BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.056.55:004.451.7
DOI: 10.62660/bcstu/3.2025.24

Development and testing of the effectiveness of hybrid cryptographic algorithms for protecting personal data in mobile applications

Vitalii Yasenenko*
Master, Senior Software Developer
TP-Link
92618, 36 Technology Str., Irvine, USA
https://orcid.org/0009-0004-4801-9541

Abstract. With the rapid growth in the number of mobile applications and the volume of processed personal data, the need for effective means of the protection increased. The aim of the study was to develop hybrid cryptographic algorithms and to analyse the possibilities of the integration into mobile applications to enhance the protection of personal data. The study implemented hybrid cryptographic algorithms that combined symmetric and asymmetric encryption methods, modelled the operation using Python-based software modules, and assessed the efficiency considering the characteristics of the mobile environment. The main results showed that the combination of symmetric encryption algorithms Advanced Encryption Standard and ChaCha20 ensured complete data preservation during decryption and reduced the risk of compromise due to the two-layer encryption structure. It was also found that during testing of asymmetric methods, the Rivest-Shamir-Adleman algorithm successfully protected symmetric keys, and the Elliptic Curve Cryptography algorithm enabled both parties to compute a shared secret without transmitting the key itself, which improved resistance to interception. The results of the software implementation confirmed the identity of input and output data, demonstrating the reliability of the hybrid encryption approach. On Android platforms, Keystore, Bouncy Castle, Spongy Castle, and Conscrypt provided fast encryption using Advanced Encryption Standard (~275.6 milliseconds with hardware acceleration) and ChaCha20 (~2 milliseconds), as well as efficient key exchange via Elliptic Curve Cryptography (~15.8 milliseconds) compared to Rivest-Shamir-Adleman (~2,532.8 milliseconds), with support for post-quantum algorithms. On iPhone Operating System platforms, CryptoKit, CommonCrypto, and Open Secure Sockets Layer offered similar encryption speeds, while Secure Enclave optimised Elliptic Curve Cryptography, although post-quantum algorithms required additional optimisation. The hybrid approach reduced memory usage, making the algorithms effective for mobile devices. The obtained results could be used by mobile application developers to improve data protection in financial, medical, and corporate systems on smartphones running Android and iPhone Operating System platforms

Keywords: Advanced Encryption Standard; ChaCha20; Rivest-Shamir-Adleman; Elliptic Curve Cryptography; Android; iPhone Operating System

INTRODUCTION

In the context of the rapid spread of digital technologies, mobile devices became the primary environment for storing and transmitting confidential information, which necessitated the improvement of protection methods. Given the limited resources of mobile devices – particularly regarding energy consumption, performance, and memory – traditional cryptographic methods were not always effective in mobile environments. The problem was compounded by the fact that modern cyber threats use complex mechanisms to bypass protection, including traffic analysis, attacks on key infrastructure, data interception through compromised communication channels, or local interference. Since symmetric methods provided high speed but required

Article's History: Received: 14.05.2025; Revised: 29.07.2025; Accepted: 15.09.2025.

Suggested Citation:
Yasenenko, V. (2025). Development and testing of the effectiveness of hybrid cryptographic algorithms for protecting personal data in mobile applications. *Bulletin of Cherkasy State Technological University*, 30(3), 24-36. doi: 10.62660/bcstu/3.2025.24.

*Corresponding author

secure key exchange, and asymmetric ones ensured secure exchange but demanded more resources, there arose a need to develop hybrid solutions that would combine the advantages of both approaches.

For example, O. Borysenko & A. Tymoshenko (2024) focused on the risks of data leakage in cloud environments, emphasising the importance of encryption during storage and transmission, as well as the use of Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocols and access control to ensure the confidentiality of personal data. V. Semerenska (2025) analysed post-quantum cryptographic algorithms and concluded that lattice-based schemes – particularly Kyber and Nth-degree truncated polynomial ring unit – offered the best balance between performance and resistance to quantum attacks, also stressing the appropriateness of using hybrid solutions combining classical and post-quantum algorithms to protect critical systems. In addition, A. Sereda *et al*. (2023) analysed cryptographic algorithms used in mobile operating systems, reviewed approaches to protecting confidential information, disk space encryption, and end-to-end encryption, and proposed methods for evaluating the resilience and efficiency of cryptographic systems in mobile environments.

In turn, M.A. Sayed (2024) compared traditional cryptographic algorithms such as Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA), and hybrid models with machine learning-based models (autoencoders, Generative Adversarial Network, Random Forest) for encrypting personal data, demonstrating the advantages of the hybrid approach in balancing speed, security, and scalability. The study by G.A. Nwatuzie *et al*. (2025) proposed a hybrid cryptographic model combining AES, data encryption standard (DES), and Rivest's Cipher 6 to enhance encryption speed, scalability, and resistance to attacks in cloud environments, with support for automated key management and passwordless authentication. A. Muthaura & J. Kandiri (2024) developed and tested a cryptographic algorithm using Base64 and a fixed 512-bit length to enhance data protection, which showed optimal results in encryption speed, attack resilience, and plaintext vulnerability. Researchers H. Fan *et al*. (2022) also proposed a static "black-box" testing method for detecting the use of cryptographic algorithms in binary files without access to source code, revealing the widespread use of outdated and vulnerable algorithms such as Message Digest 4 and Secure Hash Algorithm 1.

The findings of R. Neve & R. Bansode (2024) presented the lightweight hybrid cryptographic algorithm Hybrid-SIMON-SPECKey, combining the SIMON round function and SPECK key schedule, which achieved 50% faster execution and 20% less memory usage compared to the original algorithms, while maintaining attack resilience under the avalanche effect criterion. The conclusions of A. Gour *et al*. (2024) confirmed the high effectiveness of the proposed hybrid cryptographic algorithm combining AES for symmetric encryption, Elliptic Curve Cryptography (ECC) for key exchange, and digital signature algorithm for digital signing in ensuring data confidentiality, integrity, and protection while maintaining computational efficiency. Furthermore, R.R. Pitale *et al*. (2024) underlined the importance of well-grounded cryptographic algorithm selection (including DES, 3DES, RSA, AES, International Data Encryption Algorithm, Blowfish, and homomorphic encryption) to ensure data confidentiality and integrity in the context of growing digital interaction, highlighting the role of algorithmic complexity and key management in countering modern threats.

The gaps in the cited studies lay in the insufficient focus on optimising hybrid cryptographic algorithms for mobile platforms, considering the hardware limitations, and in the limited analysis of the impact of such algorithms on performance and energy efficiency in mobile applications. In contrast, the present study focused on examining the prospects for applying hybrid cryptographic algorithms based on a mobile operating system to improve the protection of users' personal data. The research included the following objectives: to develop hybrid cryptographic algorithms combining symmetric and asymmetric encryption methods; to analyse the potential for implementing the developed algorithms in mobile operating system environments; and to carry out a comparative analysis of the effectiveness of hybrid solutions versus traditional cryptographic methods based on performance, energy consumption, and security levels.

MATERIALS AND METHODS

At the first stage of the study, a concept and basic implementation of hybrid cryptographic algorithms combining symmetric and asymmetric encryption methods were developed, taking into account the specifics of mobile environments. The main focus was placed on achieving a balance between cryptographic strength and computational efficiency. Within this stage, a justification for the hybrid approach was provided, where symmetric algorithms such as AES and ChaCha20 were used for primary data encryption, while asymmetric methods including RSA and ECC were used for secure key transmission. For clarity, a visualisation of the hybrid cryptographic algorithm using symmetric and asymmetric encryption was presented.

During testing, four separate software modules were created in Python using the cryptography library in the Visual Studio Code environment. The first programme implemented data encryption using AES-256 in Cipher Block Chaining mode with Public Key Cryptography Standard No. 7 padding. The second programme applied stream encryption using ChaCha20 with a unique nonce for sequential message protection. In turn, the third programme demonstrated encryption of the symmetric key using RSA with Optimal Asymmetric Encryption Padding and verified the correctness of its decryption. The fourth implemented key exchange via

the ECC-based Elliptic Curve Diffie-Hellman protocol, generating public and private key pairs for two parties and computing a shared secret.

At the second stage of the study, a comprehensive analysis of the integration of hybrid cryptographic algorithms into mobile applications was carried out, taking into account the peculiarities of the mobile environment based on comparative analysis methods and empirical evaluation of algorithm efficiency in a test environment. The focus was placed on assessing hardware constraints such as Central Processing Unit, Random Access Memory, energy consumption, and execution environments including Android Runtime (ART)/Dalvik and iOS runtime, as well as secure key storage capabilities (Keystore, Keychain). The analysis of mobile environment factors for cryptographic algorithms was conducted based on criteria such as computational resources, energy consumption, code execution characteristics across different mobile operating systems, support for libraries and Application Programming Interfaces, and secure key storage mechanisms (Ali *et al.*, 2025; Khalid *et al.*, 2025). For this purpose, a comparison of the performance of symmetric and asymmetric algorithms was conducted in terms of encryption time, key generation, memory usage, and energy consumption.

In addition, through structural-functional analysis, frameworks, and libraries supporting hybrid and post-quantum algorithm implementation were examined. For Android, Android Keystore and Bouncy Castle were considered, along with Spongy Castle and Conscrypt. Furthermore, the potential use of hybrid cryptography in combination with deep learning methods for detecting malware on Android was explored. In particular, a model combining a deep learning classifier with a cryptographic approach based on ECC and Blowfish was analysed. In the context of iOS, CryptoKit and CommonCrypto were reviewed. Additionally, tools such as Secure Enclave and OpenSSL were analysed. Based on this, a comparative analysis of mobile platforms was conducted using the above-mentioned tools, assessing the capabilities for implementing hybrid and post-quantum cryptographic algorithms. The analysis of iOS's comprehensive approach to data security covered information protection and SSL/TLS protocols (Kalphana *et al.* 2024; Prodduturi, 2025). Thus, this study carried out an analysis of the development, implementation, and adaptation of hybrid cryptographic algorithms, considering the characteristics of mobile platforms, which allowed for the identification of optimal combinations of encryption and key exchange methods to enhance the security and performance of mobile applications.

RESULTS

Development of hybrid cryptographic algorithms based on symmetric and asymmetric encryption methods

The protection of personal data in mobile applications requires a specific approach to cryptography, taking into account both the increased security requirements and the limited resources of devices. In this context, the development of hybrid cryptographic algorithms that combine symmetric and asymmetric encryption methods emerges as an effective solution that enables a balance between performance and cryptographic strength. The concept of hybrid encryption is based on the combination of two fundamental cryptographic principles. On the one hand, symmetric algorithms – such as AES (e.g., AES-256) or ChaCha20 – provide extremely fast encryption of large volumes of data with minimal processor load. On the other hand, asymmetric methods – particularly RSA or ECC – allow for the secure transmission of keys even in untrusted environments, as these methods rely on the computational complexity of factoring large numbers or computing the discrete logarithm on elliptic curves.

In the hybrid approach, data is encrypted using a symmetric method, which reduces processing time, while the symmetric key is encrypted using an asymmetric algorithm – and only the key is transmitted via an open channel. This combination minimises the risk of key interception without sacrificing speed, which is critically important in environments with limited mobile device performance. Such hybrid schemes can be effective in many security protocols, such as TLS; however, adapting these schemes to the specifics of mobile environments requires careful optimisation and testing. To better understand the main stages of hybrid encryption, a diagram is provided that illustrates the sequence of actions: from symmetric key generation to data encryption and the transmission of encrypted messages with a protected key, as well as subsequent decryption by the recipient (Fig. 1).

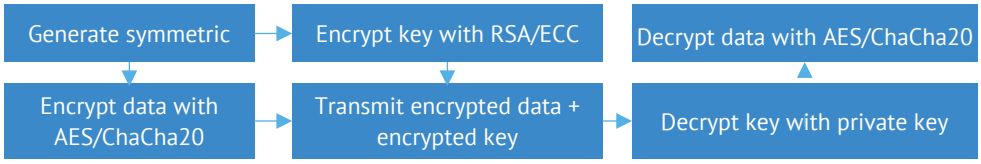

Figure 1. Scheme of operation of a hybrid cryptographic algorithm using symmetric and asymmetric encryption
Source: created by the author

The diagram illustrates the logic of combining symmetric and asymmetric encryption within a hybrid approach. One of the key stages in implementing a hybrid cryptographic algorithm is the correct organisation of the key generation and secure transmission process. In mobile environments, where resources

are limited, it is necessary to ensure not only cryptographic robustness but also the optimisation of this process in terms of performance and energy consumption. Within the hybrid approach, the symmetric key (to be used for the primary encryption of data) is generated on the sender's side using a cryptographically secure random number generator. This key is then encrypted with the recipient's public key (RSA or ECC) and transmitted as part of the message. That is, only the recipient, who possesses the corresponding private key, can decrypt the symmetric key and use it to decrypt the data. This approach effectively separates the tasks: symmetric algorithms perform the fast encryption/decryption of data, while asymmetric algorithms are used solely for the secure transmission of the key. Such a division of workload is critically important for mobile devices, where computational power and energy resources are limited.

Among symmetric cryptographic algorithms actively used in modern security systems, AES and ChaCha20 hold a special place. AES is a block-based symmetric encryption algorithm that provides reliable data protection and is widely used as an international standard. In contrast, ChaCha20 is a stream cipher optimised for high performance on devices without specialised cryptographic instructions, which makes it particularly effective in mobile applications with limited resources. Its strength lies in chaotic bit mixing and minimal use of complex operations. In Python, both algorithms can be implemented using the cryptography library, which supports flexible configuration of operating modes and encryption parameters (Fig. 2).

```
chacha_decryptor = Cipher(
    algorithms.ChaCha20(chacha_key, chacha_nonce),
    mode=None,
    backend=default_backend()
).decryptor()
aes_encrypted_restored = chacha_decryptor.update(chacha_encrypted)

print("🧩 [ChaCha20] Decrypted (AES encrypted message):", aes_encrypted_restored)

aes_decryptor = Cipher(
    algorithms.AES(aes_key),
    modes.CBC(aes_iv),
    backend=default_backend()
).decryptor()
padded_plaintext = aes_decryptor.update(aes_encrypted_restored) + aes_decryptor.finalize()

unpadder = padding.PKCS7(128).unpadder()
decrypted_message = unpadder.update(padded_plaintext) + unpadder.finalize()

print("✅ [AES-256] Final decrypted message:", decrypted_message)
```

Figure 2. Code snippet for implementing hybrid encryption using AES-256 and ChaCha20 in Python
Source: created by the author

The programme demonstrates the implementation of hybrid symmetric encryption, where data is first encrypted using AES-256 in Cipher Block Chaining mode with padding (extending the message to a multiple of the block length required for correct block cipher operation) according to Public Key Cryptography Standard No. 7, and then the result is additionally encrypted with the ChaCha20 algorithm. In the first stage, a random 256-bit key for AES and a 128-bit initialisation vector are generated. The data is padded to the required block length and encrypted. The resulting ciphertext is then re-encrypted using ChaCha20 with its own 256-bit key and a nonce (a unique number that does not repeat). During the decryption phase, the reverse sequence of actions is applied: ChaCha20 first removes the outer encryption layer, and then AES-256 decrypts the data, removing the padding to recover the original message. The programme demonstrates the step-by-step transformation of the input message: first, the original text is displayed, followed by its encrypted version after applying AES-256, then the result after additional ChaCha20 encryption. After the decryption steps, the AES-encrypted block is first restored, and finally, the original message is fully recovered (Fig. 3).

```
📝 Original message: b'Confidential data for hybrid symmetric encryption'
🔒 [AES-256] Encrypted message: b'\xfcw\xbeq\xfd\xa8y\x1a]%\t\xa7\xbb\xbek\xea\xb48\xd7\xf
aV\xce\x19\xa3n\xf7\xf2\xae\xe9\xc8C\x00jg\rO\xb5\r\xf6\xe1\xca\xda\xff\xf4\xb0\x95\x9ew\x
18b\xcb\xa8m\x9aQ\\\xf8;wf3\x82ju'
🌀 [ChaCha20] Encrypted message: b'\xea~\x1b\x9fp=\xeb\x86\xc7\n s\x9f\xcbR\xa1D\x9doE\xd0
\xc2\x90]\xd8(\xb0\xe8\xe5\xdf\x99c8\xb0B&u\xbbg`\xe0Y\xfcWY\xe4 v\xb4\x12\x12\x8f)\xdf\xe
c\x8b\xd6\xfe\xdd\xeaR\x88\xf8\x07'
🧩 [ChaCha20] Decrypted (AES encrypted message): b'\xfcw\xbeq\xfd\xa8y\x1a]%\t\xa7\xbb\xbe
k\xea\xb48\xd7\xfaV\xce\x19\xa3n\xf7\xf2\xae\xe9\xc8C\x00jg\rO\xb5\r\xf6\xe1\xca\xda\xff\x
f4\xb0\x95\x9ew\x18b\xcb\xa8m\x9aQ\\\xf8;wf3\x82ju'
✅ [AES-256] Final decrypted message: b'Confidential data for hybrid symmetric encryption'
```

Figure 3. Result of executing the hybrid symmetric encryption and decryption program
Source: created by the author

The obtained result confirms that the encryption and decryption of data using the combined symmetric approach was performed correctly: the encryption structure was preserved, and the output data were fully identical to the input message. Compared to traditional symmetric algorithms, such as the standalone use of AES or ChaCha20, the hybrid encryption method demonstrates a balanced compromise between security and performance. The use of AES in Cipher Block Chaining mode ensures high cryptographic strength, particularly due to standard support for hardware acceleration on modern mobile processors, which positively impacts encryption speed and energy consumption. In contrast, the additional application of ChaCha20, which is optimised for software implementations on resource-constrained devices, enhances resistance to attacks while only slightly increasing computational costs. This hybrid combination is especially relevant for mobile applications, where it is necessary to ensure both high data security and efficient use of device resources. Compared to the standalone use of either AES or ChaCha20, the hybrid algorithm may slightly lag in performance but provides more comprehensive protection, reducing the risk of potential vulnerabilities in individual methods. Thus, it is a promising solution for mobile platforms with heterogeneous hardware support for cryptography and varying performance and security requirements.

On the other hand, asymmetric algorithms such as RSA and ECC are important components of modern hybrid encryption, especially in the context of mobile applications. RSA is based on the mathematical complexity of factoring large prime numbers, and is most commonly used for the secure transmission of symmetric keys. At the same time, ECC, which is based on the algebra of elliptic curves, provides equivalent cryptographic strength with significantly shorter key lengths, making it more energy-efficient and suitable for mobile platforms. Both approaches can be integrated into hybrid cryptographic schemes to combine the strengths – the high speed of symmetric algorithms with secure key exchange through asymmetric methods. Similar to symmetric methods, these algorithms can also be implemented in Python using the cryptography library (Fig. 4).

```
29  def rsa_decrypt_key(private_key, encrypted_key):
30      decrypted_key = private_key.decrypt(
31          encrypted_key,
32          padding.OAEP(
33              mgf=padding.MGF1(algorithm=hashes.SHA256()),
34              algorithm=hashes.SHA256(),
35              label=None
36          )
37      )
38      return decrypted_key
39
40  def generate_ecc_keys():
41      private_key = x25519.X25519PrivateKey.generate()
42      public_key = private_key.public_key()
43      return private_key, public_key
44
45  def compute_shared_secret(private_key, peer_public_key):
46      shared_secret = private_key.exchange(peer_public_key)
47      derived_key = HKDF(
48          algorithm=hashes.SHA256(),
49          length=32,
50          salt=None,
51          info=b'handshake data',
52          backend=default_backend()
```

Figure 4. Results of executing the hybrid asymmetric encryption and decryption code
Source: created by the author

This code demonstrates two key mechanisms of hybrid asymmetric encryption: asymmetric encryption of a symmetric key using RSA and key exchange via ECC. First, a random symmetric key (32 bytes) is generated, which is then encrypted using the RSA public key with the Optimal Asymmetric Encryption Padding algorithm, based on SHA-256, ensuring enhanced resistance to attacks. Decryption is then performed using the private key, followed by a verification of the original and recovered keys, confirming the correctness of the encryption. Next, key pair generation for two parties (A and B) is demonstrated using the X25519 algorithm. Each party uses its private key and the other party's public key to compute a shared symmetric secret through the Elliptic Curve Diffie-Hellman protocol. The results show that both parties obtained identical keys, indicating a successful and secure exchange (Fig. 5).

This approach allows a shared secret to be established without transmitting the key itself, which significantly increases resistance to interception and manipulation in open environments. Moreover, the hybrid encryption method combining RSA and ECC has substantial advantages compared to using each of these methods individually. The traditional use of RSA alone for data protection provides a reliable level of security but is characterised by relatively large

key sizes and slower performance, which is especially noticeable on mobile devices with limited resources. In turn, using ECC alone offers high cryptographic strength with significantly shorter key lengths and better performance; however, its standalone use may complicate compatibility with existing protocols and may require additional configuration. The current approach combines the strengths of both algorithms: RSA effectively protects the symmetric key, while ECC ensures secure and efficient key exchange, enhancing the overall security and performance of the system. Thus, the hybrid approach optimises the balance between security, processing speed, and resource consumption, making it particularly appropriate for mobile applications and environments with heightened data protection requirements.

Figure 5. Results of executing the hybrid asymmetric encryption and decryption code
Source: created by the author

Analysis of the possibilities of integrating hybrid cryptographic algorithms into mobile applications and assessment of the effectiveness

The integration of hybrid cryptographic algorithms into mobile applications requires consideration of several specific constraints inherent to the mobile environment. The main factors affecting the implementation of cryptographic solutions on mobile devices include: limited hardware resources (Central Processing Unit, Random Access Memory, battery); specific runtime environments, such as ART/Dalvik, iOS runtime; and restrictions related to energy consumption and security maintenance. At the same time, these factors also encompass the availability of advanced hardware security tools (Secure Enclave, Advanced RISC Machine (ARM) Crypto Extensions), which open up new possibilities for cryptographic optimisation. Table 1 provides a comparative overview of the main constraints and capabilities of the mobile environment in the context of implementing hybrid cryptographic algorithms that combine symmetric and asymmetric encryption.

Table 1. Analysis of mobile environment factors for cryptographic algorithms

Factor	Limitation	Opportunities for symmetric algorithms	Opportunities for asymmetric algorithms
Central Processing Unit	Limited processing power of mobile processors requires efficient algorithms	AES: efficient with hardware acceleration (encryption time ~275.6 ms)	RSA: high load due to slow operations (key generation ~2,532.8 ms)
		ChaCha20: faster in software (~2 ms)	ECC: faster, lower load (~15.8 ms)
Random Access Memory	Limited RAM (2-8 GB) requires low memory usage	AES: low usage (~17.74 MB)	RSA: moderate memory usage
		ChaCha20: higher (~148.4 MB), needs optimisation	ECC: less usage due to compact keys
Energy consumption	Limited battery capacity requires energy-efficient algorithms	AES: high without hardware acceleration	RSA: high due to complex calculations
		ChaCha20: lower due to speed in software	ECC: lower due to smaller keys and faster operations
Android runtime	ART/Dalvik affects cryptographic operations, limiting performance	Algorithms are supported via Android Security library, AES is more efficient with hardware acceleration	Both algorithms are supported, ECC is faster and more efficient on ARMv8
iOS runtime	iOS runtime with CommonCrypto and Secure Enclave has limitations on implementation flexibility	Supported via CommonCrypto, ES is more efficient with Secure Enclave	Both algorithms are supported, ECC is more efficient due to hardware support
Libraries and Application Programming Interface	Limited support for complex cryptographic libraries in mobile operating systems	Available via Android Security and CommonCrypto, providing flexible integration	Supported through standard libraries, ECC is easier to integrate due to smaller keys
Secure key storage	Need to protect keys from unauthorised access	Use Keystore (Android) and Keychain (iOS) for secure storage of symmetric keys	Use Keystore/Keychain, ECC is more efficient due to smaller key sizes

Notes: ARMv8 – Advanced RISC Machine version 8
Source: created by the author

Thus, symmetric algorithms, particularly AES and ChaCha20, demonstrate high efficiency when hardware acceleration or software execution optimisation is available, whereas asymmetric methods (RSA, ECC) differ significantly in terms of resource load, with ECC being more suitable for mobile environments due to smaller key sizes and faster operations. Given the above characteristics of cryptographic algorithms, it is also important to understand the capabilities provided by modern mobile platforms for the implementation. Different operating systems, such as Android and iOS, offer the own cryptographic frameworks, libraries, and key storage and processing tools, which directly affect the adaptation and optimisation of hybrid solutions.

The Android platform provides developers with a wide range of cryptographic tools, particularly Android Keystore, which ensures secure key storage, and the Bouncy Castle library, which supports various cryptographic algorithms. The Spongy Castle and Conscrypt libraries are also important. These tools enable effective implementation of both symmetric and asymmetric encryption algorithms. In particular, the use of hybrid cryptographic models, such as the combination of ECC and Blowfish, in conjunction with deep learning, improves the accuracy of malware detection on Android devices. For example, a model combining deep learning (AlexNet classifier) with hybrid cryptography (ECC and Blowfish) for detecting ransomware on Android achieves a detection accuracy of 99.89%, outperforming traditional methods such as Graph Neural Network (94.76%), Convolutional Neural Network (95.76%), and Random Forest (96%) (Kalphana *et al*. 2024).

On the other hand, iOS offers a set of cryptographic frameworks, notably CryptoKit and CommonCrypto, which provide access to modern encryption, hashing, and digital signature algorithms. To enhance security, iOS also actively uses hardware means, such as the Secure Enclave, which securely stores keys in isolation and performs critical cryptographic operations, and OpenSSL, which offers flexible support for a broad range of cryptographic algorithms. For example, the comprehensive approach of iOS to data security includes protection of data "at rest", during transmission, and secure communication via SSL/TLS protocols (Prodduturi, 2025). The comparison of different mobile platforms was illustrated in Table 2.

Table 2. Comparative analysis of mobile platforms

Tool	Platform	Algorithm support	Features and limitations	Opportunities for hybrid and post-quantum algorithms
Keystore	Android	AES, RSA, ECC, partially Kyber, Dilithium	Hardware-secured key storage, limited support for ChaCha20 and post-quantum algorithms	Secure key storage for AES, RSA, ECC; post-quantum algorithms require external libraries
Conscrypt			Optimised for Android, integrated with ART, limited support for ChaCha20 and SPHINCS+	Fast for AES, ECC with ARMv8 Crypto Extensions; post-quantum algorithms require additional libraries
Bouncy Castle		AES, ChaCha20, RSA, ECC, Kyber, Dilithium, Falcon, SPHINCS+	Flexible, but resource-intensive, needs optimisation for mobile devices	Full support for hybrid (AES+ChaCha20, RSA+ECC) and post-quantum algorithms, but slower without hardware acceleration
Spongy Castle			A lighter version of Bouncy Castle, adapted for Android, but less active support	Efficient for hybrid and post-quantum algorithms on older devices, supports all algorithms
CryptoKit	iOS	AES, ChaCha20, ECC, partially Kyber	Modern, iOS-optimised, limited support for RSA and post-quantum algorithms	Suitable for hybrid schemes (ChaCha20+ECC), partial support for Kyber, requires OpenSSL for other post-quantum algorithms
Common-Crypto		AES, ChaCha20, RSA, ECC	Reliable but less flexible, no direct support for post-quantum algorithms	Full support for hybrid schemes, but post-quantum algorithms require external libraries
OpenSSL		AES, ChaCha20, RSA, ECC, Kyber, Dilithium, Falcon, SPHINCS+	Flexible but complex integration due to the closed iOS ecosystem, high resource intensity	Supports all hybrid and post-quantum algorithms, but needs optimisation for iOS

Notes: SPHINCS – Stateless Practical Hash-based Incredibly Nice Crypto Signature
Source: created by the author

Overall, Keystore is the standard Android mechanism for securely storing cryptographic keys in an encrypted, hardware-backed container, ensuring a high level of data protection. An addition to this is Conscrypt – a cryptographic provider from Google, specifically optimised for working with TLS and integrated into the ART runtime environment, which contributes to the efficient execution of cryptographic operations on Android devices. For extended algorithm support, Bouncy Castle was used – a universal cryptographic library offering a wide range of symmetric, asymmetric, and post-quantum algorithms, including Kyber, Dilithium, and SPHINCS+. For resource-constrained devices, Spongy Castle is available – a lightweight and Android-adapted version of Bouncy Castle that ensures compatibility with legacy devices while maintaining support for core algorithms.

On the iOS platform, CryptoKit plays an important role – a modern framework that supports ChaCha20, ECC, and is closely integrated with the Secure Enclave hardware module, enhancing the security of cryptographic operations. Also in use is CommonCrypto – a low-level C-library from Apple, which focused on implementing basic cryptographic functions with support for AES, RSA, and ECC, which provides a stable though less flexible implementation. No less important is OpenSSL – a powerful and flexible library supporting all major and post-quantum algorithms, including Falcon and Dilithium; however, its integration into iOS is often complicated by the closed nature of Apple's ecosystem. Thus, these tools create a broad field for selecting optimal cryptographic components when building hybrid security models, allowing the adaptation of algorithms to the specifics of a given mobile platform.

Therefore, on Android, the hybrid method ChaCha20 + AES provides fast encryption (~2 ms for ChaCha20, ~275.6 ms for AES with hardware acceleration), whereas RSA + ECC optimises key exchange (~15.8 ms for ECC versus ~2,532.8 ms for RSA), which is significantly faster than using RSA alone. On iOS, thanks to CryptoKit and Secure Enclave, ChaCha20 + AES achieves comparable encryption speed, and ECC reduces load compared to RSA, ensuring better energy efficiency. The separate use of AES or RSA is slower and less energy-efficient, especially without hardware acceleration, whereas the hybrid approach optimises performance and security, particularly through the use of Keystore (Android) and Keychain (iOS) for key protection. Moreover, hybrid algorithms reduce execution latency and resource consumption, which is critical for mobile devices. For instance, in the hybrid approach, ChaCha20 reduces reliance on hardware acceleration, unlike AES, and ECC provides faster, and more compact key exchange compared to RSA. Although individual algorithms like ChaCha20 (~148.4 MB memory usage) may impose greater memory load than AES (~17.74 MB), the hybrid method compensates for this through comprehensive security and adaptability to various hardware configurations, making it more effective for mobile applications.

DISCUSSION

The results of this study highlighted that hybrid cryptographic algorithms provide effective protection of personal data in mobile applications due to the combination of symmetric and asymmetric encryption and optimisation for the mobile environment. In turn, the study by V.B. Bhimanapati *et al.* (2024) applied artificial intelligence and machine learning to detect vulnerabilities and behavioural threats in mobile applications. Thus, both studies align: the present one – in the field of cryptographic protection, and the mentioned one – in the aspect of dynamic monitoring and adaptive security.

The results showed that the proposed hybrid cryptographic algorithms not only enhance the security of personal data but are also effectively adapted to the limited resources of mobile devices, ensuring performance, energy efficiency, and low memory usage. At the same time, the study by A. Grace (2025) focused on adaptive solutions for mobile security, which apply machine learning and behavioural analytics to detect threats in real time. Both approaches complement each other: the current study strengthens protection through cryptographic methods at the data level, while the cited study was aimed at active threat detection and prevention at the level of user behaviour and system activity.

The current conclusions also demonstrated that the integration of symmetric (AES, ChaCha20) and asymmetric (RSA, ECC) algorithms within hybrid cryptography allows achieving a balance between high security and performance, which is especially important for mobile applications with limited resources. Meanwhile, in the study by V.R. Kanagavalli & A. Meenakshi (2024), the main focus was placed on the theoretical analysis of the interrelation between machine learning and cryptography, in particular – how machine learning can improve cryptographic methods, or how encryption can be used to protect models and machine learning data. Both studies demonstrated the potential for interdisciplinary integration, albeit from different perspectives: the current research focused on the implementation of hybrid algorithms in mobile environments, whereas the mentioned study offered a conceptual generalisation of the interaction possibilities between machine learning and cryptography.

In addition, in the conducted study, AES-256 was implemented as a key element of hybrid symmetric encryption, providing high cryptographic strength, efficiency due to hardware acceleration, and adaptation to mobile environments with limited resources. Similarly, in the study by W. Abbas *et al.* (2025), AES was applied for message encryption, ensuring fast and reliable protection of personal messages in mobile networks, compatible with most Android devices. The results of both studies align, confirming the high efficiency of AES in mobile environments as a universal tool for the protection of confidential data.

The conclusions of this study emphasised that the implementation of hybrid cryptographic algorithms, particularly using AES, ChaCha20, RSA, and ECC, not only increases the level of protection of personal data in mobile applications but also takes into account the requirements for data confidentiality and security in the modern digital environment. Similarly, in the study by P.M. Bandara *et al.* (2025), the importance of cryptographic technologies such as encryption and anonymisation for complying with international data protection standards, particularly the General Data Protection Regulation (2016) and the California Consumer Privacy Act (2018), was highlighted. Both studies complement each other, showing that cryptographic solutions can serve as both a technical means of data

protection and a tool for ensuring compliance with privacy legislation.

The ChaCha20 algorithm analysed in this study is a high-performance stream cipher optimised for mobile applications with limited resources, where it, as part of a hybrid scheme with AES-256, provides a balanced compromise between security and performance, in particular due to its fast software implementation and moderate energy consumption, although it requires some memory optimisation. In the study by F.A. Fadhil *et al*. (2024), ChaCha20 was applied for encrypting data before hiding these data in video frames using the Laplacian of Gaussian algorithm, which demonstrated high speed, reliable protection, and preservation of video quality, confirmed by experimental metrics. Therefore, the results of the current study confirmed the conclusions of the mentioned study, demonstrating the versatility and efficiency of ChaCha20 for both mobile cryptography and multimedia information protection.

Regarding the RSA algorithm, the current results indicated that it provides a high level of security but involves significant computational costs and slow key generation, which limits its efficiency on mobile devices. Similarly, the authors Y. Gao *et al*. (2023), in the blockchain model, applied RSA for data encryption and digital signing, noting its high cryptographic strength but also emphasising the need for optimisation due to high computational load. This coincides with the current conclusions regarding the limitations of RSA in the mobile environment.

The results of this study generally align with the findings presented in the work by Z. Chen *et al*. (2023), which proposed a hybrid cryptographic algorithm Hybrid AES and ECC, combining symmetric data encryption using AES and asymmetric key encryption using ECC. As in the current study, the authors emphasised the efficiency of the hybrid approach in enhancing security, optimising key management, and ensuring a balance between performance and protection. Both studies demonstrated the advantages of hybrid cryptography under resource constraints: in this case – mobile applications, in the mentioned study – blockchain systems.

The conducted study emphasised that ECC provides high cryptographic strength, fast key generation (~15.8 ms), and low energy consumption in mobile applications. Moreover, in the study by Y. Yan (2022), ECC was noted as one of the strongest and most efficient modern cryptographies, capable of optimisation through hardware acceleration. Thus, both studies agree in the conclusion that ECC is the optimal choice for secure and efficient encryption under limited mobile resources. While the current work focused on the implementation of hybrid algorithms for the mobile environment, including RSA and ECC, the study by C. Gitonga (2025) analysed the risks for these algorithms in the context of quantum attacks and considered post-quantum alternatives such as CRYSTALS-Kyber and SPHINCS+. That is, the current work has an advantage in the applied aspect, as it offers adapted solutions for modern mobile platforms.

Whereas the present study focused on the implementation of hybrid cryptographic algorithms for the protection of personal data in mobile applications, the work by F. Ogwara *et al*. (2025) concentrated on the application of machine learning and stochastic methods for threat detection in mobile cloud computing environments. Both approaches are complementary: the current work proposed universal solutions for data encryption and storage, while the other study focused on dynamic monitoring and anomaly detection. The advantage of the current approach lies in its lower implementation complexity and compatibility with a wide range of mobile platforms without the need for cloud infrastructure.

In discussing the effectiveness of hybrid cryptographic schemes for protecting personal data in mobile applications, it is instructive to reference V.V. Lukichov *et al*. (2023), who proposed an adaptive multi-layer protection method combining steganography and cryptography. Their approach demonstrates how these two techniques can synergistically enhance resistance to attacks, assessed using metrics such as MSE and PSNR. Compared to their work, the proposed hybrid scheme similarly integrates symmetric encryption with asymmetric authentication mechanisms (e.g. AES with RSA or ECC), aiming to achieve a balance of confidentiality and integrity tailored to mobile environments. Additionally, the study by V.I. Malinovskyi *et al*. (2024) presented a mathematical model for evaluating cyber-threats and informational impacts within microcontroller systems. Their analysis of specific risk vectors and threat dynamics provides a useful parallel to the mobile application context described in this study, where separation of key management (via ECC/RSA) and symmetric encryption modules enhances both scalability and robustness.

In general, this study implemented a hybrid symmetric encryption approach combining AES-256 and ChaCha20, which enabled a balance between performance and security on mobile devices. Compared to the results of M. Vinothkumar & S. Ram (2025), who proposed a modified AES with chaotic key generation to enhance the security of electronic documentation, the current implementation demonstrates lower computational costs and better suitability for the mobile environment due to AES hardware acceleration support and the software efficiency of ChaCha20. Although the mentioned approach strengthens the dynamic nature of keys, it requires more complex integration into mobile platforms. In turn, compared to the study by W.B. Nugroho *et al*. (2024), in which ChaCha20 was successfully applied for image encryption, the current approach was aimed not only at images but at universal encryption of textual data in mobile applications, providing additional resistance through cascading encryption and key storage via Keystore/Keychain.

The conclusions of this study regarding RSA (key generation ~2,532.8 ms) confirmed that not only its high reliability but also its limitations in mobile

environments due to significant computational costs, which aligns with the findings of J.C. Vimala (2023), who emphasised its advantages in cloud security but acknowledged implementation complexity. In turn, R. Alsowail (2025) demonstrated the efficiency of ECC in video encryption with an encryption time of 4.49 seconds and security of 98.5%. At the same time, the current approach using ECC (Central Processing Unit ~15.8 ms) confirmed its speed and resource efficiency for mobile applications, making the combined use of RSA and ECC an optimal solution.

Additionally, the current conclusions regarding the hybrid use of cryptographic algorithms are consistent with the results of the study by Z. Yasqi *et al*. (2025), which confirmed the efficiency of combining symmetric (ChaCha20) and asymmetric (RSA) algorithms to ensure reliable data protection against various cyber threats. Both studies note successful encryption and decryption operation, as well as confirm the system's resistance to unauthorised access. Meanwhile, the present study focuses on performance optimisation and energy efficiency in the mobile environment, which is a logical continuation of the directions proposed by the authors for further development – improving algorithm efficiency and reducing execution time.

Thus, the results of the study confirmed that hybrid cryptographic algorithms are an effective and practical solution for the protection of personal data in mobile applications, ensuring a balance between security, performance, and resource efficiency. The proposed approaches were adapted to the specifics of the mobile environment and meet modern requirements for confidentiality and information protection. The implementation of such solutions will contribute to raising the overall level of cybersecurity in mobile technologies.

CONCLUSIONS

At the first stage of the study, a concept of hybrid encryption was proposed, combining symmetric (AES, ChaCha20) and asymmetric (RSA, ECC) cryptographic algorithms with the aim of achieving a balance between performance, security, and efficient use of mobile device resources. The results demonstrated the practical implementation of two variants of the hybrid approach in Python: symmetric double encryption using AES-256 and ChaCha20, as well as a scheme for secure transmission of a symmetric key using RSA and key exchange based on ECC (X25519). The obtained results emphasised the correctness of both approaches, confirming the feasibility for mobile applications, where limited computational resources require optimisation of cryptographic solutions without compromising the level of personal data protection.

The second stage of the study showed that effective integration of hybrid cryptographic algorithms into mobile applications is only possible with a deep understanding of the mobile environment's specifics. For example, symmetric algorithms (AES, ChaCha20) demonstrated high performance on devices with hardware support – AES achieved encryption times of ~275.6 ms with low memory usage (~17.74 MB), whereas ChaCha20 provided even higher speed (~2 ms), although it consumed more memory (~148.4 MB). In the context of asymmetric algorithms, ECC proved significantly more efficient than RSA: ECC key generation took ~15.8 ms versus ~2532.8 ms for RSA, which also positively affected power consumption and overall system load. Accordingly, hybrid models enable an optimal balance between speed, security, and resource consumption. Platform-specific features, such as the use of Android Keystore or iOS Keychain, support for Secure Enclave, ARM Crypto Extensions, further enhance these advantages, especially on modern devices. Analysis of such tools confirmed that Android provides broad support for both classical and post-quantum algorithms (via Bouncy Castle, Spongy Castle, Conscrypt), while iOS offers a more tightly controlled but hardware-optimised environment (CryptoKit, CommonCrypto). Thus, the proposed hybrid approach, combining symmetric encryption ChaCha20 or AES with asymmetric ECC or RSA, demonstrated better efficiency than traditional algorithms in terms of both speed and energy efficiency.

A limiting aspect of this study is that testing was carried out on a limited number of devices and emulators, which does not allow for a full coverage of the variability of mobile platforms and scenarios of the practical use. Also, performance analysis of the algorithms was not conducted in the context of different hardware configurations, particularly with regard to long-term load and changes in power consumption under real operating conditions. In this regard, it is advisable in further studies to expand the experimental base through testing on various devices and long-term monitoring of resource consumption, as well as to analyse the possibilities of adaptive integration of hybrid and post-quantum cryptographic algorithms depending on the dynamics of the environment and security threats.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abbas, W., Joshua, S.R., Abbas, A., & Lee, J.-H. (2025). An end-to-end GSM/SMS encrypted approach for smartphone employing advanced encryption standard (AES). *ArXiv*. doi: 10.48550/arXiv.2503.18859.

[2] Ali, A.K., Raza, A., & Arif, H. (2025). AI-driven dynamic selection of post-quantum algorithms for mobile application security. *Asian Bulletin of Big Data Management*, 5(2), 51-62. doi: 10.62019/eq34ar93.
[3] Alsowail, R. (2025). Advanced video encryption using the opposition lotus effect-elliptic curve cryptography in signal processing applications. *Signal Image and Video Processing*, 19(5), article number 409. doi: 10.1007/s11760-025-03899-x.
[4] Bandara, P.M., Abeyrathne, P., & Sandirigama, M. (2025). Cryptographic technologies for guaranteeing compliance of data privacy to international data protection laws – a preliminary study. In D. Dahanayake & M. Rabindrakumar (Eds.), *Transformative applied research in computing, engineering, science and technology* (pp. 237-243). London: CRC Press. doi: 10.1201/9781003616368-32.
[5] Bhimanapati, V.B., Jain, S., & Pandian, P.K. (2024). Security testing for mobile applications using AI and ML algorithms. *Journal of Quantum Science and Technology*, 1(2), 44-58. doi: 10.36676/jqst.v1.i2.15.
[6] Borysenko, O., & Tymoshenko, A. (2024). Overview of personal data protection methods in the cloud environment. *Infocommunication and Computer Technologies*, 1(7), 31-34. doi: 10.36994/2788-5518-2024-01-07-04.
[7] California Consumer Privacy Act. (2018, June). Retrieved from https://oag.ca.gov/privacy/ccpa.
[8] Chen, Z., Gu, J., & Yan, H. (2023). HAE: A hybrid cryptographic algorithm for blockchain medical scenario applications. *Applied Sciences*, 13(22), article number 12163. doi: 10.3390/app132212163.
[9] Fadhil, F.A., Tawfiq, F., & Thamer, M. (2024). Enhancing data security using laplacian of gaussian and Chacha20 encryption algorithm. *Journal of Intelligent Systems*, 33(1), article number 20240191. doi: 10.1515/jisys-2024-0191.
[10] Fan, H., Meng, L., Zheng, F., Mingyu, W., & Bowen, X. (2022). Black-box testing of cryptographic algorithms based on data characteristics. In J. Lin & Q. Tang (Eds.), *Applied cryptography in computer and communications* (pp. 153-169). Cham: Springer. doi: 10.1007/978-3-031-17081-2_10.
[11] Gao, Y., Guo, L., & Zhang, T. (2023). Exploring and envisioning the application of blockchain technology for privacy data protection. *Applied and Computational Engineering*, 19(1), 123-131. doi: 10.54254/2755-2721/19/20231020.
[12] General Data Protection Regulation. (2016, May). Retrieved from https://gdpr-info.eu/.
[13] Gitonga, C. (2025). The impact of quantum computing on cryptographic systems: Urgency of quantum-resistant algorithms and practical applications in cryptography. *European Journal of Information Technologies and Computer Science*, 5(1), 1-10. doi: 10.24018/compute.2025.5.1.146.
[14] Gour, A., Malhi, S.S., Singh, G., & Kaur, G. (2024). Hybrid cryptographic approach: For secure data communication using block cipher techniques. *E3S Web of Conferences*, 556, article number 01048. doi: 10.1051/e3sconf/202455601048.
[15] Grace, A. (2025). *Advancements in mobile device security: Developing AI-powered applications for enhanced protection*. Retrieved from https://www.researchgate.net/publication/389173473_Advancements_in_Mobile_Device_Security_Developing_AI-Powered_Applications_for_Enhanced_Protection.
[16] Kalphana, K.R., Aanjankumar, S., Surya, M., Ramadevi, M.S., Ramela, K.R., Anitha, T., Nagaraj, N., & Ramaswamy, K. (2024). Prediction of android ransomware with deep learning model using hybrid cryptography. *Scientific Reports*, 14(1), article number 22351. doi: 10.1038/s41598-024-70544-x.
[17] Kanagavalli, V.R., & Meenakshi, A. (2024). A survey of cryptographic data protection and machine learning. In J.A. Ruth, V.G. Mahesh, P. Visalakshi, R. Uma & A. Meenakshi (Eds.), *Machine learning and cryptographic solutions for data protection and network security* (pp. 1-11). London: IGI Global. doi: 10.4018/979-8-3693-4159-9.ch001.
[18] Khalid, R., Najat, Z., & Sayda, S.J. (2025). A hybrid approach to cloud data security using ChaCha20 and ECDH for secure encryption and key exchange. *Kurdistan Journal of Applied Research*, 10(1), 66-82. doi: 10.24017/science.2025.1.5.
[19] Lukichov, V.V., Baryshev, Y.V., Kondratenko, N.R., & Malinovskyi, V.I. (2023). Adaptive multi-layer information protection method combining steganography and cryptography. *Information technology and computer engineering*, 3, 4-11. doi: 10.31649/1999-9941-2023-58-3-4-11.
[20] Malinovskyi, V.I., Kupershtein, L.M., & Lukichov, V.V. (2024). A mathematical model for assessing cyber-threats and informational impacts in microcontrollers. *Information technology and computer engineering*, 59(1), 69-82. doi: 10.31649/1999-9941-2024-59-1-69-82.
[21] Muthaura, A., & Kandiri, J. (2024). Data protection in healthcare information systems using cryptographic algorithm with Base64 512 bits. *Open Journal for Information Technology*, 7(1), 11-22. doi: 10.32591/coas.ojit.0701.02011m.
[22] Neve, R., & Bansode, R. (2024). Attack analysis on hybrid-SIMON-SPECKey lightweight cryptographic algorithm for IoT applications. *Indian Journal of Science and Technology*, 17(10), 932-940. doi: 10.17485/IJST/v17i10.2811.
[23] Nugroho, W.B., Susanto, A., Sari, A., Rachmawanto, E.H., & Doheir, M. (2024). A robust and imperceptible for digital image encryption using Chacha20. *Jurnal Teknik Informatika*, 5(2), 397-404. doi: 10.52436/1.jutif.2024.5.2.1470.

[24] Nwatuzie, G.A., Ijiga, O.M., Idoko, I.P., Enyejo, L.A., & Ali, E.O. (2025). Design and evaluation of a user-centric cryptographic model leveraging hybrid algorithms for secure cloud storage and data integrity. *American Journal of Innovation in Science and Engineering*, 4(2), 49-65. doi: 10.54536/ajise.v4i2.4482.
[25] Ogwara, F., Petrova, K., Yang, M.L., & MacDonell, S.G. (2025). Mindpres: A hybrid prototype system for comprehensive data protection in the user layer of the mobile cloud. *Sensors*, 25(3), article number 670. doi: 10.3390/s25030670.
[26] Pitale, R.R., Tajane, K.D., Mahajan, P.B., Nehate, N.S., Mulimani, A.J., & Lokhande, D.S. (2024). Cryptographic algorithm development and application for encryption and decryption. In D. Goyal, A. Kumar, D. Singh, M. Paprzycki, P. Jain, B.B. Gupta & U.P. Singh (Eds.), *ICIMMI '23: Proceedings of the 5th international conference on information management & machine intelligence* (article number 27). New York: Association for Computing Machinery. doi: 10.1145/3647444.3647853.
[27] Prodduturi, S.M. (2025). Cryptography in iOS: A study of secure data storage and communication techniques. *International Journal on Science and Technology*, 16(1). doi: 10.71097/IJSAT.v16.i1.1403.
[28] Sayed, M.A. (2024). A comparative study of machine learning-based and traditional cryptographic methods for personal data encryption. *ResearchGate*. doi: 10.13140/RG.2.2.33661.88803.
[29] Semerenska, V. (2025). Quantum-resistant cryptographic algorithms for critical infrastructures. *Visnyk of Kherson National Technical University*, 2(1), 204-209. doi: 10.35546/kntu2078-4481.2025.1.2.27.
[30] Sereda, A., Datsenko, I., Pavlenko, V., & Samarai, V. (2023). Stability and efficiency of cryptographic algorithms used in mobile devices. *Infocommunication and Computer Technologies*, 2(4), 178-190. doi: 10.36994/2788-5518-2022-02-04-21.
[31] Vimala, J.C. (2023). Securing cloud environments: A comparative analysis of RSA and MRGA for enhanced data protection. *Journal on Cloud Computing*, 10(1), 38-46. doi: 10.26634/jcc.10.1.19937.
[32] Vinothkumar, M., & Ram, S. (2025). Blockchain-based modified AES with chaotic random key generation for secured E-medical data sharing. *CLEI Electronic Journal*, 28(2). doi: 10.19153/cleiej.28.2.14.
[33] Yan, Y. (2022). The overview of elliptic curve cryptography (ECC). *Journal of Physics Conference Series*, 2386, article number 012019. doi: 10.1088/1742-6596/2386/1/012019.
[34] Yasqi, Z., Hayaty, N., & Bettiza, M. (2025). Cryptography of Chacha20 and RSA algorithms for text security. *Journal of Computer Networks Architecture and High Performance Computing*, 7(1), 290-301. doi: 10.47709/cnahpc.v7i1.5345.

Розробка та тестування ефективності гібридних криптографічних алгоритмів для захисту персональних даних у мобільних додатках

Віталій Ясененко

Магістр, старший розробник програмного забезпечення
TP-Link
92618, вул. Технологій, 36, м. Ірвайн, США
https://orcid.org/0009-0004-4801-9541

Анотація. Зі стрімким зростанням кількості мобільних додатків та обсягу оброблюваних персональних даних підвищується потреба у впровадженні ефективних засобів їхнього захисту. Мета роботи полягала в розробці гібридних криптографічних алгоритмів і в аналізі можливостей їх інтеграції в мобільні додатки для підвищення рівня захисту персональних даних. У дослідженні реалізовано гібридні криптографічні алгоритми, які поєднують симетричні та асиметричні методи шифрування, проведено моделювання їхньої роботи на прикладі програмних модулів мовою Python, а також здійснено оцінку їх ефективності з урахуванням характеристик мобільного середовища. Основні результати показали, що поєднання алгоритмів симетричного шифрування Advanced Encryption Standard і ChaCha20 забезпечує повне збереження даних при дешифруванні та дозволяє зменшити ризики компрометації завдяки двошаровій структурі шифрування. Також встановлено, що під час тестування асиметричних методів алгоритм Rivest-Shamir-Adleman успішно захищає симетричні ключі, а алгоритм Elliptic Curve Cryptography забезпечує обчислення спільного секрету двома сторонами без передачі самого ключа, що підвищує стійкість до перехоплення. Результати програмної реалізації підтвердили ідентичність вхідних і вихідних даних, що засвідчує надійність гібридного підходу до шифрування. З іншого боку, на Android платформах Keystore, Bouncy Castle, Spongy Castle і Conscrypt забезпечують швидке шифрування Advanced Encryption Standard (~275,6 мілісекунд з апаратним прискоренням) і ChaCha20 (~2 мілісекунди), а також ефективний обмін ключами через Elliptic Curve Cryptography (~15,8 мілісекунд) порівняно з Rivest-Shamir-Adleman (~2532,8 мілісекунд), з підтримкою постквантових алгоритмів. На iPhone Operating System платформах CryptoKit, CommonCrypto і Open Secure Sockets Layer забезпечують схожу швидкість шифрування, а Secure Enclave оптимізує Elliptic Curve Cryptography, хоча постквантові алгоритми потребують додаткової оптимізації. Гібридний підхід знижує використання пам'яті, що робить алгоритми ефективними для мобільних пристроїв. Отримані результати можуть використовувати розробники мобільних додатків для покращення захисту даних у фінансових, медичних та корпоративних системах на смартфонах з операційними системами Android і iPhone

Ключові слова: розширений стандарт шифрування; ChaCha20; Rivest-Shamir-Adleman; криптографія на основі еліптичних кривих; Android; операційна система iPhone