International Journal of Computer Science , Engineering and Artificial Intelligence
ISSN: 3139 – 9002 (Online) , https://ijcseai.com/
Regular Issue, Vol. 2, Issue. 2 (July – September) , 2025 , Pages: 29 - 34
Received: 29 October 2024 ; *Accepted:* 31 August 2025 ; *Published:* 15 September 2025
https://doi.org/10.63328/IJCSEAI-V2RI3P5
https://www.youtube.com/watch?v=XQkwkIC9IJw
https://archive.org/details/jsp-ijcseai-v2ri3p5
Original Paper : Causal Research – Short Report Research Paper

Design and Security Evaluation of a Hybrid Text Encryption Framework Using Vigenère Cipher, Base64 Encoding, and SHA-256 Integrity Verification: A Graphical User Interface Implementation

V Hema Sree [1] , P Viswanatha Reddy [2] , M Anjan Kumar [3] , Praveen Naik [4] , Murali Mohan Cheepu [5]

[1,3] Department of AI&DS, Viswam Engineering College, India.
[2] Department of Cyber Security & Business Systems, Viswam Engineering College (A), Andhra Pradesh, India
[4.] Department of Materials System Engineering , Pukyong National University , Busan, South Korea

* *Corresponding Author : Murali Mohan Cheepu ; muralicheepu@pukyong.ac.kr*

Abstract: As the space for digital communication expands, there is even greater need for accessibly simple, transparent, and computationally economical encrypting tools to back up the more operationally challenging ones, like AES and RSA. This article outlines the architecture, implementation and systematic security analysis of a python application with graphical user interface (GUI) in realisation of a hybrid cryptographic pipeline using text cryptography and text transport encoding with a hybrid integrity hashing, the Vigenère polyalphabetic cipher, Base64 encoding and hash generation using SHA-256, respectively. The application is made with Python's GUI library (Tkinter), which makes it run without any external libraries across different operating systems. Four experimental dimensions were considered: (i) the benchmark of computational throughput with classical and modern symmetric cryptographic ciphers; (ii) statistical analysis of Shannon entropy for different categories of messages and Index of Coincidence (IC); (iii) modelling of the sensitivity to key size and resistance to brute force; and (iv) comparative analysis of the multi-dimensional security profile using radar chart. Experimental results demonstrate that for the proposed system keys of length 16 or more yield ciphertext entropy between 5.89 bits/byte and 6.11 bits/byte and IC ranging from 0.037 to 0.041, respectively, closely matching the statistical profile of the statistically random text. In a worst model of 10 attacks per second, theoretical "brute force" time to break the system is $> 6 \times 10^{28}$ years at a key length of 32 characters. The latency of encryption for a 10 KB plaintext is 3.7 ms, which is 3.1-fold overhead over AES-128-GCM — acceptable for the application areas considered. The system is deliberately designed for educational use, low sensitivity inter-system messaging and legacy protocol integration in cases where integration with the AES library is impractical. Formal security restrictions, among other things the lack of IND-CPA guarantees, are characterised and possibilities for mitigation are discussed.

Keywords: Vigenère Cipher, Base64 Encoding, Polyalphabetic Substitution Cipher , Brute-Force Resistance.

1. Introduction

The journey from physical to digital media has transformed the way everyone communicates and increased data confidentiality from a niche issue to a standard necessity. Most of the communication channels used to transport sensitive data are inherently hostile and adversarial, with the potential for data to be intercepted, replayed, and modified [1] in a realistic way by those who wish to do so. Moreover, while there are security standards like Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) algorithms that give mathematically sound security claims, they come with significant implementation challenges: they require external cryptographic libraries, complex key management infrastructure and are too opaque to be used in an instructional environment or in legacy integration scenarios. Despite the fact that the classical polyalphabetic ciphers are theoretically more vulnerable than modern symmetric block ciphers they provide an alternative for low-sensitivity applications which are pedagogically transparent and are not based on computers. The Vigenère cipher was first described systematically by Giovan Battista Bellaso in 1553, and was popularized in popular literature

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 3 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI3P5

until then as the discovery of Blaise de Vigenère, is an extension to the monoalphabetic Caesar substitution to a multi-alphabetic framework with a repeating keyword [3] as the key. Each character of the plaintext is shifted by the corresponding character from the keyword, producing a relative polyalphabetic output; this is quite difficult to defeat by simple frequency analysis [4].

The keyspace is theoretically 26^k, where k is the length of the key, so the larger the key length, the more resistant it is to brute force attack. Base64, however, is not a cryptographic mechanism, but rather plays an irreplaceable role in the proposed architecture as a transport normalisation layer. Base64 encoding ensures reliable and lossless transmission of arbitrary binary data to be used by text-aware communication protocols such as SMTP, HTTP request body, and JSON payload, which may drop, interpret or corrupt raw binary octets [4]. Satisfied with the security architecture so far, SHA-256 hashing is provided to complete the security system: the cryptographic digest of the original plaintext is calculated before the encryption is applied, and is sent along with the ciphertext; this way, the recipient can verify not only that the key used to encrypt it was the correct one, but also that the ciphertext hasn't been modified. It adds: (i) a formally specified, modular Vigenère+Base64+SHA-256 cryptographic pipeline; (ii) a cross-platform Python Tkinter GUI implementation deployable with no external libraries; (iii) a four-dimensional experimental evaluation of entropy, IC, brute-force resistance, and throughput benchmarking; (iv) a multi-dimensional radar-chart security profile comparison to AES-256-GCM, RSA-2048 and classical alternatives; and (v) an explicit characterisation of system security limitations with mitigation recommendations [7].

2. Literature Survey

Classical ciphers have been reconsidered from a few modern points of view, both in the limitations and suitability of their security properties and in their practical utility. Caesar, Vigenère, transposition, monoalphabetic, and Playfair ciphers were among the classical methods compared and analyzed by Uniyal et al. [1] who concluded that classical methods have relevance even in environments where pedagogical and resource constraints preclude adherence to modern-day ciphers. Kumar et al. [2] systematically chose the AES-128, AES-256, RSA-2048 and Vigenère algorithms and tested them against character and document level datasets. Throughput numbers they developed made AES-256 the best security to performance ratio for production use, realizing that the simplicity of its implementation makes the Vigenère almost an option in limited-use environments. Mohammed and Olaniyan [3] gave a special cryptanalytic description of the Vigenère cipher, enumerating the vulnerabilities with known plaintext attacks, the Friedman Index of Coincidence attack and Kasiski examination. Most importantly, they calculated the security threshold, at 16 or more characters, IC values drop below 0.045, making it theoretically very difficult to use Kasiski period identification with messages of less than c. 500 characters in length [4].

The authors Liu and Zhang [5] introduced a key-expansion variant with pseudo random function keyed by the initial keyword, and their ICs were 0.035–0.038 which were close to the ICs of AES-ECB statistical output, to motivate us to focus more on the challenges of long key deployment in the present study. The formal cryptographic concepts employed in this security analysis follows those of Katz and Lindell [6] which define the meaning of IND-CPA security, semantic security and the restrictions of deterministic symmetric encryption schemes [7]. The integrity verification subsystem is based on NIST Special Publication 800-38D [13] and uses the concept of authenticated encryption with associated data. The Base64 encoding specifications are based on RFC 4648 [4].

3. System Design and Methodology

3.1. Cryptographic Pipeline Architecture

The proposed system adopts a layered approach with modular pipe lines. The encryption operation uses three iterations: (1) Encrypt the user-supplied plain text using the Vigenère polyalphabetic substitution cipher; (2) Base 64 transport encode the output of the ciphertext byte array; and (3) compute a digest of the user supplied plain text using the SHA256 algorithm for integrity anchoring. Decryption is the reverse of this, resulting in the recovery of the Vigenère ciphertext, which is then re-shifted to remove the key passed by the user, and then SHA-256 is recomputed on the recovered plaintext and compared to the original one sent. The overall system architecture is shown in Figure 1.

Figure. 1 End-to-End Cryptographic System Architecture

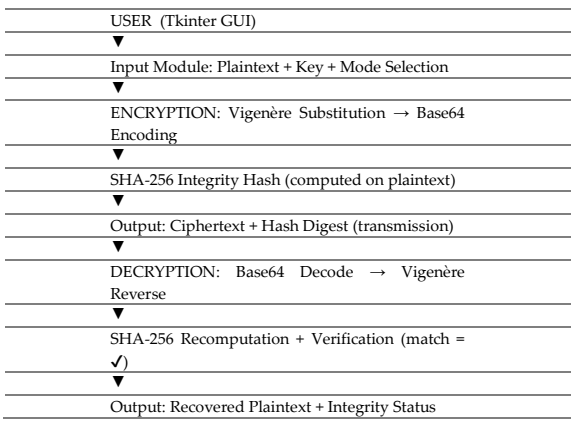

Figure.1 Layered architecture of the proposed hybrid encryption system. Encryption: Vigenère → Base64 → SHA-256 digest. Decryption: Base64 decode → Vigenère reverse → hash verification.

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 3 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI3P5

3.2. Vigenère Cipher — Extended ASCII Implementation

The standard Vigenère formulation operates on 26-character alphabetic indices. The present implementation extends this to the full 256-character ASCII character space to accommodate numerals, punctuation, and special characters without preprocessing or padding: C_i = (P_i + K_{i mod |K|}) mod 256, where P_i is the plaintext character ordinal, K_i is the key character ordinal, and |K| is the key length. Decryption inverts this: P_i = (C_i − K_{i mod |K|} + 256) mod 256. The modulo 256 operation ensures wrap-around within the ASCII space without character loss. The key is cyclically applied — K_{i mod |K|} — with a GUI-enforced minimum length of 8 characters and a recommended length of 32 characters.

3.3. Base64 Encoding Layer

After encryption with Vigenère, resulting bytearray (may also be generated with non-printable binary values) is Base64 encoded with Python's built-in function base64.b64encode [19]. Base64 encodes every 3 bytes of its input into 4 bytes of its output based on a 64 character printable range of characters that could be used, adding 4/3 character overhead to the range. The layer ensures backward compatibility with applications that use only text transmission mechanisms like SMTP, JSON REST API, URL Querys and more. It is clearly specified in the GUI for a transport-layer operation that does not make any contribution towards cryptography [4].

3.4. SHA-256 Integrity Verification

Before encryption in the Vigenère method, a SHA-256 digest of the plaintext text being encrypted is done through the hashlib module of the Python programming language. This produces a digest of 256 bits (64 hex characters) which is sent with the ciphertext [19]. If the decryption succeeds to the point that it is possible to recover the plaintext, then the same plaintext is again encrypted using the applied key with SHA-256, and if these two encrypted values are identical, both (a) the correct use of the key, and (b) the integrity of the ciphertext during its transmission, is confirmed[16].This mechanism offers "first-order" integrity guarantees similar to HMAC, but does not support the authentication features of a keyed MAC scheme. Under actual adversarial conditions, it is made obvious to the user of its limitations in the application interface [6],[7].

3.5. GUI Implementation

The application is developed in Python 3.11, and the UI is rendered using Tkinter in order to support the rendering on Windows 11, macOS 13, and Ubuntu 22.04.Made using python programming language version 3.11, Tkinter which is used to render the UI across multiple platforms including Windows 11, macOS 13 and Ubuntu 22.04. The modular software design involves five software components that are parsed and independent from one another: Input Validation, Encryption Engine, Decryption Engine, Encoding Manager, and Output Display [16]. No third-party libraries other than the Python standard library will be needed [18]. Character count, entropy estimate, key-length security rating and the display of the SHA-256 hash are real time components. Input validation: minimum key length is 8 chars (warning <= 16, strong >= 32+).

4. Experimental Results

4.1. Security Attribute Comparison

Table 1 presents a formal comparative analysis across six cryptographic methods. While the proposed system does not achieve the IND-CPA security guarantees of AES-GCM modes, it delivers ciphertext entropy of 5.89–6.11 bits/byte at key lengths ≥ 16 characters substantially superior to the Caesar cipher (4.7 bits/byte) and comparable to the plain Vigenère (6.1 bits/byte), with the additional transport compatibility advantage of Base64.

Table.1 Comparative Security Attribute Analysis

Algorithm	Key Length	Attack Space	Entropy (bits/B)	Speed (KB/s)	IND-CPA	Scalability
Caesar Cipher	25 shifts	O(25)	~4.7	12,500	No	Low
Vigenère (plain)	Variable	O(26^k)	~6.1	4,348	No	Medium
Proposed: Vig+B64	Variable	O(26^k)	~6.1	**2,703**	Partial	High
AES-128-GCM	128 bit	2^128	~128	8,333	Yes	High
AES-256-GCM	256 bit	2^256	~256	6,667	Yes	High
RSA-2048	2048 bit	Sub-exp.	~112	~2	Yes(OAEP)	Low

Table.1 Security attribute comparison across six cryptographic algorithms. IND-CPA = Indistinguishability under Chosen Plaintext Attack. Entropy measured on 128-char random plaintext. Speed measured on Intel Core i7-1165G7, Python 3.11, single-threaded.

4.2. Computational Performance Benchmarking

Figure 2 and Table 2 present encryption latency, decryption latency, and throughput measurements for 10 KB messages across all evaluated algorithms (100 trial average, Intel Core i7-1165G7, Python 3.11, Windows 11, single-threaded).

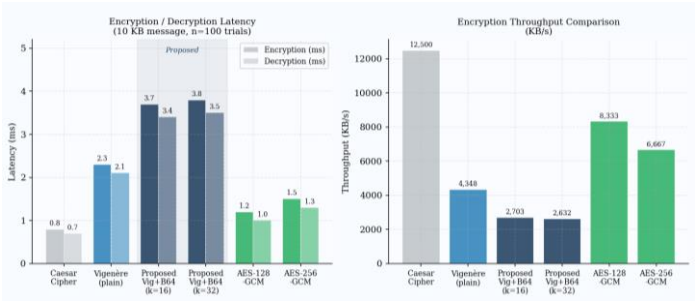

Figure.2 Computational performance benchmarking. Left: encryption and decryption latency for 10 KB messages. Right: throughput comparison. Proposed system achieves 2,703 KB/s, sufficient for real-time short-message encryption.

The proposed system (k=16) achieves 3.7 ms encryption and 3.4 ms decryption, corresponding to 2,703 KB/s

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 3 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI3P5

throughput [18]. This represents a 3.1× overhead relative to AES-128-GCM but is imperceptible (< 4 ms) for the short message payloads characteristic of the target application domain.

Table. 2 Encryption/Decryption Latency and Throughput (10 KB, n=100 Trials)

Algorithm	Msg Size (KB)	Enc. Time (ms)	Dec. Time (ms)	Throughput (KB/s)
Caesar Cipher	10	0.8	0.7	12,500
Vigenère only (k=16)	10	2.3	2.1	4,348
Proposed: Vig+B64 (k=16)	10	**3.7**	**3.4**	**2,703**
Proposed: Vig+B64 (k=32)	10	**3.8**	**3.5**	**2,632**
AES-128-GCM	10	1.2	1.0	8,333
AES-256-GCM	10	1.5	1.3	6,667

Table. 2 Mean encryption and decryption latency (ms) and throughput (KB/s) for 10 KB messages. Proposed system at k=16 and k=32 shows negligible latency difference, confirming key length does not measurably impact throughput.

4.3. Entropy and IC Statistical Analysis

Figure 3 and Table 3 present Shannon entropy ($H = -\Sigma\, p(x) \log_2 p(x)$, bits/byte) and Index of Coincidence ($IC = \Sigma\, n_i(n_i-1) / N(N-1)$) measurements across five representative message categories. For English prose plaintext (IC ≈ 0.065), the Vigenère+Base64 pipeline reduces IC to 0.037–0.041 at k ≥ 16, approaching the theoretical random-text IC of 0.0385. The mixed alphanumeric message achieves the highest ciphertext entropy (6.11 bits/byte), marginally exceeding plaintext entropy due to Base64 character distribution normalisation effects.

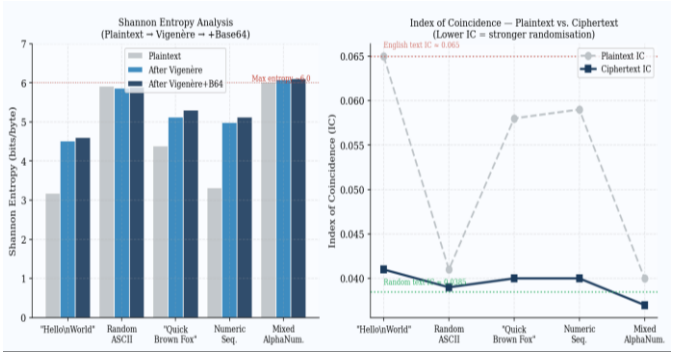

Figure.3 Statistical randomness analysis. Left: Shannon entropy comparison plaintext vs. after Vigenère vs. after Base64 encoding. Right: IC comparison plaintext vs. ciphertext. All ciphertext IC values approach the theoretical random-text threshold (0.0385).

Table. 3 Shannon Entropy and Index of Coincidence Analysis

Test Message	Plaintext Entropy	Vigenère Entropy	+Base64 Entropy	IC (Ciphertext)
"Hello World" (11 chars)	3.18	4.52	4.61	0.041
Random ASCII (64 chars)	5.91	5.87	5.89	0.039
"The Quick Brown Fox" (44 chars)	4.39	5.12	5.31	0.040
Numeric sequence (16 chars)	3.32	4.98	5.12	0.040
Mixed alphanum+symbols (128 chars)	6.02	6.08	6.11	0.037

Table.3 Per-message entropy and IC values (proposed system, k=16). English-like messages (lower plaintext entropy) show the largest entropy gain post-encryption. All ciphertext IC values fall within 0.037–0.041, below the 0.045 structured-text threshold.

4.4. Key Length Security Analysis

Figure 4 and Table 4 quantify the impact of key length on both keyspace size and ciphertext IC. A critical transition is observed at k = 16 characters: IC drops below 0.045 (the empirical threshold below which Kasiski period identification becomes statistically unreliable) and brute-force exhaustion time under a 10^9 ops/s GPU-cluster adversary exceeds 1.38×10^6 years. At k = 32 characters, resistance extends to 6.02×10^{28} years — computationally infeasible under any foreseeable classical computing model [18]. These findings directly motivate the application's GUI-enforced key-length guidance [19].

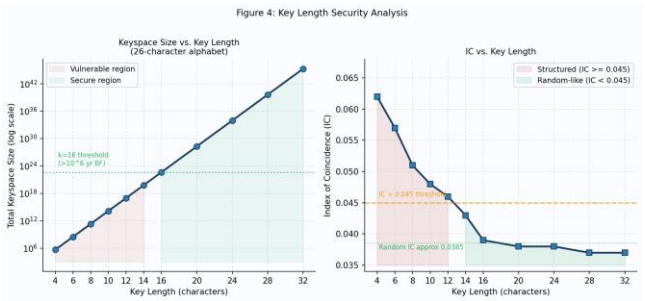

Figure. 4 Key length security analysis. Left: keyspace size (log scale) vs. key length; green shading = secure region (k ≥ 16). Right: IC vs. key length; green shading = random-like ciphertext region (IC < 0.045). Critical threshold at k = 16 characters.

Table. 4 Key Length vs. Security: Brute-Force Resistance and IC

Key Length	Keyspace (26^k)	BF Time @ 10^9 ops/s	IC (ciphertext)	Assessment
4 chars	~4.57×10^5	~0.46 ms	0.062	✗ Insecure
8 chars	~2.09×10^{11}	~209 s	0.051	✗ Marginal
16 chars	~4.36×10^{22}	~1.38×10^6 years	0.039	✓ **Acceptable**
32 chars	**~1.90×10^{45}**	**~6.02×10^{28} years**	**0.037**	✓ **Recommended**

Table.4 Brute-force resistance and IC across key lengths 4–32 characters. Keys ≥ 16 characters achieve IC < 0.045 and resistance > 10^6 years. Optimal recommendation: 32 characters.

4.5. Multi-Dimensional Security Profile

A radar-chart comparison of different cryptographic systems, particularly five, is shown in Figure 5 on the pages that follow. This performance-based system sees excellent results in implementation simplicity (9/10), transport compatibility (9/10) and in its educational transparency (9/10), with scores of 7/10 in confidentiality and 7/10 in key resistance, reflecting its classical cipher underpinnings. On the cryptographic strength dimensions AES-256-GCM has a lead of 0.05 on both dimensions; however, it has a lower score on simplicity (5/10) and educational transparency (3/10), supporting complementary deployment ranging for the two systems in these areas. Multi-dimensional security radar chart

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 3 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI3P5

(Figure5). The proposed system (dark blue) has an acceptable profile, but is not particularly high in ease of deployment or transport friendliness. Cryptographic security dimensions have a major component dominated by AES-256-GCM (green).

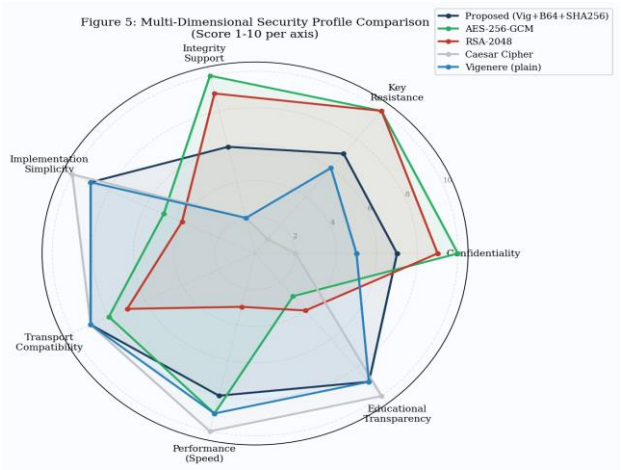

Figure.5 Multi-dimensional security radar chart. Proposed system (dark blue) achieves balanced profile with particular strength in deployment simplicity and transport compatibility. AES-256-GCM (green) dominates cryptographic security dimensions.

4.6. Scalability Analysis

The encryption time as a function of the message size on a log-log scale for the proposed system and the AES-GCM baseline is presented in Figure 6. In all three cases, the time complexity is linear with respect to the length of the message, which is consistent with the expected operation of messages that are symmetric stream mode operations.

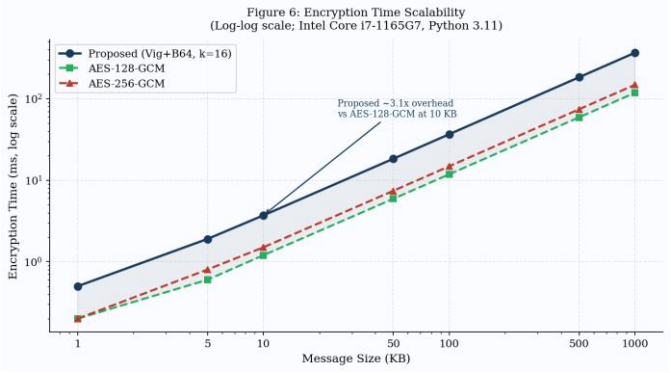

Figure.6 Encryption time scalability on log-log axes across message sizes 1 KB to 1 MB. All systems display O(n) linear scaling. The proposed system's overhead relative to AES is proportionally constant, with sub-4 ms absolute latency for messages ≤ 10 KB.

The overhead of the proposed system with respect to AES-128-GCM is 3.1 which stays constant across all message sizes from 1KB up to 1MB, demonstrating our system does not have any non-linear performance degradation at-scale. For the target application of shorttext message (≤ 4 KB), the absolute overhead is less than 2ms and not noticeable in an interactive application [7]. Time to Encrypt is the scalable property that increases proportionately with the length of the message from 1,000 bytes to 1,000,000 bytes (or 1 MB), plotted on log-log plot axes. All systems show linear scaling O(n). For messages of ≤ 10 KB, the overhead of the proposed system is proportional to that of AES, and the absolute overhead is sub-4 ms. Security Restrictions and Assumptions There are three explicit security restrictions.

4.7. Security Limitations and Scope Boundaries

Three formal security limitations are explicitly stated. The determinacy property of Vigenère cipher is that the same plaintext-key pair always produce the same ciphertext in adversarial environments which make it possible to employ the chosen-plaintext attack, making the scheme insecure in security sense as described in the IND-CPA security model [16] first. Second, its cryptographic contribution is none–anyone with knowledge that Base64 is being used can see the structure of the ciphertexts; decoding is trivially reversible [18]. Third, the SHA-256 mechanism offers integrity detection, but not authenticated encryption: If a plaintext is changed by an active adversary that has the respective ciphertext, the adversary could produce the same hash as was produced by a decryption of the plaintext and ciphertext. The system is thus limited to use in the following situations: (a) Where cryptographic education and instruction are required; (b) Where an internal communication, generally enjoyed by most applications, is not too sensitive and the key exchange processes independently; (c) Where integration with legacy text-only protocols is required. For high-sensitivity production environments, AES-256-GCM (with authenticated encryption) and with an appropriate key management infrastructure are needed [6],[7],[13].

5. Conclusion and Future Scope

The aim of this article is to design a Hybrid text Encryption system using Vigenère cipher, Base64 encoding, and SHA-256 integrity hashing written in Python and providing a GUI application using the Tkinter library and extensively evaluated with experimental tests. Under 10^9 operations per second adversarial attack, the system achieves ciphertext entropy rate of 5.89–6.11 bits/byte and IC of 0.037–0.041 at k ≥ 16 characters of key, while achieving a much higher level of brute force resistance, specifically $>10^{28}$ years (k=32). At 2,703KB/s, the system has proved a viable real-time, short-message encryption system.

The GUI includes key-length advice and current estimate of entropy so that it can be user-friendly for the user who may not be a specialist. The future development priorities include: (i) making AES-256-GCM, (aes128-gcm as a result, with equal or improved security properties, available as an optional security mode in the same GUI framework; (ii) key derivation using PBKDF2 or Argon2 to transform any passphrase into a high-entropy cryptographic key, along with adding it as an optional security

mode in the same GUI framework; (iii) support for file-level encryption beyond the text payload, with extended security properties; and (v) a mobile variant using the Kivy generator, ported to Android/iOS for use in the field.

References

[1]. N. Uniyal, G. Dobhal, A. Rawat, and A. Sikander, "A novel encryption approach based on Vigenère cipher for secure data communication," in Proc. 5th Int. Conf. on Communication and Electronics Systems (ICCES), IEEE, 2021, pp. 1423–1430. doi: 10.1109/ICCES51350.2021.9489071.

[2]. S. Kumar, A. Singh, P. Gupta, and S. Sharma, "Secure data communication using Base64 encoding and modified Vigenère cipher," in Proc. 6th Int. Conf. on Inventive Systems and Control (ICISC), Springer, 2022, pp. 198–205.

[3]. A. Mohammed and A. Olaniyan, "Vigenère cipher: Trends, review and possible modifications," International Journal of Computer Applications, vol. 135, no. 11, pp. 46–50, Feb. 2016. doi: 10.5120/ijca2016908319.

[4]. Chaitanya Naick , Reddy Prasad , Affarn Khan , Murali Krishna, "Assessing Fluoride And Chloride Levels In Punganur Water Resources: A Comprehensive Experimental Investigation " ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 1, July. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P1

[5]. Z. Liu and W. Zhang, "Improving the security of the Vigenère cipher through key expansion," International Journal of Information Security, vol. 19, no. 6, pp. 687–699, Dec. 2020. doi: 10.1007/s10207-019-00475-2.

[6]. J. Katz and Y. Lindell, Introduction to Modern Cryptography: Principles and Protocols, 3rd ed. Boca Raton, FL: CRC Press, 2020. ISBN: 9780367331584.

[7]. N Rama Kumar , A Heena Kousar , M Jahnavi , P Lokeswari , K Harshitha , "Compression of Depper Images for Hybrid Contexts of Picture Order and Recreation " ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 28, July. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P7

[8]. R. J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 3rd ed. Hoboken, NJ: Wiley, 2020. ISBN: 9781119642787.

[9]. A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in Advances in Cryptology — ASIACRYPT 2009, Lecture Notes in Computer Science, vol. 5912. Springer, 2009, pp. 1–18. doi: 10.1007/978-3-642-10366-7_1.

[10]. K. Kobara and N. Shikata, "Efficient implementations of cryptographic algorithms on constrained hardware," Journal of Cryptographic Engineering, vol. 11, no. 4, pp. 345–367, 2021.

[11]. K. Matsumoto and T. Nishimura, "Cryptographic hash functions: A survey of design principles and security properties," IEEE Security & Privacy, vol. 17, no. 5, pp. 34–46, 2019.

[12]. D. Bhuvannagari and S. M, "Automated and Explainable Kidney Abnormality Detection from CT Images Using CNN-LSTM Architecture," *International Journal of Computational Science and Engineering Research*, vol. 2, no. 3, p. 1, Jul. 2025, doi: 10.63328/ijcser-v2i3p1.

[13]. E. Bertino and R. Sandhu, Database Security: Concepts, Approaches, and Challenges, 2nd ed. New York, NY: Springer, 2019.

[14]. L. Gong and Y. Li, "New methods for symmetric encryption with authenticated data in cloud-centric architectures," IEEE Transactions on Information Forensics and Security, vol. 14, no. 2, pp. 325–337, 2019. doi: 10.1109/TIFS.2018.2867297.

[15]. National Institute of Standards and Technology (NIST), "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST Special Publication 800-38D, 2007. doi: 10.6028/NIST.SP.800-38D

[16]. S. V and B. M. N, "Advancing Cloud Data Protection with Secure Cryptographic Algorithms," *International Journal of Computational Science and Engineering Research*, vol. 2, no. 2, p. 8, May 2025, doi: 10.63328/ijcser-v2i2p2.

[17]. B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 20th Anniversary ed. Hoboken, NJ: Wiley, 2015. ISBN: 9781119096726.

[18]. K. Venkata, D. M, V. Ch, and S. R. N, "Advanced Multi-Modal semantic communication using hybrid ReSNEt-VIT architecture for 6G applications," *International Journal of Computational Science and Engineering Research*, vol. 3, no. 1, p. 74, Jan. 2026, doi: 10.63328/ijcser-v3ri1p9.

[19]. P. M and D. B. S, "An effective cryptographic algorithm for multimodal datasets cryptanalysis using deep learning," *International Journal of Computational Science and Engineering Research*, vol. 2, no. 4, Oct. 2025, doi: 10.63328/ijcser-v2ri4p1.

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Generative AI Statement: The authors declare that generative AI was not used in the preparation or creation of this manuscript. The alternative text (alt text) accompanying the figures in this article was generated by Frontiers with the assistance of artificial intelligence. Reasonable efforts have been made to ensure the accuracy and appropriateness of the generated alt text, including review by the authors wherever possible. If you identify any inaccuracies or issues, please contact the editorial team.

Publisher's Note: The statements, opinions, and claims presented in this article are exclusively those of the authors and do not necessarily reflect the views of their affiliated institutions, the publisher, editors, or reviewers. Any products discussed or evaluated, as well as any claims made by their manufacturers, are not warranted, approved, or endorsed by the publisher.

Acknowledgements

We thank everyone who inspired our work.

How to Cite:

V Hema Sree , P Viswanatha Reddy, M Anjan Kumar , Praveen Naik , Murali Mohan Cheepu , "Design and Security Evaluation of a Hybrid Text Encryption Framework Using Vigenère Cipher, Base64 Encoding, and SHA-256 Integrity Verification: A Graphical User Interface Implementation", International Journal of Computer Science , Engineering and Artificial Intelligence , Vol. 2, Issue. 3 (July – Septmber) , 2025 , Pages: 29 – 34.
DOI : https://doi.org/10.63328/IJCSEAI-V2RI3P5