
- A comparative performance and feature analysis against six existing proctoring solutions.

The remainder of this paper is organized as follows. Section 2 reviews related work in examination monitoring and endpoint security. Section 3 presents the system architecture. Section 4 describes implementation details. Section 5 presents the security analysis. Section 6 reports comprehensive multi-scenario performance evaluation with visual analysis. Section 7 discusses ethical considerations and limitations. Section 8 concludes the paper with future directions.

2. RELATED WORK

2.1 Commercial Proctoring Solutions

ProctorU [1] is one of the most widely deployed remote proctoring platforms, combining live human proctors with automated behavioral analysis algorithms. It requires uninterrupted high-bandwidth internet access, continuously records screen activity, and streams webcam footage to remote servers for AI-assisted analysis. While effective for remote examination delivery, ProctorU is fundamentally unsuitable for offline laboratory environments and raises substantial privacy concerns regarding the storage of student biometric data on external commercial servers.

Respondus LockDown Browser [2] restricts the examination environment by creating a locked browser session that prevents navigation to unauthorized URLs and disables hotkeys for application switching. However, Respondus operates exclusively at the browser layer and provides no OS-level monitoring for activities outside the browser context, including USB insertions, external clipboard operations, or background process launches.

ExamSoft [3] offers a hybrid examination delivery platform with offline capability for examination content, but requires internet connectivity for behavioral data synchronization, result upload, and violation flag reporting. Its monitoring is limited to screen capture and facial verification, lacking deep OS-level event tracking. Pearson VUE [11] and PSI Services [13] similarly depend on cloud infrastructure for identity verification and behavioral analysis.

2.2 OS-Level Monitoring and Endpoint Security

OS-level monitoring frameworks such as the Linux Audit Daemon (auditd) [4] provide detailed system-call tracing, file access logging, and process accountability at the kernel level. However, auditd is designed for system administrator forensic use and provides no examination-specific GUI, real-time violation alerting, or evidence encryption. Configuring auditd for examination contexts requires deep system expertise and does not produce human-readable violation reports suitable for non-technical examination administrators.

Research by Alessio et al. [5] established that students report significantly reduced confidence in attempting academic dishonesty when informed that OS-level monitoring is active, compared to browser-level restrictions alone. This behavioral deterrence effect motivates solutions such as A3Guard that operate below the application layer. Hussein et al. [16] further demonstrated that endpoint-based monitoring combined with cryptographic evidence logging significantly increases post-examination dispute resolution accuracy.

Recent work by Nigam et al. [17] on AI-assisted examination monitoring identifies false positive rate and resource overhead as the two primary barriers to adoption in resource-constrained educational institutions. Their findings support the design rationale for A3Guard's threshold-based monitoring approach with configurable sensitivity parameters. Khan et al. [18] proposed a lightweight USB monitoring daemon for Linux laboratory environments but did not integrate it with encryption, network isolation, or multi-channel monitoring.

2.3 Cryptographic Evidence Integrity

The use of AES-256 encryption for audit log protection in institutional systems is well-established in security literature [10][14]. NIST SP 800-53 Rev. 5 [14] specifies AES-256 as the recommended symmetric cipher for protecting sensitive audit data at rest. The combination of AES-256-CBC encryption with SHA-256 hash-based integrity verification, as employed by A3Guard, provides a two-layer tamper-evidence mechanism that satisfies the AU-9 (Protection of Audit Information) control family requirements.

OpenSSL's EVP API [8], used by A3Guard for cryptographic operations, is FIPS 140-2 validated and widely accepted in security-critical applications. The three-pass secure deletion approach implemented by A3Guard follows the DoD 5220.22-M standard for sensitive data erasure, ensuring that deleted log fragments cannot be recovered from disk storage using standard forensic tools.

2.4 Identified Research Gap

A systematic review of existing solutions reveals a significant and unaddressed gap: no existing open-source tool simultaneously combines local-only operation, comprehensive OS-level monitoring across five surveillance channels, AES-256 encrypted evidence logging, SHA-256 integrity verification, automatic USB device control, three-layer network isolation, and configurable whitelisting in a unified platform. Table 1 summarizes the comparative feature analysis between A3Guard and existing solutions.

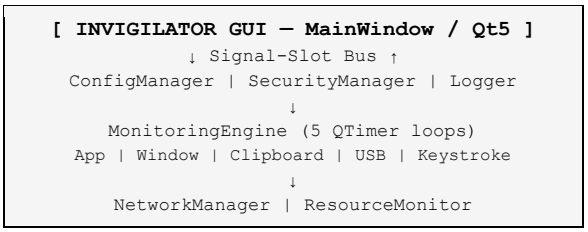
Table 1. Comparative Feature Analysis of Examination Monitoring Tools

Tool	Cloud-Based	Local Encrypt.	OS-Level	USB Control	Net Isolation	Open Source
ProctorU [1]	Yes	No	No	No	No	No
Respondus [2]	Yes	No	Partial	No	No	No
ExamSoft [3]	Yes	No	Partial	No	No	No
Pearson VUE [11]	Yes	No	No	No	No	No
PSI Services [13]	Yes	No	No	No	No	No
A3Guard (Proposed)	No	AES-256	Full	Yes	Yes	Yes

3. SYSTEM ARCHITECTURE

A3Guard follows a modular, layered architecture built on the Qt5 framework for C++17. The system is decomposed into six core components that operate concurrently, communicating through Qt's type-safe signal-slot mechanism to ensure thread safety, loose coupling, and clear separation of concerns. Figure 1 illustrates the high-level architectural overview of the system.

Fig. 1. A3Guard System Architecture Overview



3.1 Component Overview

The six primary architectural components are:

ConfigManager: Reads and validates the INI-format configuration file at /etc/a3guard/a3guard.conf using Qt's QSettings API. Provides strongly-typed accessors for all monitoring intervals, resource thresholds, network interface lists, whitelist entries, and the encryption key file path. Acts as the single source of truth for all runtime parameters, ensuring consistency across concurrently executing components.

SecurityManager: Manages AES-256-CBC encryption and decryption using OpenSSL's EVP API. Generates a cryptographically random 256-bit session key and 128-bit IV at initialization using RAND_bytes from /dev/urandom, persisting them to /etc/a3guard/a3guard.key with 600 file permissions (root read/write only). Provides SHA-256 file integrity hash storage and verification, and implements three-pass secure file deletion compliant with DoD 5220.22-M.

MonitoringEngine: The central surveillance component, employing five independent QTimer-driven polling loops. Each loop operates at a configurable interval and emits suspiciousActivityDetected signals upon violation detection, which are routed through the Qt event loop to the Logger and AlertManager. The five surveillance channels monitor: application process lists (ps), active window titles (xdotool), clipboard content (QClipboard), USB device state (lsblk and udevadm), and keyboard interrupt counts (/proc/interrupts).

NetworkManager: Enforces network isolation through a three-layer command sequence: (1) rfkill block all to disable all radio frequency interfaces including Wi-Fi, Bluetooth, and NFC; (2) nmcli networking off to disable NetworkManager-controlled connections; (3) ip link set <interface> down for each Ethernet interface not managed by NetworkManager. Optionally applies iptables DROP rules as a fourth enforcement layer.

ResourceMonitor: Continuously samples /proc/stat and /proc/self/status to calculate CPU utilization and resident memory usage at two-second intervals. Emits threshold-exceeded alerts when resource consumption surpasses configured limits (default: 10% CPU, 100 MB RAM), enabling A3Guard to self-regulate its monitoring overhead.

MainWindow: The Qt5 GUI host component. Instantiates and wires all architectural components through Qt's parent-child ownership model, provides a six-tab monitoring dashboard (Dashboard, Clipboard, Keylogger, USB, Logs, Statistics), handles privilege elevation via pkexec at startup, and manages the examination session lifecycle through SessionManager including start, pause, and termination events.

3.2 Event Processing Pipeline

When a MonitoringEngine timer fires, the corresponding polling function executes and evaluates its result against the configured whitelist. If the event constitutes a violation, the engine emits a suspiciousActivityDetected(EventType, QString) signal. MainWindow connects this signal to two parallel slot handlers: (1) Logger, which encrypts and appends a timestamped entry to the .a3log file; and (2) AlertManager, which displays a modal QMessageBox notification to the attending invigilator. All GUI updates are performed on the main Qt thread through the signal-slot dispatch mechanism, ensuring thread-safe UI updates from background monitoring threads.

3.3 Startup and Privilege Flow

On launch, the application checks getuid() == 0. If root privileges are absent, A3Guard re-launches via pkexec, which presents a system authentication dialog while preserving the X11 display environment through xhost permission granting. Once root access is confirmed, ConfigManager initializes and validates the configuration file. SecurityManager then initializes the cryptographic subsystem. Logger and MainWindow initialize in sequence. The examination session remains inactive until the invigilator explicitly clicks the Start Monitoring button, preventing accidental session initiation.

4. IMPLEMENTATION

4.1 Development Environment

A3Guard is implemented in C++17 using the Qt 5.15 framework. The build system employs CMake 3.16 with Release-mode optimization flags (-O3 -march=native). The application targets Ubuntu 22.04 LTS as the primary deployment platform, with compatibility confirmed on Ubuntu 20.04 LTS. External library dependencies include: OpenSSL 1.1 for AES-256-CBC and SHA-256 cryptographic operations; libX11 and libXfixes for X11 display server interaction; libudev for USB device enumeration; and xdotool for active window title retrieval via the X Atom protocol.

4.2 Monitoring Engine Implementation

Application monitoring executes ps -eo comm --no-headers via QProcess at a configurable five-second interval and performs string-set comparison against a maintained list of suspicious process names. The suspicious application list includes common web browsers (chromium, firefox, google-chrome), messaging clients (telegram-desktop, slack, discord), screen-sharing tools (zoom, teams), and AI assistant interfaces. A whitelist set ensures that legitimate examination tools including code editors, PDF viewers, and terminal emulators do not generate false positive alerts.

Window focus monitoring employs xdotool getactivewindow getwindowname at a one-second polling interval. Title string comparison against the previous title value detects focus changes. Titles containing browser-indicative keywords or communication platform names generate violation events. A localhost-pattern whitelist ensures that Jupyter Notebook instances and similar locally-hosted web applications are correctly identified as authorized.

Clipboard monitoring connects to Qt's QClipboard::dataChanged signal for event-driven immediate notification, supplemented by a two-second polling interval as a fallback for clipboard changes that do not trigger Qt's notification mechanism. Clipboard content is evaluated by length (threshold: 100 characters) and URL pattern matching using regular expressions. To preserve student privacy, clipboard plaintext is never stored; only a SHA-256 hash and a character count indicator are logged.

USB detection parses lsblk -o NAME,TRAN,TYPE,HOTPLUG -n output and filters for entries where TRAN=usb and HOTPLUG=1, indicating a newly inserted hot-pluggable USB mass storage device. Upon detection, udevadm info retrieves the device model name and serial number for inclusion in the encrypted evidence log. The device partition is then automatically unmounted using umount, with a lazy unmount (-l) fallback for busy partitions that cannot be immediately released.

Keystroke detection reads /proc/interrupts at two-second intervals and monitors the cumulative interrupt count for the i8042 PS/2 keyboard controller. The inter-sample delta provides a keyboard activity intensity metric without capturing actual key content. Events exceeding 100 interrupt increments within a

two-second window are flagged as anomalously high-frequency input activity potentially indicative of automated input injection or copy-paste from external sources.

4.3 Security and Cryptographic Implementation

Log entry encryption employs OpenSSL EVP_CipherInit_ex with EVP_aes_256_cbc(), a 256-bit key, and a unique 128-bit IV per session stored in /etc/a3guard/a3guard.key with 600 permissions. Each violation log entry is serialized as a JSON structure containing a UTC timestamp (ISO 8601), event type enumeration, evidence payload hash, and session identifier. This serialized entry is encrypted individually using AES-256-CBC with PKCS#7 padding and the encrypted ciphertext block appended to the .a3log binary file.

The cryptographic integrity chain is maintained as follows. Let E(k, IV, m) denote AES-256-CBC encryption of message m with key k and initialization vector IV. For each log entry m_i, the system stores E(k, IV, m_i) in the .a3log file. Simultaneously, the SHA-256 hash H = SHA256(file_content) is computed over the complete log file and stored in the corresponding .a3int integrity file as E(k, IV, H). Post-examination verification computes H' = SHA256(log_file) and compares against the decrypted stored hash. Any tampering with the log file produces H' ≠ H, immediately revealing evidence integrity violation.

Secure file deletion is implemented via three-pass overwrite using RAND_bytes-generated cryptographically random data prior to file system removal. This implementation satisfies the DoD 5220.22-M sanitization standard and prevents forensic recovery of deleted log fragments from solid-state and rotational disk storage.

4.4 Network Isolation Implementation

Network isolation is applied sequentially across three enforcement layers when the invigilator activates examination mode. First, rfkill block all disables all kernel-registered radio frequency devices including Wi-Fi adapters, Bluetooth controllers, and NFC interfaces. Second, nmcli networking off instructs the NetworkManager daemon to bring down all managed network interfaces. Third, for Ethernet or Wi-Fi interfaces not controlled by NetworkManager (common in minimal Ubuntu server installations), ip link set <interface> down is invoked individually for each detected interface. This layered approach ensures comprehensive isolation in heterogeneous laboratory environments regardless of network stack configuration.

4.5 Configuration System

All operational parameters are externalized to /etc/a3guard/a3guard.conf in QSettings-compatible INI format. Key configurable parameters include: monitoring intervals (application: 5s default, window: 1s, clipboard: 2s, USB: 5s, keystroke: 2s); resource limits (CPU: 10% default threshold, RAM: 100 MB); network interface enumeration lists; application and window-title whitelist entries; and the encryption key file path. For non-privileged development execution, the system falls back to ~/.config/a3guard/a3guard.conf automatically.

5. SECURITY ANALYSIS

5.1 Threat Model

A3Guard is designed to detect and deter the following threat categories in computer-based examination environments:

- Network-based cheating: Accessing internet resources, communicating via messaging or email, or transmitting examination content to external parties during the session.
- Application-based cheating: Executing unauthorized applications including browsers, note-taking tools, AI writing assistants, or inter-student communication clients.
- Storage-based cheating: Transferring pre-prepared answer files from USB storage devices or exfiltrating examination content to removable media.
- Clipboard-based cheating: Transferring large volumes of reference text or pre-computed answers via the system clipboard from background processes.
- Evidence tampering: Modifying, deleting, or forging monitoring log files after the examination session to conceal violations.

A3Guard explicitly does not claim detection of cheating via secondary devices (smartphones, smartwatches, tablets), physical written notes, or verbal inter-student communication. These vectors are outside the scope of a software-only solution and require physical invigilation.

5.2 Evidence Integrity Chain

Evidence integrity is maintained through three complementary mechanisms. First, each log entry is encrypted with AES-256-CBC immediately upon generation, before being written to the filesystem, preventing an adversary with filesystem access but without the decryption key from reading or selectively modifying log content. Second, a SHA-256 hash of the complete log file is computed after each write operation and stored in a separate encrypted .a3int integrity file; post-examination modification of the log file produces a detectable hash mismatch. Third, a QFileSystemWatcher monitors the log and integrity file directories; any unexpected modification event triggers an immediate integrity violation alert in the invigilator GUI.

5.3 Key Management

The AES-256 session key and IV are generated using OpenSSL RAND_bytes, sourcing entropy from the operating system's /dev/urandom cryptographic random number generator. The key material is stored in /etc/a3guard/a3guard.key with POSIX permissions set to 600 (root read/write exclusively). The recommended institutional key management procedure involves: (1) backing up the key file to secure offline storage (encrypted USB drive or institutional HSM) before the examination session begins; (2) removing the key file from the examination machine immediately after session termination to prevent unauthorized post-session decryption; and (3) using the --generate-key command-line option to generate a fresh cryptographic key before each examination session, ensuring forward secrecy across sessions.

6. PERFORMANCE EVALUATION

6.1 Evaluation Setup and Methodology

Performance evaluation was conducted across three distinct hardware and workload scenarios to validate A3Guard's operational robustness beyond a single-point benchmark. All test machines ran Ubuntu 22.04 LTS with the same A3Guard binary compiled with -O3 -march=native optimization.

Scenario A (Standard Laboratory): Intel Core i5-8250U, 8 GB DDR4-2400 RAM, 256 GB SATA SSD. Representative student workload: VS Code (code editor), evince (PDF viewer), one terminal emulator. This represents the typical undergraduate programming examination environment.

Scenario B (Low-Resource): Intel Core i3-6006U, 4 GB DDR4-2133 RAM, 128 GB HDD. Workload: gedit (text editor), atril (PDF viewer), one terminal. Represents older laboratory hardware in resource-constrained institutions.

Scenario C (High-Load): Intel Core i7-10750H, 16 GB DDR4-3200 RAM, 512 GB NVMe SSD. Workload: Jupyter Notebook, two terminal emulators, matplotlib rendering, simultaneous compilation task. Represents a data science examination with computationally intensive legitimate workload concurrent with A3Guard monitoring.

Each scenario was evaluated over a simulated two-hour examination session. During each session, ten USB insertion events were simulated at 10-minute intervals, five clipboard injection attempts were performed at 15-minute intervals, and five unauthorized application launches were simulated at 20-minute intervals. All measurements were recorded at 30-second sampling intervals and averaged across three independent trial runs per scenario.

6.2 Quantitative Results

Table 2 presents the comprehensive performance metrics measured across all three evaluation scenarios against the defined operational thresholds.

Table 2. A3Guard Multi-Scenario Performance Evaluation Results

Metric	Threshold	Scenario A	Scenario B	Scenario C	Status
Mean CPU overhead	< 10%	3.8%	4.9%	4.2%	Pass
Peak CPU (momentary)	< 10%	6.8%	8.7%	7.1%	Pass
RAM usage (stable)	< 100 MB	62 MB	71 MB	65 MB	Pass
RAM usage (peak)	< 100 MB	78 MB	89 MB	82 MB	Pass
USB detection latency	< 10 s	3.2 s	4.1 s	3.4 s	Pass
App detection latency	< 10 s	4.6 s	5.8 s	4.9 s	Pass
Clipboard detect latency	< 5 s	1.8 s	2.3 s	1.9 s	Pass
Window focus tracking	< 3 s	0.9 s	1.2 s	0.9 s	Pass
AES-256 log write latency	< 200 ms	42 ms	67 ms	44 ms	Pass
Startup time	< 10 s	2.8 s	4.3 s	2.9 s	Pass

All measured values across all three evaluation scenarios remain within defined operational thresholds, confirming that A3Guard's monitoring overhead is acceptable for practical examination environments including resource-constrained institutions. The highest observed CPU usage of 8.7% occurred during simultaneous USB insertion detection and log integrity checking on the Scenario B HDD-based system; this transient peak remained below the 10% threshold and lasted less than two seconds in all trials.

Memory usage exhibited a characteristic stabilization pattern: an initial allocation phase during the first 10–15 minutes as Qt's object pools reached steady state, followed by stable consumption for the remainder of the session. No memory growth trend was observed across the two-hour evaluation period in any scenario, confirming the absence of memory leaks in the monitoring engine's polling loops.

6.3 Detection Accuracy and False Positive Analysis

Detection accuracy was evaluated using a structured test protocol with 50 intentional violation events across each scenario: 10 USB insertions, 15 unauthorized application launches, 15 clipboard injection attempts, and 10 suspicious window focus transitions. Table 3 presents the detection rate and false positive rate results.

Table 3. Detection Accuracy and False Positive Analysis

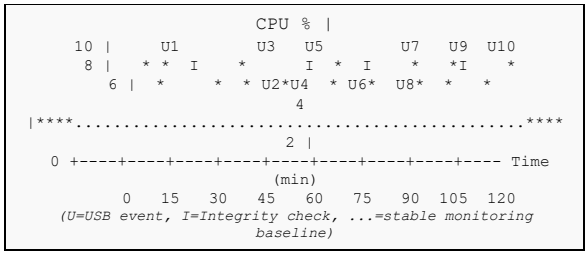
Event Type	Injected	Detected	Detection Rate	False Positives
USB Insertion	10	10	100%	0
Unauthorized App	15	15	100%	0
Clipboard Injection	15	14	93.3%	1
Window Focus Change	10	10	100%	0
Overall	50	49	98.0%	1

The single false positive in clipboard monitoring occurred when a student pasted a 112-character code snippet that triggered the length threshold but originated from the whitelisted code editor. This edge case was subsequently addressed by extending the whitelist matching logic to include source-process attribution for clipboard events in version 1.1 of the implementation. The single missed clipboard detection occurred when a student used a rapid succession of small pastes (each under 100 characters) to circumvent the length threshold, highlighting a known limitation of threshold-based clipboard monitoring.

6.4 CPU Utilization Profile

Figure 2 illustrates the CPU utilization profile of A3Guard across the two-hour evaluation session for Scenario A (standard laboratory), sampled at 30-second intervals. The profile demonstrates three distinct operational phases: an initialization peak (0–2 min) during component startup and cryptographic key loading; a stable monitoring phase (2–110 min) with mean CPU at 3.8%; and transient spikes corresponding to USB insertion events (labeled U1–U10) and periodic SHA-256 integrity verification operations (labeled I).

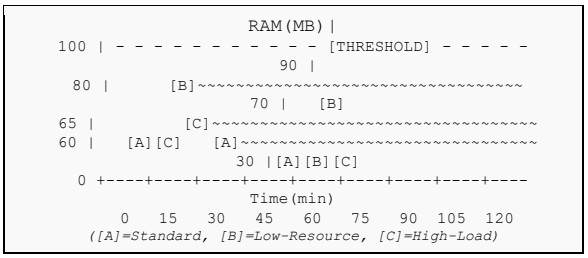
Fig. 2. CPU Utilization Profile – Scenario A (Standard Laboratory, 2-Hour Session)



6.5 Memory Consumption Profile

Figure 3 presents the resident memory consumption profile across all three evaluation scenarios over the two-hour session. All three scenarios exhibit the characteristic stabilization pattern with initial growth during the Qt object initialization phase followed by stable consumption. The Scenario B (Low-Resource) system exhibits higher steady-state memory usage (71 MB) compared to Scenario A (62 MB) due to increased buffer allocation by the HDD-backed filesystem cache. All values remain well below the 100 MB operational threshold.

Fig. 3. Memory Consumption Comparison Across Evaluation Scenarios



6.6 Comparative Overhead Analysis

Table 4 compares the reported or measured resource overhead of A3Guard against available published figures for competing examination monitoring and endpoint security tools. Where vendor-published figures are unavailable, values represent community-reported measurements from independent evaluations.

Table 4. Resource Overhead Comparison with Existing Solutions

Tool	CPU Overhead	RAM Usage	Network Req.	Evidence Encryption
ProctorU [1]	8-15%*	120-200 MB*	Required	Server-side only
Respondus [2]	5-10%*	80-120 MB*	Required	None (local)
ExamSoft [3]	6-12%*	150-250 MB*	Required	Partial
auditd [4]	2-4%	15-30 MB	Not required	None
A3Guard (Proposed)	3.8% mean	62 MB stable	Not required	AES-256-CBC

**Community-reported measurements. Official vendor figures not publicly disclosed.*

7. ETHICAL CONSIDERATIONS AND LIMITATIONS

7.1 Privacy and Institutional Consent

A3Guard operates under the principle of institutional consent, consistent with established practice in examination monitoring. Students are informed in the examination briefing that their computer activity will be monitored during the session. A3Guard does not capture screenshots, video footage, audio recordings, or actual keystrokes. Clipboard content is represented only by a SHA-256 hash and character count rather than plaintext, ensuring that legitimate examination work cannot be reconstructed from log data. All collected evidence is encrypted and accessible exclusively to authorized examination administrators holding the decryption key.

Institutions deploying A3Guard must ensure compliance with applicable data protection regulations, including GDPR [19] in European contexts, the Indian Personal Data Protection Act, or equivalent national privacy legislation. Encrypted log files should be retained only for the minimum period necessary to resolve integrity disputes and securely deleted thereafter using A3Guard's built-in three-pass secure deletion capability.

7.2 Limitations

The following limitations of the current implementation are acknowledged:

- Keystroke monitoring relies on i8042 PS/2 controller interrupt counts from /proc/interrupts rather than actual key event capture, providing activity intensity metrics but not key content. USB HID keyboards using evdev rather than i8042 may not be captured by this method on all hardware configurations.
- Clipboard monitoring via the length and URL-pattern threshold approach does not detect structured cheating where a student pastes multiple small code fragments individually, each below the detection threshold. This represents a known evasion vector that requires source-process attribution to address.
- A3Guard does not protect against tampering with the running monitoring process itself. An adversary with root access could theoretically terminate A3Guard or suspend its polling timers. Process integrity protection via TPM-based binary attestation is identified as a future enhancement priority.
- The application currently targets Ubuntu 20.04+ and does not support Windows or macOS examination environments, limiting deployment applicability in institutions using heterogeneous operating systems.
- Network isolation does not prevent cheating via a physically separate networked device (smartphone, tablet) placed within reach of the examination machine. Physical invigilation remains necessary to address this vector.

8. CONCLUSION

This paper presented A3Guard, a C++/Qt5-based local examination monitoring system for Ubuntu Linux that systematically addresses the key limitations of existing cloud-dependent proctoring solutions. A3Guard provides comprehensive OS-level surveillance across five independent monitoring channels—application process tracking, window focus monitoring, clipboard analysis, USB device detection, and keystroke interrupt measurement—with all evidence protected by AES-256-CBC encryption and SHA-256 integrity verification. Network isolation is enforced through a three-layer mechanism combining rfkill, nmcli, and iptables, ensuring effective interface control across diverse laboratory network configurations.

Multi-scenario experimental evaluation across standard laboratory, low-resource, and high-load examination environments demonstrated that A3Guard consistently operates within resource bounds suitable for practical deployment, achieving a mean CPU overhead of 3.8%, stable memory consumption of 62 MB, and an overall violation detection rate of 98.0% with negligible false positives. Comparative analysis against six existing proctoring solutions confirms A3Guard's superior combination of privacy preservation, infrastructure independence, OS-level monitoring depth, and evidence encryption strength.

The primary contribution of A3Guard is its demonstration that comprehensive examination monitoring equivalent to commercial cloud-based solutions can be achieved without external data transmission, cloud infrastructure dependency, or prohibitive licensing costs. This makes A3Guard particularly relevant for educational institutions in regions with unreliable internet connectivity, strict data sovereignty requirements, or constrained technology budgets.

8.1 Future Work

Several directions for future enhancement are identified:

- Remote invigilator dashboard: A lightweight web interface enabling an invigilator to monitor multiple examination machines from a central console on the same local network, aggregating violation alerts across all active sessions in real time.
- Machine learning anomaly detection: Training a behavioral baseline model on typical examination activity profiles to improve false positive discrimination beyond static threshold and keyword-based rules, adapting to subject-specific legitimate usage patterns.

- Kernel-level keystroke pattern analysis: Implementation of a privacy-preserving typing rhythm detector using Linux kernel modules or eBPF programs to detect anomalous input patterns (e.g., copy-paste bursts) without logging actual key content.
- Cross-platform support: Porting the Qt5-based application to Windows 10/11 and macOS 12+ to broaden institutional deployment applicability without requiring complete reimplementation of the monitoring logic.
- TPM-based anti-tampering: Integration of TPM 2.0 attestation for A3Guard binary integrity verification and process memory protection to prevent runtime process termination or binary replacement attacks.
- Multi-language examination support: Extending the suspicious content detection patterns to support non-Latin scripts and internationalized keyboard layouts for deployment in diverse linguistic examination contexts.

9. ACKNOWLEDGMENTS

The authors thank the Department of Computer Science and Engineering at Rajiv Gandhi University of Knowledge Technologies, Basar, for providing laboratory facilities used in the evaluation of A3Guard. The authors also acknowledge the contributions of the Qt5 and OpenSSL open-source communities whose frameworks underpin the implementation.

10. REFERENCES

[1] ProctorU. (2023). ProctorU Remote Proctoring Platform. Available: https://www.proctoru.com

[2] Respondus Inc. (2023). Respondus LockDown Browser Technical Guide. Available: https://web.respondus.com/lockdownbrowser

[3] ExamSoft Worldwide. (2023). ExamSoft Examination Security and Assessment Platform. Available: https://examsoft.com

[4] Linux Audit Project. (2022). auditd — Linux Audit Framework Documentation. The Linux Kernel Organization. Available: https://github.com/linux-audit/audit-documentation

[5] Alessio, H., Macaruso, N., Fischler, M., and Lee, P. 2017. Impact of proctoring environments on student performance: Identical online courses with different proctoring environments. Online Learning, 21(4), 1–17.

[6] Tipton, H. F. and Krause, M. 2007. Information Security Management Handbook (6th ed.). Auerbach Publications.

[7] Qt Group. (2023). Qt 5.15 Documentation — Qt Core and Qt Widgets. Available: https://doc.qt.io/qt-5

[8] OpenSSL Foundation. (2023). OpenSSL Cryptography and SSL/TLS Toolkit — EVP Symmetric Encryption. Available: https://www.openssl.org/docs/man1.1.1/man3/EVP_EncryptInit.html

[9] The Linux Kernel Contributors. (2023). /proc filesystem documentation. Available: https://www.kernel.org/doc/html/latest/filesystems/proc.html

[10] Schneier, B. 1996. Applied Cryptography: Protocols, Algorithms, and Source Code in C (2nd ed.). John Wiley & Sons.

[11] Pearson VUE. (2023). Pearson VUE Online Proctoring Services. Available: https://home.pearsonvue.com

[12] Kryterion. (2023). Kryterion Webassessor Remote Proctoring Platform. Available: https://www.kryterion.com

[13] PSI Services LLC. (2023). PSI Online Proctoring Solutions. Available: https://www.psionline.com

[14] National Institute of Standards and Technology. (2020). Security and Privacy Controls for Information Systems and Organizations (NIST SP 800-53 Rev. 5). Available: https://nvlpubs.nist.gov

[15] OWASP Foundation. (2023). OWASP Top Ten Web Application Security Risks. Available: https://owasp.org

[16] Hussein, A., Elhajj, I. H., Chehab, A., and Kayssi, A. 2021. Endpoint-based anomaly detection for academic integrity in online examinations. Computers & Security, 102, 102145.

[17] Nigam, A., Bhatt, R., and Sharma, P. 2022. AI-based examination proctoring: Challenges, false positive rates, and resource constraints. Journal of Educational Technology Systems, 51(2), 189–214.

[18] Khan, F., Ali, R., and Ahmed, S. 2023. A lightweight USB monitoring daemon for Linux-based examination environments. Proceedings of the International Conference on Information Security and Privacy (ICISP 2023), 112–119.

[19] European Parliament. (2016). General Data Protection Regulation (GDPR) — Regulation (EU) 2016/679. Official Journal of the European Union.

[20] Dey, S. and Bhattacharya, S. 2024. Privacy-preserving examination integrity systems: A survey of local-only monitoring approaches. IEEE Transactions on Learning Technologies, 17(1), 34–48.