Gandhara Journal of Natural Sciences (GJNS) | **Performance Analysis of Cryptographic Techniques on Pashto-Language Text** | Volume No. 1 | Issue No. 2 | p. 49 - 59 | e-ISSN: 3106-1575 | https://gjns.kdru.edu.af

Submitted: March 17, 2026 | **Revised:** April 04, 2026 | **Accepted:** June 12, 2026 | **Published:** June 16, 2026

Performance Analysis of Cryptographic Techniques on Pashto-Language Text

Mohammad Nawid Sadat[1], Mohammad Khan Haidary[2], Abdul Wakil Baidar[2], Noorullah Noori[2]

A B S T R A C T

Purpose:

This study aimed to conduct a comprehensive comparative analysis of symmetric and asymmetric cryptographic algorithms and evaluate their performance when applied to texts in the Pashto language. The study focused on assessing encryption and decryption efficiency, security strength, and resistance to brute-force attacks while exploring the applicability of cryptographic techniques to a non-Latin-based language.

Method:

A set of widely used symmetric cryptographic algorithms (DES, 3DES, AES, and Blowfish) and asymmetric cryptographic algorithms (RSA, ElGamal, and ECC) were implemented and tested using Python in a controlled computing environment. The evaluation considered factors such as encryption speed, decryption speed, and security performance. To ensure accuracy and reproducibility, the experiments were conducted under controlled conditions, including standardized hardware specifications, operating systems, software versions, and input data formats.

Results:

The findings revealed that AES achieved the highest performance among the symmetric algorithms in terms of encryption and decryption speed, making it the most efficient option for processing Pashto text. Among the asymmetric algorithms, ECC demonstrated the strongest security characteristics and superior resistance to brute-force attacks while maintaining efficient performance. The results further confirmed that conventional cryptographic techniques can be successfully applied to Pashto-language texts when appropriate character encoding and linguistic considerations are properly configured.

Practical Implications:

The findings highlight the suitability of AES for applications requiring fast and efficient encryption and ECC for environments demanding strong security protection. The study provides valuable guidance for developers, organizations, and policymakers seeking to implement secure communication systems and data protection mechanisms for Pashto-language digital applications and information systems.

Originality/Novelty:

This study represents one of the first comprehensive investigations into the performance of symmetric and asymmetric cryptographic algorithms using Pashto-language text. It contributes to the fields of cryptography, cybersecurity, and language-oriented computing by demonstrating the adaptability of established encryption methods to a low-resource, non-Latin language context and supporting the advancement of local cybersecurity initiatives.

Keywords: Asymmetric; Cryptography; Symmetric; Pashto; Performance.

Authors Affiliations:

[1]Department of Mathematics, Faculty of Education, Helmand University, Helmand, Afghanistan.

[2] Department of Mathematics, Faculty of Education, Kabul University, Kabul, Afghanistan

***Corresponding e-mail:** Mnsadat1992@helu.edu.af **S**

ORCID:

1. Introduction

In the modern digital age, where information is exchanged at unprecedented volumes and speeds, the need for robust data protection mechanisms has become more critical than ever. From individual users to multinational organizations and state institutions, the confidentiality, integrity, and authenticity of digital communication lie at the heart of trust and functionality in cyberspace. Cryptography, as a fundamental pillar of cybersecurity, provides systematic techniques to secure data from unauthorized access, tampering, and malicious manipulation.

Over the past several decades, both symmetric and asymmetric cryptographic algorithms have evolved to address the rapidly changing landscape of cyber threats. Symmetric algorithms, such as the Data Encryption Standard (DES) and its enhanced successor Triple DES (3DES), initially offered reliable methods of encryption but were later rendered inadequate due to advances in computational power and the emergence of brute-force attack techniques (Alabdulrazzaq & Alenezi, 2022; Marwaha et al., 2018; Ratnadewi et al., 2018). In response, the Advanced Encryption Standard (AES) was developed, balancing speed, flexibility, and high-level security, leading it to become the globally accepted standard for symmetric encryption (Al-Shabi, 2019; Kaushik et al., 2023; Maqsood et al., 2017; Marwaha et al., 2018; Najem et al., 2025; Stallings, 2017; Zhang, 2021). Similarly, Blowfish, introduced by Bruce Schneier, is a symmetric cipher known for its customizable key size (32–448 bits) and efficiency in resource-constrained environments, though its adoption has been limited due to the lack of formal standardization (Ali et al., 2020; Maqsood et al., 2017; Marwaha et al., 2018; Suguna et al., 2016; Zhang, 2021) .

In the realm of asymmetric cryptography, RSA has long been the cornerstone of secure key exchange and digital signatures. Despite its robustness, RSA suffers from performance limitations, particularly in systems where memory and processing power are constrained (Asagba & Nwachukwu, 2014; Mahto & Yadav, 2017). In recent years, alternatives such as Elliptic Curve Cryptography (ECC) and ElGamal have gained attention. ECC offers comparable levels of security to RSA but with significantly smaller key sizes, making it highly suitable for mobile and IoT environments (Amirkhanova et al., 2024; Xie, 2023). ElGamal, although secure and versatile, tends to be computationally intensive and is typically employed in more specialized domains (Astuti et al., 2023; ElGamal, 1985; Maqsood et al., 2017; Mishra & Kar, 2017; Peranginangin, 2024; Suguna et al., 2016; Teranishi & Kogiso, 2021).

Several comparative studies have examined the performance, complexity, and computational efficiency of cryptographic algorithms. For example, Marqas et al. (2020) conducted a performance-based comparison of AES and RSA with respect to encryption speed and key efficiency, while Alenezi et al. (2020) evaluated symmetric encryption algorithms using consistent Python-based test environments. However, a noticeable limitation across much of this literature is that the testing environments and datasets are almost exclusively based on English or standardized artificial text inputs. This approach overlooks the linguistic, structural, and encoding complexities inherent in non-Latin-script languages.

This gap becomes especially problematic in the context of multilingual computing systems, where local languages like Pashto spoken by millions across Afghanistan and neighboring regions are underrepresented in encryption performance analyses. Pashto utilizes a modified Arabic script with complex Unicode character sets, bidirectional text flow, and contextual letter shaping, all of which may affect algorithm behavior during practical implementation. While previous studies have addressed algorithmic strengths and weaknesses in general terms, very few if any have evaluated how these algorithms perform when applied to real Pashto-language text data, or what cryptographic trade-offs exist in such localized environments.

To address this crucial gap, the current study presents a comparative performance analysis of widely used symmetric (DES, 3DES, AES, Blowfish) and asymmetric (RSA, ECC, ElGamal) encryption algorithms, using real-world Pashto text as the encryption target. The algorithms are implemented using standardized Python cryptographic libraries within a controlled computational environment. The evaluation focuses on practical performance metrics, including encryption and decryption time, with a uniform input structure designed to simulate realistic usage scenarios.

Unlike other research that delves into theoretical constructs, this study is grounded in measuring real-time execution efficiency. Furthermore, it uniquely positions itself by assessing algorithmic performance on a script and language that presents structural challenges uncommon in Western languages. This adds both novelty and contextual relevance to the field of applied cryptography.

Ultimately, the findings of this study aim to assist cybersecurity engineers, system designers, and researchers in choosing cryptographic algorithms that are not only secure but also optimized for performance in multilingual computing environments. By shedding light on how these algorithms behave

when applied to Pashto-language data, the study contributes to filling a notable gap in the literature and opens the door to further exploration of language-aware encryption systems.

2. Methods and Materials

It presents the methodological approach adopted in conducting a comparative study of symmetric and asymmetric cryptographic techniques. The core objective of the study is to evaluate and compare the selected cryptographic algorithms based on performance efficiency (in terms of encryption and decryption time). To achieve this, a series of controlled experiments were carried out on a selected text sample using Python under a consistent computational environment.

2.1 Study Design

This study follows a quantitative experimental design. Each cryptographic algorithm is implemented, executed, and evaluated under the same conditions, using the same input data and key lengths (where applicable). The comparative analysis involves:

- Performance testing using practical encryption/decryption of a text sample;
- Tool-based implementation and automation using Python for accurate timing and computation.

2.2 Experimental Environment

The experiments were conducted on a standard, single-user computing environment with the following specifications:

- Device Name: LAPTOP-MNSADAT
- Processor: Intel(R) Core (TM) i7-8650U CPU @ 1.90GHz, Quad-Core
- Installed RAM: 8.00 GB (7.88 GB usable)
- Hard Disk: 512GB SSD
- Operating System: Windows 11 Pro, Version 23H2
- System Type: 64-bit Operating System, x64-based processor
- Installed on: February 17, 2025

This setup provides a consistent computational baseline for evaluating algorithmic performance.

2.3 Programming Tools and Frameworks

Table 1 indicates the tools and programming frameworks used for the implementation and testing of algorithms.

Table 1: Tools used in testing of algorithm

Tool/Library	Purpose
Google Collaboratory	simulation and real-time timing analysis
Python 3	Algorithm development and testing
PyCryptodome	Implementation of symmetric and some asymmetric primitives
Cryptography Library	Support for additional public-key schemes

2.4 Cryptographic Techniques Studied

2.4.1. Symmetric Algorithms

Table 2 shows the specifications used in Algorithm.

Table 2: Specification of symmetric Algorithm

Algorithm	Description	Key Size	Block Size
DES	Traditional block cipher with Feistel structure	56 bits	64 bits
3DES	Triple application of DES	112/168 bits	64 bits
AES	Advanced Encryption Standard with key flexibility	128/192/256 bits	128 bits
Blowfish	Fast cipher with variable-length keys	32–448 bits	64 bits

2.4.2. Asymmetric Algorithms

Table 3 illustrates the specifications of asymmetric Algorithm.

Table 3: Specification of asymmetric Algorithm

Algorithm	Description	Key Size	Notes
RSA	Widely used public-key encryption	1024/2048/4096 bits	Secure but computationally intensive
ECC	Elliptic Curve Cryptography	160–521 bits	High security with small key sizes
ElGamal	Public-key system based on Diffie-Hellman	1024–4096 bits	Supports encryption and signatures

2.5 Input Data and Test Procedure

A standardized text sample was used for encryption and decryption in all techniques. The same file was used across all tests to ensure consistency:

- Text File Size: 1.42KB

2.6 Testing Procedure

1. The original text file is encrypted using the selected technique.
2. The cipher text is then decrypted to verify correctness.
3. Execution time for both operations is recorded using built-in timers (time.perf_counter ()) in Python.
4. The encryption and decryption steps are repeated 100 times to calculate average values and reduce variance.
5. Brute-force attack feasibility is simulated by calculating total key combinations and estimating time based on assumed cracking speed (1 trillion keys/second).
6. Charts and tables are generated to visualize comparisons across all metrics.

2.7 Input Data and Test Procedure

The following metrics were used to evaluate and compare the cryptographic algorithms indicated in Table 4.

Table 4: Metrics were used in Algorithm.

Metric	Description
Encryption Time	Average time to convert plaintext to ciphertext (ms)
Decryption Time	Average time to retrieve plaintext from ciphertext (ms)
Key Size	Length of cryptographic key in bits

3. Results

This presents the outcomes of implementing and evaluating various cryptographic methods, accompanied by an in-depth analytical assessment. Different algorithms were developed and tested using the Python programming environment, all within a standardized and controlled computing setup. The performance of each algorithm was measured against key criteria, such as encryption and decryption time using keys of different lengths. All the collected data were organized into structured tables and visualized using graphs to clearly demonstrate the effectiveness of each method. Furthermore, the results were critically analyzed to identify the strengths, weaknesses, and practical usability of each algorithm in real-world cryptographic contexts. The gathered results were systematically presented in tables, and visual representations through graphs were employed to enhance clarity. Each table summarizes the core performance metrics of the evaluated algorithms, particularly the encryption and decryption times expressed in milliseconds. Comparative bar charts were used to highlight performance differences across various algorithms. These visual tools not only showcase the computational efficiency of each cryptographic method but also help determine which algorithm is most suitable based on practical constraints such as limited resources and specific performance requirements.

3.1 Symmetric Algorithms Performance Evaluation

Now, the encryption and decryption times of the symmetric algorithms DES, TDES, AES, and Blowfish will be measured. The goal is to compare the performance of these algorithms and select the most efficient one.

3.1.1 Encryption Time

To evaluate the performance of various symmetric encryption algorithms, a 1.42KB Pashto text was encrypted using DES, 3DES, AES, and Blowfish with different key sizes. The encryption time was recorded for each algorithm using the implementation tools specified in the methodology section. The results are visualized in the accompanying Figure1.

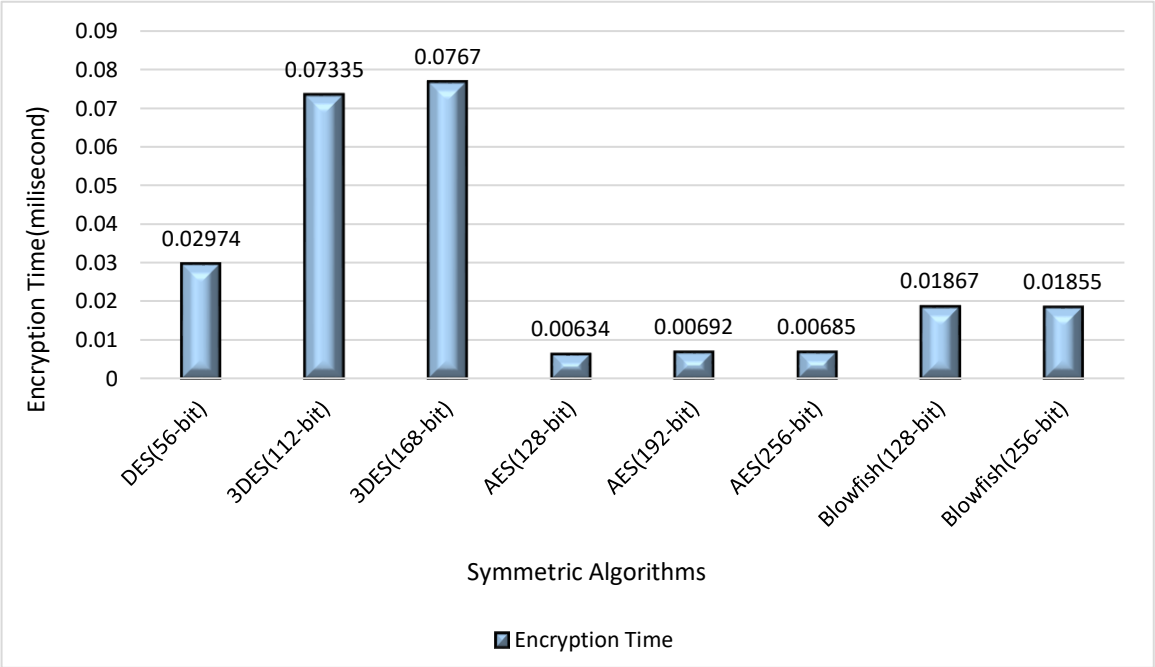

Figure 1. Symmetric Algorithms Encryption Time for 1.42KB Pashto Text.

Figure1 illustrates the encryption times of the DES, 3DES, AES, and Blowfish algorithms, measured based on different key lengths. These results were obtained by executing the Python codes provided in Appendix A. The results clearly show that AES performs significantly better compared to the other algorithms. Even when the key length increases from 128 bits to 256 bits, AES maintains a consistently low encryption time approximately 0.006 milliseconds demonstrating its high efficiency and ability to handle large keys without delay.

In contrast, DES, which uses a relatively short 56-bit key, requires about 0.03 milliseconds to encrypt data, indicating its inferior performance compared to AES. The 3DES algorithm, which implements 112-bit or 168-bit keys, shows a noticeable increase in encryption time, even higher than DES, suggesting that 3DES operates more slowly.

Blowfish, on the other hand, presents a middle ground. It encrypts data faster than DES and 3DES but remains slower than AES. Notably, Blowfish maintains nearly the same encryption time for key sizes ranging from 128 to 256 bits, indicating its effective management of larger key sizes without increasing encryption time. Overall, these findings highlight the superiority of AES, making it the optimal choice when fast encryption is required.

3.1.2 Decryption Time

In this section, the performance of selected symmetric encryption algorithms is evaluated based on their decryption time. The algorithms under consideration include DES, 3DES, AES, and Blowfish, each tested with varying key sizes. A 1.42KB Pashto text file was used as the input data for the experiment. The decryption time, measured in milliseconds (ms), reflects the computational efficiency of each algorithm when retrieving the original plaintext from the encrypted data. The results are summarized in Figure 2, providing a comparative view of how key size and algorithm design affect decryption speed.

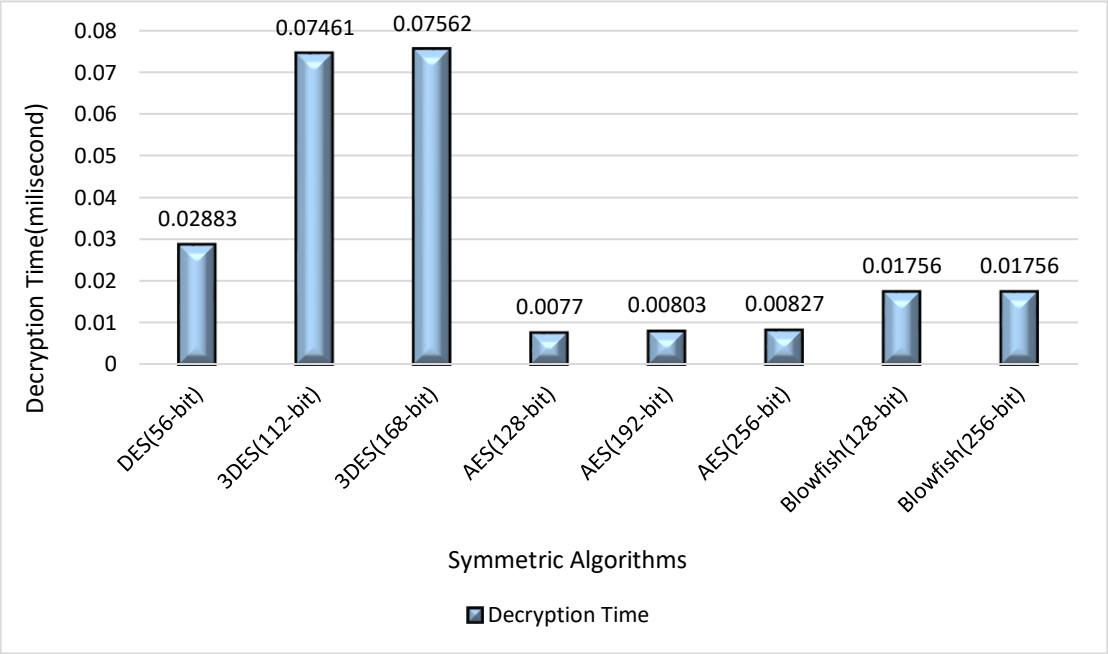

Figure2. Symmetric Algorithms Decryption Time for 1.42KB Pashto Text.

Figure 2 presents the decryption times of four symmetric encryption algorithms (DES, 3DES, AES, and Blowfish) based on decrypting a 1.42 KB Pashto text using various key sizes. These results were obtained by executing the Python codes provided in Appendix A. The results indicate significant performance differences among the algorithms. AES consistently recorded the fastest decryption times across all key sizes, with the shortest time being approximately 0.0077 milliseconds using a 128-bit key.

Blowfish also performed efficiently, registering a decryption time of around 0.01756 milliseconds for both 128-bit and 256-bit keys, suggesting that key length has minimal impact on its performance. In contrast, DES and TDES demonstrated noticeably weaker performance. DES required about 0.0288 milliseconds, and TDES showed a decryption time of 0.0176 milliseconds for both 112-bit and 168-bit keys, which were the highest among all algorithms tested. These findings indicate that AES is the most efficient algorithm in terms of speed, followed by Blowfish, making both suitable for time-sensitive or resource-constrained applications.

3.2 Asymmetric Algorithms Performance

This section evaluates the computational performance of asymmetric encryption algorithms. Algorithms such as RSA, ElGamal, and ECC are tested with different key sizes, and their encryption and decryption times are measured in order to compare their speed and efficiency.

3.2 1 Encryption Time

This section analyzes the encryption performance of three widely used asymmetric encryption algorithms: RSA, ElGamal, and Elliptic Curve Cryptography (ECC). Each algorithm was tested using multiple key sizes to assess how encryption time varies with key length. The benchmarking was conducted on a Pashto text sample of 1.42 KB to simulate the encryption of small, real-world messages in local language contexts. The objective was to measure the efficiency of each algorithm in terms of encryption time and determine which method is most suitable for low-latency, high-security applications shown in Table 5.

Table 5. Asymmetric Algorithms Encryption Time for 1.42KB Pashto Text.

Algorithm	Key Size	Encryption Time (ms)
RSA	1024-bit	0.407
RSA	2048-bit	0.753
RSA	4096-bit	1.356
ElGamal	1024-bit	10.594
ElGamal	2048-bit	72.545
ElGamal	3072-bit	216.921
ECC	256-bit	0.055
ECC	384-bit	0.050
ECC	521-bit	0.049

Table 5 Illustrates the encryption times of asymmetric algorithms. These results were obtained by executing the Python codes provided in Appendix B. Results clearly shows that the performance of these algorithms varies significantly based on the key size and structure. RSA was tested with 1024, 2048, and 4096-bit keys, recording encryption times of 0.407 ms, 0.753 ms, and 1.356 ms, respectively. This indicates a direct correlation between key length and encryption time the 4096-bit key takes approximately three times longer than the 1024-bit key. Nevertheless, RSA still demonstrates better speed performance compared to other algorithms.

ElGamal, tested with 1024, 2048, and 3072-bit keys, showed significantly higher encryption times: 10.594 ms, 72.545 ms, and 216.921 ms, respectively. This suggests that as the key size increases, ElGamal's performance degrades rapidly. The graph reveals that ElGamal is the slowest among RSA and ECC and exhibits poor efficiency with larger keys.

For Elliptic Curve Cryptography (ECC), three standard curves P-256, P-384, and P-52 were evaluated. Their corresponding encryption times were 0.055 ms, 0.050 ms, and 0.049 ms, respectively. These results indicate that, unlike other algorithms, ECC may even perform slightly better as key size increases. Overall, the data confirms that ECC is the fastest asymmetric encryption algorithm among those tested.

3.2.2 Decryption Time

In addition to encryption time, the efficiency of asymmetric cryptographic algorithms can also be evaluated based on their decryption performance. Decryption is a critical phase in secure communication systems, especially in real-time applications where delays can impact performance and user experience. The following table presents the decryption time (in milliseconds) for RSA, ElGamal, and ECC algorithms across different key sizes using a 1.42 KB Pashto text sample. The goal of this analysis is to compare how these algorithms perform during the decryption phase and to assess their suitability for practical use in terms of computational cost.

As shown in the Table 6 These results were obtained by executing the Python codes provided in Appendix B, ECC once again demonstrates superior performance in terms of decryption speed, with decryption times consistently ranging between 0.07 ms and 0.08 ms across all three key sizes. In contrast, RSA exhibits increasing decryption times as the key size grows, reaching up to 9.867 milliseconds for the 4096-bit key. Similarly, ElGamal shows a significant rise in decryption time with larger key sizes, peaking at 109.298 milliseconds for the 3072-bit key.

These findings reaffirm the efficiency of ECC, particularly in scenarios where fast decryption is essential. The results also highlight the high computational cost associated with RSA and ElGamal, which, although offering strong security levels, may pose challenges for systems with limited processing capabilities.

Table 6. Asymmetric Algorithms Decryption Time for 1.42KB Pashto Text.

Algorithm	Key Size	Decryption Time (ms)
RSA	1024-bit	0.634
RSA	2048-bit	2.064
RSA	4096-bit	9.867
ElGamal	1024-bit	5.773
ElGamal	2048-bit	36.891
ElGamal	3072-bit	109.298
ECC	256-bit	0.079
ECC	384-bit	0.068
ECC	521-bit	0.066

5. Discussion

The analytical findings of this study reveal significant performance variations among cryptographic algorithms in terms of execution time, security strength, and resistance to brute-force attacks. These variations are primarily influenced by key size, internal structure, and the underlying encryption mechanism of each algorithm. A 1.42KB Pashto-language text sample was employed during evaluation to examine how these algorithms respond to Unicode-based character sets and numeric representations inherent in the Pashto script.

Among symmetric algorithms, AES demonstrated the fastest performance across all its three key sizes. This aligns with the results of previous studies(Alam Hossain et al., 2016; Thirupalu & Reddy, 2019), which emphasize the algorithm's efficiency and speed. In contrast, 3DES recorded the longest execution time due to its triple-phase encryption design. Within the asymmetric category, ECC emerged as the most efficient in terms of speed, a finding consistent with Mahto and Yadav (2017). On the other hand, ElGamal exhibited the slowest performance across its key variations, while RSA also required significant execution time, rendering both less suitable for encrypting large text files.

It is important to nte that encryption algorithms operate not on human language semantics but on the encoded binary representations of data. That is, they do not directly interact with the meaning or linguistic structure of a language but rather with the byte-level inputs derived from encoding systems. For instance, a 1KB file containing Pashto, Dari, Arabic, or English text once encoded via UTF-8 will be processed based on its byte size, not linguistic content. Thus, for equal file sizes, the execution time of encryption will be comparable across languages.

However, language does indirectly influence encryption performance through its impact on file size. For example, English texts often use ASCII encoding where each character occupies 1 byte, whereas Pashto and similar scripts rely on Unicode, typically using 2–3 bytes per character. Therefore, translating an English text into Pashto naturally increases the file size, which in turn affects encryption and decryption time. While language itself does not play a direct role in encryption logic, its encoding impact specifically in terms of data volume and structure can substantially influence performance outcomes.

Overall, this discussion illustrates that the choice of cryptographic algorithm should be guided by both security and performance considerations. For securing electronic content in Pashto, AES and ECC represent the most balanced and efficient choices, offering strong encryption with relatively fast execution.

6. Conclusion

This study, through a detailed analytical evaluation of key cryptographic algorithms, reveals that the effectiveness of encryption is not solely determined by the algorithm's internal structure but is also closely associated with factors such as key size, the volume of encoded data, and the encoding mechanism of the language used. Based on empirical results, AES and ECC were identified as the most suitable algorithms for practical deployment, offering optimal trade-offs between high security and fast execution time. In contrast, 3DES and ElGamal demonstrated significantly higher computational overheads, making them more applicable in limited-use scenarios.

It is noteworthy that this research specifically focused on Pashto-language content. Moreover, the findings underscore the fact that encryption algorithms do not interact directly with natural language semantics; rather, they operate on binary-encoded data. However, the encoding mechanism of a given language can influence the volume of information being processed, thereby indirectly affecting execution time.

In conclusion, the selection of an encryption algorithm should consider the size of the text, the required level of security, and the computational capacity of the system in use. For Pashto which often involves high-byte Unicode characters algorithms with fast, block-based processing and strong encryption capabilities, such as AES, or those with lower computational requirements like ECC, are most preferable. This study serves as a foundational contribution to the field of cryptography for the Pashto language and can act as a reference point for future research in language-specific encryption performance.

Recommendations

- Evaluation of Post-Quantum Secure Cryptographic Algorithms: For Pashto-language digital content, it is essential to assess the security of encryption algorithms resilient to quantum computing attacks, such as lattice-based and code-based structures. These algorithms play a critical role in ensuring security against potential future threats posed by quantum computers. Therefore, they must be specifically studied and experimentally tested for protecting Pashto digital information.
- Assessment of Advanced Cryptographic Protocols in Local Contexts: The implementation of advanced encryption protocols like TLS (Transport Layer Security) and Zero-Knowledge Proofs should be explored in regional and local environments. These protocols are vital for enhancing confidentiality, trust, and security in digital communication networks. Their applicability and effectiveness must be thoroughly evaluated for systems operating in or related to the Pashto language.
- Extending Research to Multilingual Environments: Future studies should encompass multilingual settings to compare the effectiveness, security challenges, and resilience of encryption techniques across different scripts such as Arabic, Dari, and Pashto. Such comparative evaluations will help optimize and harmonize encryption methods in multilingual systems, thereby improving digital data protection in localized contexts.

Leveraging Machine Learning for Cryptographic Vulnerability Detection: Applying machine learning techniques to predict and detect weaknesses in encryption based on language-specific structural patterns presents a promising research direction. This approach can enable automated identification of high-risk areas and support proactive threat mitigation. Advancing such capabilities will strengthen digital security for the Pashto language and contribute significantly to the broader field of cybersecurity.

Acknowledgements
The authors thank the Faculty of Mathematics, Kabul University, for their technical support.
Funding information
No funding is available for the manuscript.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Al-Shabi, M. A. (2019). A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security. *International Journal of Scientific and Research Publications (IJSRP)*, *9*(3), p8779. https://doi.org/10.29322/ijsrp.9.03.2019.p8779

[2] Alabdulrazzaq, H., & Alenezi, M. N. (2022). Performance Evaluation of Cryptographic Algorithms: DES, 3DES, Blowfish, Twofish, and Threefish. *International Journal of Communication Networks and Information Security*, *14*(1), 51–61. https://doi.org/10.54039/ijcnis.v14i1.5262

[3] Alam Hossain, M., Biddut Hossain, M., Shafin Uddin, M., & Md Imtiaz, S. (2016). Performance Analysis of Different Cryptography Algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering*, *6*(3), 2277. www.ijarcsse.com

[4] Ali, K., Akhtar, F., Memon, S. A., Shakeel, A., Ali, A., & Raheem, A. (2020). Performance of Cryptographic Algorithms based on Time Complexity. *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies: Idea to Innovation for Building the Knowledge Economy, ICoMET 2020*, *August 2024*. https://doi.org/10.1109/iCoMET48670.2020.9073930

[5] Amirkhanova, D. S., Iavich, M., & Mamyrbayev, O. (2024). Lattice-Based Post-Quantum Public Key Encryption Scheme Using ElGamal's Principles. *Cryptography*, *8*(3), 31. https://doi.org/10.3390/cryptography8030031

[6] Asagba, P. O., & Nwachukwu, E. O. (2014). A Review of RSA Cryptosystems and Cryptographic Protocols. *West African Journal of Industrial and Academic Research*, *10*(1), 3–16.

[7] Astuti, N. R. D. P., Setiawan, D. P., & Hakika, D. C. (2023). Comparative Study of Elgamal and Luc Algorithm in Cryptographic Key Generation. *ASEAN Engineering Journal*, *13*(4), 61–68. https://doi.org/10.11113/aej.V13.19184

[8] ElGamal. (1985). *T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in IEEE T.* https://doi.org/10.1109/TIT.1985.1057074

[9] Kaushik, B., Malik, V., & Saroha, V. (2023). A Review Paper on Data Encryption and Decryption. *International Journal for Research in Applied Science and Engineering Technology*, *11*(4), 1986–1992. https://doi.org/10.22214/ijraset.2023.50101

[10] Mahto, D., & Yadav, D. K. (2017). RSA and ECC: A comparative analysis. *International Journal of Applied Engineering Research*, *12*(19), 9053–9061.

[11] Maqsood, F., Ahmed, M., Mumtaz Ali, M., & Ali Shah, M. (2017). Cryptography: A Comparative Analysis for Modern Techniques. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 8, Issue 6). www.ijacsa.thesai.org

[12] Marwaha, M., Singh, A., & Singh, T. (2018). Comparative analysis of cryptographic algorithms. *International Journal of Advanced Engineering Technology, E-Issn 0976-3945*, *4*(September), 2013–2016. https://www.researchgate.net/publication/327664102_COMPARATIVE_ANALYSIS_OF_CRYPTOGRAPHIC_HASH_FUNCTIONS

[13] Mishra, M. R., & Kar, J. (2017). a Study on Diffie-Hellman Key Exchange Protocols. *International Journal of Pure and Apllied Mathematics*, *114*(2). https://doi.org/10.12732/ijpam.v114i2.2

[14] Najem, D. F., Kareem, S. M., & Info, A. (2025). *Review of Diffie-Hellman and ElGamal Algorithms*. *4*(June), 57–65. https://doi.org/10.52940/ijici.v4i1.99

[15] Peranginangin, A. P. (2024). Application of Number Theory in Cryptography. *International Journal of Educational Research Excellence (IJERE)*, *3*(1), 67–76. https://doi.org/10.55299/ijere.v3i1.733

[16] Ratnadewi, Adhie, R. P., Hutama, Y., Saleh Ahmar, A., & Setiawan, M. I. (2018). Implementation Cryptography Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES) Method in Communication System Based Near Field Communication (NFC). *Journal of Physics: Conference Series*, *954*(1). https://doi.org/10.1088/1742-6596/954/1/012009

[17] Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice 7th Global Edition*.

[18] Suguna, S., Dhanakoti, V., & Manjupriya, R. (2016). A STUDY ON SYMMETRIC AND ASYMMETRIC KEY ENCRYPTION ALGORITHMS. *International Research Journal of Engineering and Technology*. www.irjet.net

[19] Teranishi, K., & Kogiso, K. (2021). ElGamal-type encryption for optimal dynamic quantizer in encrypted control systems. *SICE Journal of Control, Measurement, and System Integration*, *14*(1), 59–66. https://doi.org/10.1080/18824889.2021.1906016

[20] Thirupalu, U., & Reddy, E. K. (2019). Performance analysis of cryptographic algorithms in the information security. *IJERT. NCISIOT-2019 Conference Proceedings*, *8*(2), 64–69. https://doi.org/10.13140/RG.2.2.16273.51047

[21] Xie, Y. (2023). A comparative analysis of public key cryptographic algorithms: RSA, ELGamal, and elliptic curve encryption. *Theoretical and Natural Science*, *14*(1), 91–95. https://doi.org/10.54254/2753-8818/14/20240886

[22] Zhang, Q. (2021). An overview and analysis of hybrid encryption: The combination of symmetric encryption and asymmetric encryption. *Proceedings - 2021 2nd International Conference on Computing and Data Science, CDS 2021*, 616–622. https://doi.org/10.1109/CDS52072.2021.00111.