The OSCAR PUBLISHING Services
American Journal of Applied Science and Technology

VOLUME Vol.05 Issue 11 2025
PAGE NO. 215-222

Lightweight and High-Performance Encryption for Real-Time Multimedia and Embedded Systems: Theory, Implementation Considerations, and Comparative Analysis

John R. Anderson
Department of Computer Science, University of Edinburgh

Received: 01 November 2025; **Accepted:** 15 November 2025; **Published:** 30 November 2025

Abstract: This article presents a comprehensive, publication-ready examination of lightweight and high-performance encryption strategies for real-time multimedia transmission and constrained embedded systems. Beginning with an expansive theoretical framing of the unique security requirements of streaming video, voice, and sensor data, the work synthesizes decades of prior engineering practice and academic study to articulate clear design objectives for encryption schemes in latency-sensitive environments (Aly, 2004; Shi, Changgui & Wang, 2004). We review the cryptographic primitives most relevant to such contexts—symmetric block ciphers and stream ciphers—and discuss implementation trade-offs across software, hardware (FPGA), and mobile environments (Atul et al., 2011; Hoang & Nguyen, 2012; Rouaf & Yousif, 2021). Performance metrics are examined in depth: throughput, latency, computational overhead, memory footprint, energy consumption, and compatibility with compression pipelines (Andriani, Wijayanti & Wibowo, 2018; Advani & Gonsai, 2019). Building on standards and canonical references for AES/Rijndael and guidance on governmental cryptographic deployment (Daemen & Rijmen, 2009; Daemen & Rijmen, 2010; Lee, 2009), we analyze specific MPEG video encryption approaches and lightweight real-time variants with respect to security, perceptual impact, and computational cost (Shi, Changgui & Wang, 2004; Aly, 2004). Case studies and hypotheticals address FPGA acceleration, embedded secure modules for sensor streams, and voice-over-IP protections, drawing on practical implementations and comparative performance evaluations (Atul et al., 2011; Chalermwat et al., 2011; Bassil et al., 2005). Limitations, attack surfaces, and countermeasures—covering chosen-plaintext vulnerabilities, key-management issues, and side-channel leakage—are elaborated with citations to standards, analysis pages, and implementation reports (Cole et al., 2005; Gladman, 2012; CSOR/NIST). The article closes with concrete recommendations for architects of real-time secure transmission systems: when to prefer lightweight cipher variants, how to partition cryptographic tasks between hardware and software, and how to integrate encryption with compression to maintain both confidentiality and streaming performance (Aly, 2004; Shi, Changgui & Wang, 2004; Andriani et al., 2018). This synthesis is intended to guide implementers and researchers toward designs that balance robust security with the demanding performance constraints of modern multimedia and embedded deployments.

Keywords: lightweight encryption, real-time video, AES/Rijndael, FPGA implementation, MPEG encryption, embedded security, performance analysis

INTRODUCTION

Catabolism The increasing ubiquity of streaming multimedia and networked embedded devices has placed security and performance into a tightly coupled relationship. Confidentiality for real-time video, voice streams, and sensor telemetry is no longer an optional feature; regulatory requirements, user privacy expectations, and commercial competitiveness demand that data traversing untrusted networks be protected. At the same time, the real-time nature of multimedia and the resource constraints of embedded platforms—limited CPU cycles, constrained memory, and tight power budgets—mean that conventional cryptographic

approaches, if naively applied, can degrade user experience through latency, jitter, and increased energy consumption (Aly, 2004; Shi, Changgui & Wang, 2004). Recognizing this duality—security versus performance—motivates the precise technical inquiry undertaken in this article: how to design, select, and implement encryption strategies that preserve strong confidentiality guarantees while respecting the tight latency and resource envelopes of real-time multimedia and embedded systems.

The literature provides both conceptual foundations and concrete implementations on which such an inquiry must build. The canonical specification and analysis of the Rijndael cipher—accepted as AES—provides the cryptographic baseline for secure block cipher operation (Daemen & Rijmen, 2009; Daemen & Rijmen, 2010). Practical evaluations of AES implementations in hardware (e.g., FPGA) and comparisons of AES key lengths and modes yield critical performance insights (Andriani, Wijayanti & Wibowo, 2018; Atul et al., 2011; Hoang & Nguyen, 2012). Independent studies focused on multimedia specifically—both algorithmic approaches to MPEG video encryption and lightweight encryption designs for real-time streams—expose unique design constraints and opportunities: some solutions trade partial perceptual degradation for reduced compute, while others embed encryption at the codec stage to minimize extra processing (Shi, Changgui & Wang, 2004; Aly, 2004). Moreover, recent comparative performance analyses across symmetric algorithms on mobile and embedded hardware underscore that the landscape is not monolithic; algorithm choice, implementation style, and integration with hardware acceleration dramatically affect encryption and decryption time (Advani & Gonsai, 2019; Rouaf & Yousif, 2021).

Despite these contributions, a precise synthesis is missing: a framework that systematically aligns security goals (confidentiality levels, threat models) with performance budgets (latency, throughput, energy), and that maps available algorithmic and architectural options to those joint constraints. This gap is the central problem statement of the present work: practitioners require a unified, implementation-aware guideline that permits confident selection and deployment of encryption techniques tailored to real-time multimedia and constrained embedded environments. Such a guideline must be founded on cryptographic soundness, respect engineering realities—including hardware acceleration via FPGA or specialized modules—and offer practical trade-off analyses grounded in prior empirical evaluations (Atul et al., 2011; Chalermwat et al., 2011; Andriani et al., 2018).

The contribution of this article is threefold. First, we produce an integrative theoretical framework linking threat models, desired confidentiality properties, and latency-aware design constraints. Second, we provide a detailed descriptive evaluation of candidate algorithms and implementation strategies—ranging from optimized AES modes to MPEG-aware partial-encryption techniques and FPGA-accelerated implementations—referencing practical performance studies and standards (Daemen & Rijmen, 2009; Shi, Changgui & Wang, 2004; Atul et al., 2011). Third, we synthesize prescriptive recommendations and implementation heuristics that practitioners can immediately apply to system design, including specific guidance on key management, mode selection, and hardware/software partitioning for real-time pipelines (Lee, 2009; Gladman, 2012). Every major claim is anchored to established references to ensure that recommendations are empirically and theoretically grounded (Cole et al., 2005; Bassil et al., 2005).

METHODOLOGY

Our methodology for this integrative study is analytic and comparative rather than experimental: we synthesize and elaborate upon the design, implementation, and empirical evaluations present in the provided literature to produce a coherent set of design prescriptions and theoretical analyses. The approach comprises four interrelated activities: (1) threat and requirement extraction, (2) algorithmic inventory and capability mapping, (3) implementation strategy analysis, and (4) prescriptive synthesis and validation through cross-reference to empirical studies.

Threat and requirement extraction begins by identifying the core confidentiality and integrity goals relevant to streaming multimedia and embedded telemetry. From a security-architecture perspective, a real-time multimedia stream must protect against passive eavesdropping, active tampering (where feasible), key-extraction attacks, and side-channel leakage originating from hardware implementations (Cole et al., 2005). Practically, designers often accept a model in which confidentiality is primary, authenticity is optionally provided by separate lightweight MAC or authenticated-encryption constructions, and timeliness constraints dominate system design choices (Lee, 2009; CSOR/NIST). We map these desired properties into measurable

system-level requirements: maximum allowable end-to-end latency increase induced by cryptographic processing, acceptable throughput reduction, energy-per-bit targets, and memory footprint thresholds. These operational figures are derived from reported performance ranges in mobile and FPGA implementations (Advani & Gonsai, 2019; Atul et al., 2011; Hoang & Nguyen, 2012).

Algorithmic inventory and capability mapping uses the provided corpus to enumerate candidate encryption approaches. This includes standard symmetric block ciphers such as AES/Rijndael in different modes (ECB, CBC, CTR, GCM), lightweight stream-cipher-like transformations applied to compressed video bitstreams, and domain-specific schemes that operate on codec parameters or selective portions of compressed frames (Daemen & Rijmen, 2009; Shi, Changgui & Wang, 2004; Aly, 2004). For every candidate, we analyze theoretical properties—cryptographic strength, susceptibility to particular classes of attacks, and compatibility with low-latency streaming—and link those properties to metrics from empirical studies (Andriani et al., 2018; Advani & Gonsai, 2019).

Implementation strategy analysis examines software-only implementations, hybrid CPU-FPGA acceleration, and mobile-optimized implementations, comparing their strengths and weaknesses with respect to the mapped requirements. Data on FPGA-based AES accelerators and ported AES implementations informs the hardware-software partition recommendations, showing both the magnitude of performance gains and potential complexity in development and verification (Atul et al., 2011; Hoang & Nguyen, 2012; Chalermwat et al., 2011). We also incorporate real-time constraints on codecs, such as the requirement to preserve decoder compatibility and avoid breaking synchronization or adding jitter-inducing buffering (Shi, Changgui & Wang, 2004; Aly, 2004).

Finally, prescriptive synthesis involves assembling decision rules and heuristics for implementers—e.g., when to apply full-stream encryption versus selective encryption, how to choose AES key lengths and modes under latency constraints, and when to offload cryptographic primitives to hardware. Each rule is cross-validated against at least one empirical or standards-based reference (Daemen & Rijmen, 2009; Andriani et al., 2018; Lee, 2009). The methodology deliberately avoids novel experimentation; instead, it amplifies and integrates the validated findings of prior work into a single, actionable narrative.

RESULTS

Because this work is integrative and analytic, the "results" are expressed as structured findings derived from cross-referencing the literature against the system requirements and threat models previously specified. The results are organized as a set of design observations, quantitative performance insights (drawn from empirical studies), and architectural mappings that translate security desiderata into implementation prescriptions.

1. High-level observation: Symmetric encryption remains the pragmatic choice for real-time multimedia. The literature consistently indicates that symmetric algorithms—especially AES (Rijndael)—offer the best balance of cryptanalytic strength and performance when compared to public-key alternatives for bulk data encryption in real-time settings (Daemen & Rijmen, 2009; Daemen & Rijmen, 2010). AES's widespread adoption also provides mature implementations and hardware acceleration paths, a critical factor when designing systems subjected to tight latency and energy budgets (Gladman, 2012; Lee, 2009).

2. MPEG-aware selective encryption can substantially reduce computational load with bounded confidentiality degradation. Multiple studies demonstrate that encrypting critical elements of compressed video (such as header information, motion vectors, or sign bits) can break visual intelligibility while reducing the number of bits that must be processed by the cipher, yielding beneficial throughput and latency outcomes (Shi, Changgui & Wang, 2004; Aly, 2004). The trade-off—partial protection rather than full semantic confidentiality—must be explicitly accepted by system architects and evaluated against threat models since selective encryption leaves some information intact (Shi, Changgui & Wang, 2004).

3. FPGA acceleration of AES provides significant speedups, often converting an impractical software-only solution into a deployable low-latency design. Empirical FPGA implementations show orders-of-magnitude improvements in throughput for AES encryption and decryption when properly pipelined and parallelized (Atul et al., 2011; Hoang & Nguyen, 2012). The key caveat is that FPGA development brings additional complexity, increased verification demands, and potential for side-channel leakage if physical implementation details are not carefully

managed (Chalermwat et al., 2011; Cole et al., 2005).

4. AES key length choices must balance security margin and performance. Comparative studies of AES-128, AES-192, and AES-256 indicate measurable performance differences—AES-128 is generally faster and less resource-intensive than its longer-key counterparts—while all three provide robust security under standard threat models (Andriani et al., 2018; Andriani, Wijayanti & Wibowo, 2018). For many streaming applications that require strong but not extreme key lifetimes and where efficient key rotation is possible, AES-128 presents a reasonable default. Nonetheless, system architects must align key management policies and compliance requirements (e.g., governmental mandates) with key-length choices (Lee, 2009).

5. Mobile and embedded CPU performance varies widely; algorithmic choice and implementation technique matter. Studies comparing symmetric encryption algorithms on mobile devices and embedded CPUs report that optimized implementations of AES and other block ciphers can achieve satisfactory encryption and decryption times, but non-optimized or generic implementations suffer significant slowdown (Advani & Gonsai, 2019; Rouaf & Yousif, 2021). The implication is that any deployment targeting mobile or IoT endpoints must invest in code optimization, consider lightweight cipher alternatives if the environment is severely constrained, or provide hardware acceleration when feasible (Hoang & Nguyen, 2012).

6. Modes of operation and authenticated encryption need careful selection in real-time contexts. CTR mode offers desirable parallelizability and low-latency streaming properties but requires strict nonce management. Authenticated encryption modes (e.g., GCM) provide built-in integrity but may increase computational and implementation complexity and require careful handling to avoid performance penalties or catastrophic misuse (Daemen & Rijmen, 2009; Lee, 2009). For streaming, a split approach—fast symmetric encryption for confidentiality paired with compact integrity checks or out-of-band authentication—can sometimes be preferable when absolute minimal latency is the dominant constraint.

7. Side-channel and implementation attacks are practical concerns in FPGA and embedded deployments. The literature underscores that when cryptography migrates into hardware, attackers can exploit timing, power, and electromagnetic emissions. Implementers must consider countermeasures such as masking, constant-time operations, and physical shielding where threat models include local adversaries (Cole et al., 2005; Gladman, 2012).

8. Integration with compression and codec pipelines is critical: encryption should be codec-aware. Studies on MPEG encryption specifically recommend approaches that preserve decoder compatibility while minimizing the need to re-encode or store multiple encrypted streams (Shi, Changgui & Wang, 2004; Aly, 2004). Encrypting after compression avoids computationally expensive re-encoding but requires careful identification of the compressed stream elements that, when encrypted, will render the content intelligible to attackers while leaving decoder state intact.

9. End-to-end system-level considerations—key distribution, revocation, and synchronization—are decisive in real deployments. The technical literature and practical guides emphasize that cryptographic algorithm choice is necessary but not sufficient: systems must implement reliable and low-latency key management, efficient rekeying to limit exposure windows, and robust synchronization to avoid playback artifacts in streaming (Lee, 2009; CSOR/NIST).

These results, rooted in the cited literature, comprise an actionable knowledge base for architects of real-time multimedia encryption systems. Each result is directly traceable to experimental evaluations, algorithmic analyses, or standards-based guidance present in the referenced corpus (Aly, 2004; Shi, Changgui & Wang, 2004; Daemen & Rijmen, 2009; Andriani et al., 2018; Atul et al., 2011).

DISCUSSION

The core tension analyzed throughout this article is the trade-off between cryptographic robustness and resource-constrained performance requirements. The interplay is complex: certain design choices improve raw throughput but produce awkward implications for system security; others safeguard cryptographic integrity but impose latency or energy costs that degrade real-time performance. In this discussion we parse these trade-offs in depth, examine counter-arguments, and outline nuanced implementation recommendations.

1. Full-stream encryption vs. selective encryption: security, performance, and threat modeling. Full-stream encryption (encrypting every payload bit) is

cryptographically straightforward: use a robust block cipher (such as AES) in an appropriate streaming-capable mode, ensuring nonce uniqueness and correct handling of padding and alignment (Daemen & Rijmen, 2009; Lee, 2009). The advantage is maximal confidentiality under conventional threat models. The major drawback is performance cost: encrypting the entire stream consumes CPU cycles and potentially induces latency. Selective encryption—targeting only those parts of a compressed stream that hold most of the perceptual information—reduces computational burden and is attractive for devices with constrained resources (Shi, Changgui & Wang, 2004; Aly, 2004). However, selective encryption exposes a more complex threat surface: adversaries may be able to reconstruct content from unencrypted portions, use statistical inference, or exploit structure within the codec to recover semantics. Therefore, selective encryption is a defensible design choice only when the threat model admits partial confidentiality (e.g., when the system aims to make content non-viewable by casual snoopers rather than robustly withstand determined cryptanalysis). System designers must explicitly articulate and approve this risk trade-off.

2. AES as a pragmatic standard—why it remains the backbone. AES/Rijndael provides a clear reference point: its cryptanalytic robustness is well-established, and it benefits from a vast body of optimized implementations across platforms—software libraries, CPU instruction set support (AES-NI on many processors), and mature FPGA designs (Daemen & Rijmen, 2009; Gladman, 2012; Atul et al., 2011). Adopting AES simplifies compliance and leverages community-tested code paths. The counter-argument is that AES implementations can be heavyweight for ultra-constrained devices; in such cases, lightweight cipher alternatives or tailored selective encryption may be necessary (Advani & Gonsai, 2019). Nonetheless, even lightweight designs must be justified carefully because non-standard ciphers may lack the maturity and analysis that AES enjoys.

3. Hardware acceleration: enormous performance benefits, but higher design cost and side-channel exposure. FPGA-based AES implementations can radically reduce latency and increase throughput through pipelining and parallelism (Atul et al., 2011; Hoang & Nguyen, 2012). For systems where low-latency encryption must be applied to high-bitrate video streams, FPGA acceleration can be the enabling technology. The overheads include development time, hardware cost, and complexity of ensuring side-channel resistance (Chalermwat et al., 2011). Moreover, deploying FPGAs at scale in battery-powered mobile devices may be impractical—specialized hardware accelerators integrated into SoCs, or CPU instruction set features (AES-NI), might be a better engineering compromise (Hoang & Nguyen, 2012).

4. Modes of operation and the latency/parallelism trade-off. The choice of cipher mode exerts a significant influence on latency characteristics. CBC (Cipher Block Chaining) provides good cryptographic diffusion but is inherently sequential and therefore increases encryption latency in streaming contexts; CTR (Counter) mode and some stream-cipher constructions permit parallel encryption and decryption and allow low-latency operation (Daemen & Rijmen, 2009). However, parallelizable modes demand careful nonce/IV management: reuse can be catastrophic, especially in CTR. Authenticated modes like GCM give strong integrity assurances but at a cost; in streaming scenarios where packet loss and re-ordering are common, the performance and complexity of authenticated schemes can be challenging (Lee, 2009). A practical counter-argument is to separate confidentiality from integrity—use fast CTR-mode encryption for payload confidentiality and employ lightweight per-packet signatures or MACs for integrity, though this partitioning adds protocol complexity and must be carefully synchronized (Daemen & Rijmen, 2009).

5. Key management under real-time constraints: rekeying, distribution, and revocation. The security of any symmetric system hinges on robust key management. Frequent rekeying limits exposure to a compromised key but increases control-plane traffic and processing for key distribution. For live streaming, key rotation must be done without introducing playback interruptions or synchronization errors. Solutions include using compact rekeying messages or embedding key update signals within existing control channels. Standards and government guidance emphasize secure key storage and restricted key access while acknowledging the operational difficulty of real-time key rotation in bandwidth-limited channels (Lee, 2009). The counter-argument is that in some constrained systems, rekeying intervals may be lengthened due to operational complexities—this should be mitigated by careful endpoint security and secure key storage.

6. On the desirability of authenticated encryption in streaming: the integrity-confidentiality coupling debate. Authenticated encryption simplifies

correctness: it provides confidentiality and integrity in a single primitive, reducing the complexity of protocol design. However, in streaming, authenticated encryption sometimes imposes buffering or performance penalties. For example, some AEAD schemes require entire frames or groups of frames to be buffered to compute tags, increasing latency. One pragmatic approach is to use AEAD for critical control or metadata channels and fast non-authenticated encryption for raw payloads, combined with lightweight integrity checks at strategic intervals. The trade-off involves accepting limited windows of unverified data in favor of continuous low-latency playback.

7. Side-channel risks and hardware defenses. The migration of cryptographic operations to hardware accelerators brings side-channel risks—power, time, and emanation—to the fore (Cole et al., 2005; Gladman, 2012). Countermeasures include constant-time implementation patterns, masking techniques, and architectural approaches such as redundancy and noise injection. Implementers must reconcile the performance cost of these countermeasures with the security they provide. For highly sensitive applications (e.g., secure government or defense video feeds), the extra overhead is justified; for consumer video streaming, it may be excessive unless local attackers are a realistic threat.

8. Practical heuristics for designers, drawn from empirical studies. Synthesizing empirical performance studies yields several rules-of-thumb:

- Use AES-128 in CTR mode for high-throughput streaming where parallelism is essential and key management supports strict nonce control (Andriani et al., 2018; Daemen & Rijmen, 2009).

- Employ selective encryption for severely constrained endpoints, but restrict such use to threat models permitting partial confidentiality and pair it with additional obfuscation or watermarking to deter reconstruction (Shi, Changgui & Wang, 2004; Aly, 2004).

- Offload to FPGA or hardware accelerators when sustained high-bitrate encryption is required and the development budget permits; otherwise, optimize software implementations and exploit CPU instruction set extensions (Atul et al., 2011; Hoang & Nguyen, 2012).

- Maintain clear separation of confidentiality and long-term key management: use ephemeral session keys for stream encryption and a robust out-of-band mechanism for distributing these keys (Lee, 2009).

9. Limitations of the literature and open gaps. While the cited corpus provides substantial evidence for the above points, several gaps remain. First, there is limited public data on the precise energy and latency profiles of modern SoC-integrated AES accelerators across a broad range of mobile devices; many studies focus on FPGA or desktop-class hardware (Atul et al., 2011; Advani & Gonsai, 2019). Second, the trade-offs for emerging codecs and newer video compression standards require updated empirical validation: MPEG-focused selective encryption studies remain informative but may not directly translate to newer codecs without re-evaluation (Shi, Changgui & Wang, 2004). Finally, there is a shortage of standardized methodology for evaluating perceived confidentiality: quantifying how much visual information remains after selective encryption is inherently subjective and depends on attack models and reconstruction techniques.

LIMITATIONS

This article's scope and method introduce several limitations that must be acknowledged. First, the work is integrative rather than experimental; it synthesizes and elaborates on results available in the provided literature rather than reporting new measurements. Consequently, conclusions about absolute performance—e.g., exact encryption throughput or energy-per-bit numbers—are contingent on the original studies' environments and may not generalize to all modern hardware platforms (Advani & Gonsai, 2019; Rouaf & Yousif, 2021). Second, the reliance on MPEG-era selective encryption research implies that conclusions about selective encryption's efficacy may need updating for newer codecs; the underlying principle—that codec-aware selective encryption can reduce computational load—remains valid, but practitioner application must be codec-specific (Shi, Changgui & Wang, 2004; Aly, 2004). Third, the discussion of side-channel risks and countermeasures is high-level; designing provably secure, side-channel-resistant hardware requires specialized evaluation and often cannot be distilled into general heuristics without detailed physical measurements (Cole et al., 2005; Gladman, 2012). Finally, government and standards guidance change over time; implementers must consult the latest official documents for compliance and best practices (Lee, 2009; CSOR/NIST).

Future Scope

Several future research directions emerge naturally from this synthesis. Empirical benchmarking across modern SoCs and mobile processors with integrated cryptographic accelerators is essential to inform up-to-date trade-offs—data center, edge, and mobile contexts differ substantially and demand distinct engineering decisions. Additionally, formalizing selective encryption quality metrics—combining human perceptual studies with automated reconstruction attacks—would enable more principled selection of selective encryption parameters. Research into lightweight authenticated encryption modes specifically tailored for streaming (minimizing buffering and tag computation latency) could unify confidentiality and integrity without sacrificing low-latency requirements. Finally, advancing low-overhead side-channel countermeasures that provide meaningful protection on FPGAs and low-power SoCs remains a high-impact engineering challenge (Chalermwat et al., 2011; Cole et al., 2005).

CONCLUSION

This work provides an integrative, implementation-aware synthesis of encryption strategies for real-time multimedia and embedded systems. By aligning threat models, codec constraints, and performance metrics with algorithmic and architectural options documented in the literature, we identify robust design patterns for practitioners. AES remains the pragmatic baseline for bulk data encryption (Daemen & Rijmen, 2009; Daemen & Rijmen, 2010), but the practical realities of latency and resource constraints make selective encryption, hardware acceleration, and carefully chosen modes of operation essential components of any serious system design (Shi, Changgui & Wang, 2004; Aly, 2004; Atul et al., 2011). Key management and side-channel mitigation are decisive non-algorithmic factors that must be integrated early in system architecture (Lee, 2009; Cole et al., 2005). While this synthesis cannot substitute for platform-specific benchmarking and threat-specific security analysis, it offers a coherent set of recommendations and theoretical considerations grounded in the referenced corpus to guide designers toward secure, performant real-time encryption deployments.

REFERENCES

1. Salah Aly. “A Light-Weight Encrypting For Real Time Video Transmission”, TR04-002, College of computing and digital media, Depaul University, August 2004. http://facweb.cs.depaul.edu/research/TechReports/TR04-002.pdf
2. B. Shi, W. Changgui and S. Wang. “MPEG Video Encryption Algorithms”, Multimedia Tools and Applications, Vol. 24, Issue 1, pp. 57–79, September 2004.
3. N. A. Advani and A. M. Gonsai. "Performance Analysis of Symmetric Encryption Algorithms for their Encryption and Decryption Time," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), 2019, pp. 359–362.
4. M. T. Rouaf and A. Yousif. "Performance Evaluation of Encryption Algorithms in Mobile Devices," 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), 2021, pp. 1–5. doi: 10.1109/ICCCEEE49695.2021.9429673.
5. Andriani, S. E. Wijayanti and F. W. Wibowo. "Comparison Of AES 128, 192 And 256 Bit Algorithm For Encryption And Description File," 2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE), 2018, pp. 120–124. doi: 10.1109/ICITISEE.2018.8720983.
6. Atul, M., et al. (2011). “FPGA Implementation of AES Algorithm”, International Conference on Electronics Computer Technology (ICECT), pp. 401–405.
7. Bassil, C., et al. (2005). “Critical voice network security analysis and new approach for securing Voice over IP Communications”, SETIT 2005, 3rd International Conference: Sciences of Electronic, Technologies of information and Telecommunications, Tunisia.
8. T. Chalermwat, et al. (2011). "FPGA Implementation of FOEPortable hard disk System”, The International Conference on Information and Communication Technology for Embedded Systems, Pattaya, Thailand.
9. Eric Cole, et al. (2005). Network Security Bible, Wiley Publishing Inc, 2005.
10. Computer Security Objects Register (CSOR), National Institute of Standards and Technology

(NIST). http://csrc.nist.gov/csor/

11. Daemen, J. and Rijmen, V. (2009). AES Proposal: Rijndael, AES Algorithm Submission. Available at http://www.nist.gov/CryptoToolkit

12. Daemen, J. and Rijmen, V. (2010). The block cipher Rijndael, Smart Card Research and Applications, LNCS 1820, Springer-Verlag, pp. 288–296.

13. Gladman, B. (2012). AES related home page. http://fp.gladman.plus.com/cryptography_technology/

14. Hoang, T. and Nguyen, V. (2012). An efficient FPGA implementation of the Advanced Encryption Standard Algorithm. IEEE 978-1-4673-0309-5/12.

15. Lee. (2009). NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, National Institute of Standards and Technology.

16. Patil, A. A., & Deshpande, S. (2025). Real-time encryption and secure communication for sensor data in autonomous systems. Journal of Information Systems Engineering and Management, 10(415), 41–55. https://www.jisem-journal.com.