

Paper Type: Original Article

Enhancing Data Security in Cloud-Based Engineering Systems Using Cryptographic Algorithms

Seyyed Esmacil Najafi* 

Department of Industrial Engineering, Faculty of Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran; najafi1515@yahoo.com.

Citation:

Received: 28 December 2024

Revised: 01 February 2025

Accepted: 10 May 2025

Najafi, S. E. (2025). Enhancing data security in cloud-based engineering systems using cryptographic algorithms. *Computational Engineering and Technology Innovations*, 4(2), 84-98

Abstract

Cloud systems of applications and platforms where all of the information is scattered, the distributed computing, real-time data exchange and collaborative designing environment that forms a basic part of cloud engineering systems actually leads to higher data security for safe and confidential operations. Because these platforms manage highly sensitive engineering assets which range from design files, simulation outputs, sensor logs and proprietary design models. They are vulnerable to threats such as sniffing during transport, tampering of data and unauthorized access. To address these problems, we propose a United Cryptographic Security Framework which authentication, integrity and enhances privacy of cloud-based engineering workflows. The framework utilizes SSL/TLS Hybrid system models of RSA-Rivest-Shamir-Adleman, cHaCha20 and SHA-3-Secure Hash Algorithm 3 (for secure key exchange, Authentication encoding for high speed engineering data streams encryption and also for a sustainable robust hashing & integrity verification process). RSA guarantees that session keys can only be acquired by legitimate parties, and ChaCha20 efficiently encrypts large engineering data, and is resilient to timing attacks. SHA-3 provides an extra level of security by creating collision resistant digests, allowing for the detection of unauthorized manipulation of engineering files or exchanged data. Security analysis and performance test show that the proposed cryptosystem can enhance the security of cloud-based engineering environment with moderate computation cost. Experimental results demonstrate a significant increase in the encryption speed over traditional AES-based schemes, highest integrity verification capacity compared to SHA-3 hashing and secure session establishment with the help of RSA key management. In general, the integrated RSA-ChaCha20-SHA3 structure provides a lightweight and scalable security mechanism that is well-suited for today's cloud-scale engineering systems with guaranteed secure data storage.

Keywords: Cloud based engineering system, Rivest-shamir-adleman, ChaCha20, Secure hash algorithm 3, Cryptographic security.

1 | Introduction

Cloud-based engineering systems have become an indispensable infrastructure for the modern industry, supporting scalable computing, distributed data storage, collaborative design and real-time simulation. Such platforms also enable more complex functions like CAD/CAM processing, IoT-based industrial monitoring,

digital twins and high-performance computing which in turn lead to generation and transfer of huge amount of sensitive data. Increasing cloud-based engineering work causes increased the potential for cyberrisks due to a broader distribution of cloud systems and multitasking use by numerous users [1]. Attackers often rely on security gaps in the cloud APIs, VMs, hypervisor layers, storage components and communication links, which can lead to unauthorized access to sensitive engineering information [2]. Hence there has been a fundamental need to set-up strong, multilayer security in order that cloud-based engineering applications can be secured. *Fig. 1*; Common threats which can threaten to cloud environments. These six material risks of data breach, porous APIs, insider threats, virtualization weaknesses, weak access controls and multi tenancy vulnerabilities are all large attack surfaces that can put your engineering systems' Confidentiality, Integrity and Availability (CIA) at risk [3]. Cloud-based workflow tends to add greater risk than traditional on-premises infrastructure because the systems can be accessed from anywhere, having a shared architecture.

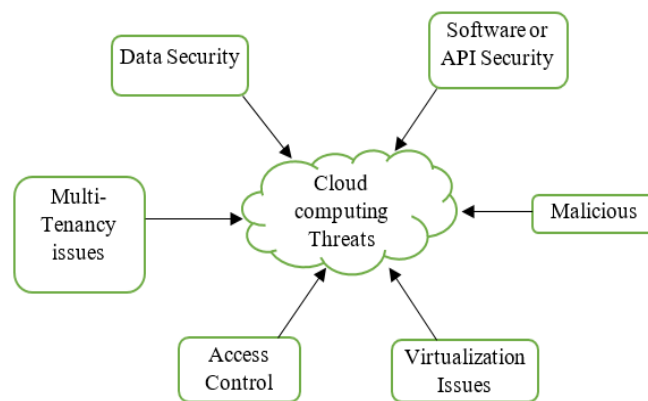


Fig. 1. Major data security threats in cloud computing.

In cloud-based engineering systems, security threats do not always come from external attacker but may be due to malpractice or miss configuration. Engineering work-flow may be composed of sensitive project designs, simulation models under copyright, sensor and actuator data history and computed indices. Conventional security approaches are not equipped well to address the needs of dynamic, distributed and multi-tenant cloud environments because of their limitations in scalability and computational overhead [4]. Therefore, efficient cryptographic solutions providing a high security strength without a loss of system performance are important in engineering environments with real-time demands. Cloud security frameworks are based upon cryptographic algorithms implementing the principles of confidentiality, integrity and authentication. Public key cryptography guarantees the security of the key management, while the symmetric ciphers grant a high-speed data encryption. Hashing functions can be used to ensure the integrity (i.e., noncorruptibility) of data, where modification of contents during computation or storage can be detected [5]. The above methods together confront data eavesdrop, tamper attack, replay attack and inside misemployment in cloud. In this paper, a compact cryptographic toolkit is presented for cloud-based engineering frameworks: such a proposed framework integrates RSA, ChaCha20 and SHA-3. RSA ensures the secure key and user data exchange between the client and cloud server, to avoid any possibility of illegal eavesdropping into session keys legally exchanged [6]. ChaCha20 is selected for encryption, which works efficiently and has extremely low delay as well as no illegal timing which can avoid side channel attack, so it is able to realize the large engineering data process and real-time rule digital twin processing [7]. SHA-3 supports strong integrity checking, and can detect data tampering while in transmission and when stored in cloud [8]. By combining these three cryptographic mechanisms, the framework improves the confidentiality, integrity and origin of engineering paths. RSA controls the initiation of secure communication, ChaCha20 encrypts dynamic engineering data with high efficiency and SHA-3 guarantees unceasing integrity checking. This integrated solution provides comprehensive protection against cloud incidents, all while keeping development operations intact. As cloud/edge/IoT-based engineering systems are moving toward a higher

degree of autonomy and distributed intelligence, cryptographic-centric security schemes become a pivotal pillar to safeguard next gen engineering infrastructures [9].

2 | Literature Review

Virushabadoss and Anithaashri designed a security framework for enhancing the security of data shared among cloud users. Fuzzy computation was employed to quantify the client's trust and HDN distribution/revocation in cloud, respectively. It also protected data access in mobile cloud through an authenticated users' data access which only verified user could have access to the sensitive data [10]. It also ensured confidentiality of the sensitive information at mobile cloud platforms and protected it from unauthorized access, misuse, and malicious tampering. In general, the method improved the security of mobile cloud systems by combining fuzzy trust evaluation with secure access control mechanisms. Ramachandran et al. [11] developed a cloud security model for the identification of cyber threats. The procedure requires a series of stages, such as data preprocessing, feature extraction and normalization, optimal parameter estimation, prediction. An enhanced PCA approach was employed to extract the representative information from the pre-processed data. Then, the hybrid GSCSO procedure was applied to select the optimum features during training and testing. The IHNN was then used to determine if the network traffic is normal or malicious. The model was compared with known bench mark intrusion detection datasets like NSL-KDD, BoT-IoT, KDD Cup'99 and CICIDS 2017 dataset. Performance metrics such as detection accuracy and F1 score were used to evaluate the model, which was compared with alternative methods. For all datasets, the GSCSO-IHNN system yielded 99.5% accuracy and 0.999 F1-score in average. On the whole, these results indicated that this model outperformed other existing security models. To sum up, the research supported by delivering a sound, accurate and efficient approach for cyber-attack detection to enhance cloud security. In conceived a Hybrid Elliptic Curve Cryptography (HECC) based cloud security technique, which was invented by Rao and Sujatha [12]. Here, keys were generated on a lightweight Edwards curve that allowed faster computation and led to smaller resource consumption. For personalization and elimination of keys to identity mismatches, Identity Based Encryption (IBE) was used for transformation from the generated secret key. Also, a key reduction technique was incorporated to reduce the key size and to let increase of AES encryption speed. The public keys were shared securely across the network via Diffie Hellman key exchange to encrypt over the wire. The HECC system was evaluated based on throughput, key generation time, encryption time, and decryption time. It was found that this security model performed better than other previous methods in all the evaluations. Key generation cost 0.000025 seconds, encryption spent 0.00349 s, which indicates the high efficiency of our system. Moreover, the performance of the framework can reach a throughput up to 693.10 kB/s, implying its applicability for secure real-time communication in public cloud. In general, the proposed one achieved higher speed, lower computational overhead and better security than the existing cloud encryption schemes. Singh et al. [13] proposed a privacy protected framework that combines homomorphic encryptions, differential privacy and secure multi-party computation step by step for users to outsource the data in a safe way on the cloud without leaking private knowledge. It combined three cryptographic methods-Homomorphic Encryption (HE) for computing over encrypted information, Differential Privacy (DP) so the servers could add limited noise to query outputs data items to prevent individual disclosures and Secure Multi-Party Computation that enabled multiple cloud servers to jointly computes a function over their inputs without revealing sensitive details. Such integrated scheme guaranteed that confidentiality was protected during storage, processing as well as multi server computation. It was created to be fast and scalable for versatile applications including the healthcare, financial, and governmental sectors. The results from the performance evaluation showed that, for small to medium-sized datasets, the proposed mechanism could work well and execution time, communication overhead and storage would grow approximately proportionally to dataset's size while scalability and efficiency of operation were good. Compared to baseline systems, this approach dramatically outperformed privacy unaware models and was just a bit slower than encrypted systems. In conclusion, the proposed mechanism provided a good level of security against different attacks and demonstrated that it holds strong promise towards the future of

privacy preserving cloud computing by allowing securely stored and shared private data. Reddy et al. [14] proposed an authentication scheme for secure cloud data storage. It added two primary properties: keeping users anonymous and being resilient from impersonation attacks against either users or cloud servers. User information including usernames and biometric data was anonymized using secure hashing so that the cloud couldn't track or misuse personal identities. On top of that, user sessions were automatically logged in and out during login as well as logout which ensured the cloud server could not follow what users did or oversee sensitive material. The paper presented full transaction that include user registration and login checks as well as safe file download/upload and session control using modern cryptographic tools such ECC (Elliptic Curve Cryptography) and CHACHA. The test and comparisons of the algorithms' performance also demonstrated that they can be effectively worked well for real applications. In conclusion, the new approaches enhanced data safety and made cloud data available for use and sharing securely, without any hassle. Hossain and Al Hasan [15] proposed a hybrid authentication technique for accessing sensitive information that utilises biometric checks along with encrypted decision. To have a robust and secure verification system the study used fingerprint recognition as the biometric technique and AES-Advanced Encryption Standard was used as a trusted encryption algorithm. The goal of the research was to ensure that only authorized users were able to access data stored in cloud computing. With biometric identification ensuring the bearing of authentication, and cryptoprotection ensuring the integrity of transactions taking place between user side devices and service providers, our system's architecture instead provided an improved authentication mechanism for cloud computing. Overall, this approach enhanced security and reduced the risk of unauthorized access to cloud-based systems [16]. Its simplicity and independence of the number of clients assigned proved that it could be used for large scale problems in a practical way. Overall, the new approach has a potential to enhance cloud security with little overhead to the operational cost. Secure model ECSHB is introduced by Verma and Kanrar, which ensures the securing of data, privacy and authentication in cloud [17]. For the realization of secure access and resource protection the methodology is based on biometric authentication along with Hyperelliptic Curve Cryptography (HECC). ECSHB provides strong security and does not consume as much processing power which means less computational complexity. The performance of the model was tested based on widely used metrics like FNMR, FMR, biometric similarity score and recognition rate. ECSHB proved also to be secure in terms of confidentiality through security threat model analysis, and it was resistant against possible attacks such as collision attack, replay attack and non-repudiation problems. ECSHB proved to be more effective in providing security and privacy for data in the Cloud, when compared with other schemes. They introduced new intrusion detection system (IDS) model with deep learning features, such as, Morlet Wavelet Kernel Function for better threat detection [18]. An MLSTM classifier was developed to detect the attacks in IoT Cloud with the Jarratt-Butterfly Optimization Algorithm (JBOA) feature selection approach to enhance the classification accuracy. The model was evaluated on the detection of BoT-IoT and various performance measures with popular benchmark methods. Simulation results showed that the proposed IDS model delivered better performance relative to some state of the art approaches in detection of different cyberattacks in IoT Cloud scenarios, and therefore verified its efficiency and superiority. Kumar and Kumar [19] proposed a secure encryption system with ECC and AES to ensure the security of data stored in the cloud, when storing user's sensitive information anonymously. The approach outperformed the state of the art and achieved higher quality.

3 | Proposed Methodology

This section summarizes the sophisticated encryption methods used in this work, namely RSA, Chacha20 and SHA3 which are further upon in the following section and represented in *Fig. 2*.

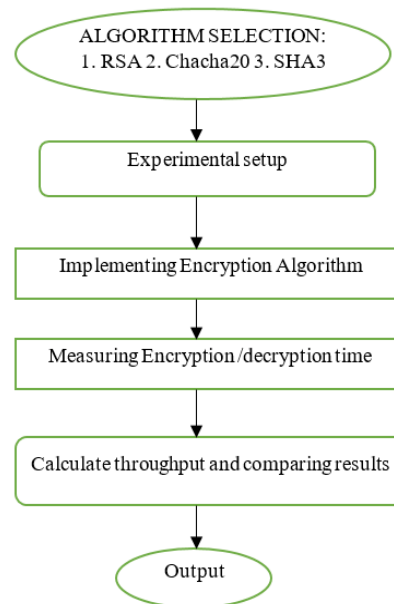


Fig. 2. Data security methodology flowchart using encryption methods.

3.1 | Rivest Shamir Adleman

Rivest, Shamir, and Adleman created one of the most popular public key algorithms, known as Rivest Shamir Adleman (RSA). It is an asymmetric method used for encrypting and decrypting data. We call it asymmetric because the two sides swap the public key for encrypting messages but conceal the private key. How RSA will work in the cloud: Cloud computing uses RSA algorithm to secure data. Data is encrypted with RSA algorithm for security. Data is secured so only authorized people can access it. Information is saved in the cloud after encryption. So, cloud provider can be contacted as needed [20]. Cloud provider verifies user and delivers data. Due to its Block Cypher, RSA maps every message to an integer. The public key can be shared with everyone, but only the cloud provider and the user have the private key needed to decrypt the data. Information encrypted with the public key can be read only by the original owner, because they are the only ones with the private key. When using the cloud, encryption is done by the provider, and then consumers decrypt it. Information locked using a public key can only be unlocked with its matching private key. The RSA process is typically carried out in three stages:

3.1.1 | Key generation

A process of key creation must precede the encryption of any data. This is handled by the user and the supplier of cloud services.

Table 1. Key generation algorithm.

Row	Steps
1	Pick two distinct prime numbers, a and b . To improve security, both primes should be chosen randomly and should have roughly the same bit size.
2	Calculate $n = a * b$.
3	Calculate Euler's totient function, $\phi(n) = (a - 1) * (b - 1)$.
4	Indicated an integer e , such that $1 < e < \phi(n)$ and greatest common divisor of $e, \phi(n)$ is 1. Now e is released as Public-Key exponent.
5	Now regulate d as follows: $d = eP - 1 \text{ P(mod } \phi(n))$ i.e., d is multiply inverse of $e \text{ mod } \phi(n)$.
6	d is kept as Private-Key component, so that $d * e = 1 \text{ mod } \phi(n)$.
7	The Public-Key consists of modulus n and the public exponent e i.e., (e, n) .
8	The Private-Key consists of modulus n and the private exponent d , which must be kept secret i.e., (d, n) .

3.1.2 | Encryption algorithm

Encryption means turning plain, readable information into a protected form that others cannot understand.

Table 2. Key encryption algorithm.

Row	Steps
1	The user must either get or submit a Public-Key (n, e) after store data using the CSP.
2	The user's data is now assigned an integer employing a padding technique, a reversible procedure that has been agreed upon.
3	$C = mP^e \pmod{n}$ is the cypher text (data) that is produced after encryption.
4	This deciphered information will now be stored by the CSP.

3.1.3 | Decryption algorithm

A decryption process serves to convert the encrypted data back to its clear form.

Table 3. Key Decryption algorithm.

Row	Steps
1	Data requests are made by cloud users to CSP.
2	The encrypted data is released by the cloud provider upon user identification verification (C).
3	A user then computes $m = C P^d \pmod{n}$ to decrypt a data stored in the cloud.
4	After getting m , retrieving the original data is as simple as reversing the padding scheme.

3.2 | ChaCha20

The model incorporates and extends the properties of the ChaCha20 cipher, a fast, secure and ARX-based (Add–Rotate–XOR) stream cipher construction for high assurance message confidentiality, integrity and resistance against most timing attack vectors. ChaCha20 works well on small messages and low-information environments such as IoT deployments, making it a desirable option for real-time communications, bits or bytes [21]. The methodology is a general one, consisting of a series of cryptographic steps: keys and state seeding, ChaCha20 core transform invocation, keystream application encryption and decryption.

3.2.1 | Key and nonce setup

ChaCha20 encryption is initiated with the generation and selection of the cryptographic parameters.

- I. 256-bit secret key (32 bytes).
- II. Unique nonce.
- III. 32-bit block counter.

The 256-bit key is held privately by the communicating entities and serves as primary component for the keystream generation. In this model, it includes an 8-byte (64-bit) nonce as well as you can see in your example but current standard ChaCha20 uses a 12-byte (96-bit) nonce. Regardless, the nonce must be different for every encryption session otherwise security is entirely broken under the same key. The block counter begins with a value of 0, and causes each portion of the keystream to be generated as an independent 64-byte block. These three variables key, nonce and block counter are the ones that lead all further encryption operations.

3.2.2 | Initialization of the ChaCha state

In the setup phase, ChaCha20 constructs a 512-bit (64 byte) internal state by arranging sixteen "words" of 32-bits each. This state is known as the ChaCha state. This state is then filled according to a strict ordering:

- I. Constants (16 bytes): in their initial four 32-bit words, values composed of a constant, by default the ASCII text “expand 32-byte k” which tells the algorithm how to treat the key data and provide es defence against related-key attacks.
- II. Key (32 bytes): the next eight 32-bit words hold a not necessarily secret 256-bit key, which provides the primary source of randomness to the state.
- III. Block Counter (4bytes): the counter, initially zero, takes the next word. It increases after each 64-byte block of keystream output, thus avoiding the same key repeatedly in the keystream.
- IV. Nonce (8 bytes): the nonce forms the least-significant words of the state. As the nonce may never repeat under the same key, it introduces an extra dimension of unpredictability. The outcome is a recognizable 64-byte ChaCha state which will be used as the starting sequence for the ChaCha20 core function.

3.2.3 | ChaCha20 core function (20-round transformation)

The heart of the ChaCha20 algorithm lies in its iterative transformation, known as the ChaCha20 core function. This function does 20 rounds of cryptographic mixing, which are set up as 10 double rounds. In each round, structured ARX operations are used on the ChaCha state.

Quarter-round operations: the key component of the ChaCha20 cipher is an iterative function called the ChaCha20 dround function. This procedure consists of 20 rounds of cryptographic mixing operations, frozen into 10 double-rounds. The ChaCha state undergoes organized ARX operations every round: modular addition, bitwise XOR, and fixed bit rotations. These operations introduce nonlinearity and diffusion, ensuring that small changes in the key or nonce propagate throughout the state.

Row and column mixing: quarter rounds are peppered throughout that cant each round to: columns of the 4×4 state matrix and diagonals of the state matrix.

This back and forth takes the influence of all words of a state over each other within a few rounds and prohibits differential and linear cryptanalysis. Once 20 rounds have been performed, the algorithm outputs an effectively randomly transformed state.

3.2.4 | Keystream generation

After 20 round transformation, ChaCha20 core state is added modulo 2 to the 32 to the original state (addition is performed word by word). This method is called feedforward and provides both nonlinearity and cryptographic security. The resulting 64-byte message block is output as the keystream block. A plaintext or ciphertext keystream is equivalent to a pseudo-random pad for encryption and decryption. Long messages If the message is long, then for each block of keystream to be generated the value of the block counter should be incremented and transformation re-applied plugging this new counter back.

3.2.5 | Encryption process

ChaCha20 is a stream cipher awarded for performance and simplicity: it encrypts data with the XOR-based mechanism.

- I. The plaintext is divided in 64 bytes blocks.
- II. 64 bytes of keystream is generated for each block.
- III. This keystream is XORed with each byte of the plaintext block to form the ciphertext block;

$$C_i = P_i \oplus K_i, \quad (1)$$

when the last block contains fewer than 64bytes of plaintext *Eq. (1)*, only the first required number of bytes from keystream is XORed. This is what allows ChaCha20 to safely handle messages of arbitrary length without any need for padding.

3.2.6 | Decryption process

As XOR is self-inverse, in ChaCha20 decryption is symmetric and employs the same keystream;

$$P_i = C_i \oplus K_i. \quad (2)$$

If it has the same key, the same nonce and the same block counter, then Eq. (2) What happens is that when you in this case Yields exactly the Correct keystream that would've bytes. Decryption of the ciphertext using this keystream perfectly restores the original plaintext. Crucially, it should not be possible to re-use nonces and counters since re-using a keystream under the same key reveals relationships between plaintexts and causes complete confidentiality loss.

Table 4. Algorithm steps for chacha 20.

Chacha20 Algorithm/ steps
Input:
• 256-bit key K • 96-bit nonce N] • 32-bit block counter C • Constant string: “expand 32-byte k”
Output:
• 64-byte keystream block (used for encryption/decryption)
Step -by- Step Procedure
Step 1. Initialize Internal State
Form a 4×4 state matrix of 32- bit words:
1. Words 0-3: Algorithm constants 2. Words 4-11: 256-bit Key 3. Word 12: Block Counter 4. Words 13-15: 96-bit nonce
This creates the 16-word ChaCha20 state.
Step 2. Make a Working Copy of the State
• Create a second 16-word array (working state) identical to the initial state. • This ensures the original state remains unchanged for the final addition step.
Step 3. Perform 20 Rounds of Transformation
ChaCha20 uses 20 rounds = 10 double rounds.
Each double round consists of:
Round A – Column Round
Apply the Quarter-Round function to the words in the following order:
(0, 4, 8, 12)
(1, 5, 9, 13)
(2, 6, 10, 14)
(3, 7, 11, 15)
Round B – Diagonal Round
Apply Quarter-Round to:
(0, 5, 10, 15)
(1, 6, 11, 12)
(2, 7, 8, 13)
(3, 4, 9, 14)
These steps ensure complete diffusion across the state.
Step 4. Quarter-Round Function
For four 32-bit words (a, b, c, d):
1. $a = a + b; d = (d \oplus a) \lll 16$ 2. $c = c + d; b = (b \oplus c) \lll 12$ 3. $a = a + b; d = (d \oplus a) \lll 8$

$$4. \quad c = c + d; b = (b \oplus c) \lll 7$$

Where “ $\lll$ ” is left rotation.

Each operation adds nonlinearity and spreads bits effectively.

Step 5. Add Working State to Original State

After the 20 rounds:

Add each word of the transformed working state to the corresponding word of the initial state modulo 2^{32} .

This “feedforward” step ensures cryptographic security.

Step 6. Produce Keystream Block

Convert the 16 resulting 32-bit words into a continuous 64-byte keystream block.

This output is used to encrypt or decrypt data.

Step 7. Encrypt / Decrypt

ChaCha20 uses XOR for both encryption and decryption:

$$\text{Ciphertext} = \text{Plaintext} \oplus \text{Keystream}$$

$$\text{Plaintext} = \text{Ciphertext} \oplus \text{Keystream}$$

Because XOR is symmetric, the same keystream block works for both operations.

Step 8. Increment Counter for Next Block

To generate the next 64-byte keystream block:

Increment the block counter CCC.

Repeat Steps 1–7.

This guarantees that each block is unique within a session.

3.1| Secure Hash Algorithm 3

The SHA-3 cryptographic hash is an algorithm that takes some input data and reduces it to a fixed size. SHAKE-3 also allows users to select the length of the hash values they want, as long or short as required. SHA-3 is an important component of existing cryptographic systems that guarantee data are secure and authentic. It is used in a number of cryptographic protocols and applications due its speed, efficiency, and security. SHA-3 The SHA-3 is an important cryptographic algorithm that can provide data security and integrity for digital transactions [22]. SHA-0, SHA-1 and SHA-2 are also some of the latest cryptographic hash functions that has been reported to be weak against attacks [10]. Despite all these weaknesses within a hash function still many Cryptographic protocols are built on such algorithms. The NIST has formally characterized SHA-3 as a separate member of the Keccak family. It covers the four SHA-3 proposals as well as two extendable output functions (XOFs), SHAKE128 and SHAKE256. The SHA-3 routines have a preset output length whereas the two SHAKE versions enable you to obtain data in different lengths, as required. So, SHA-3 is a JPRF candidate in the sense of producing pseudo-random bits [23]. Fig. 3 indicates that all the SHA-3 functions operate under the same general framework, known as the sponge construction. Such a framework is indeed very flexible and allows it to produce hash values of variable length, which means it's suitable for quite large set of applications.

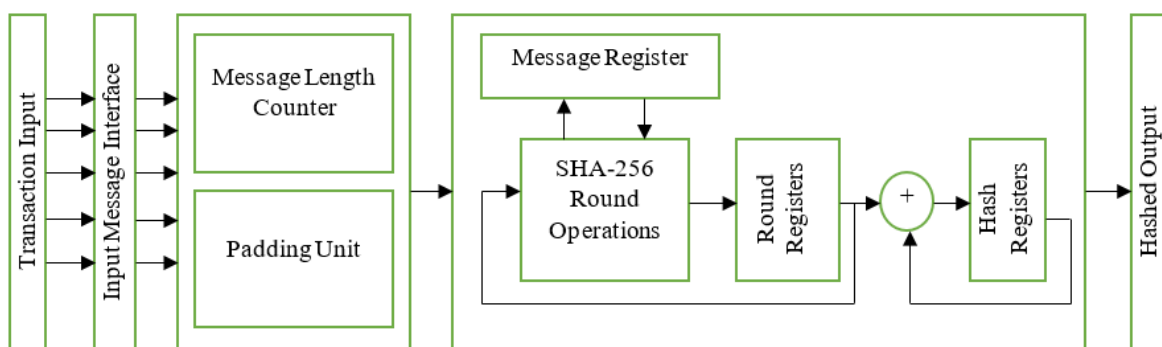


Fig. 3. Process of SHA3.

4 | Result and discussion

This section discusses how good the newly designed integrated crypto framework is. It is based on RSA for secure key exchange, ChaCha20 for symmetric encryption, and SHA-3 to verify integrity in cloud engineering systems. The experiments concentrated on three main criteria, including: the encryption/decryption time and throughput, and hashing performance, as well as benchmarking with AES-256 with SHA-256 traditional method and a typical RSA-AES hybrid approach. Table 1 illustrates how long it takes to encrypt and decode a 50 MB dataset, as well as the throughput figures for each of the three settings.

4.1 | Encryption Time

Encryption Time is the overall time it takes for the encryption algorithm to change plaintext into ciphertext. This includes the time needed for key setup, block processing, keystream generation (in the case of ChaCha20), and performing the cryptographic transformations.

$$\text{Encryption Time(ET)} = T_{\text{end}} - T_{\text{start}}, \quad (3)$$

where, the Eq. (3), T_{start} = Time at which encryption begins, T_{end} = Time at which encryption ends

4.2 | Decryption Time (DT)

Decryption Time tells you how long it takes to turn ciphertext back into plaintext applying the similar cryptographic keys. For stream ciphers such as ChaCha20, decryption time is nearly identical to encryption time because both operations use XOR with the keystream.

$$\text{Decryption Time (DT)} = T_{\text{end}} - T_{\text{start}}, \quad (4)$$

where, the Eq. (4) T_{start} = Time at which decryption begins, T_{end} = Time at which decryption ends.

4.3 | Throughput (TP)

During encryption or decryption, throughput is the amount of data that is processed in a certain length of time. It is a key performance indicator that reflects how fast the cryptographic system can handle large datasets.

$$\text{Throughput(TP)} = \frac{\text{Total Data Size (MB)}}{\text{Processing Time (Seconds)}}, \quad (5)$$

Or, in milliseconds:

$$\text{Throughput (TP)} = \frac{\text{DataSize (MB)}}{\text{Processing Time (ms)}} \times 1000, \quad (6)$$

where, the equation (5&6), Data Size = Size of plaintext or ciphertext processed,

Processing Time = encryption or decryption time.

4.4 | Data Security Level

Data Security Level (DSL) refers to how secure a cryptographic or security system is designed in protecting data along other factors such as the authentication strength, availability, integrity and confidentiality. These requirements highlight the importance of secure clouds for each:

- I. Confidentiality measures how effectively the system prevents unauthorized access to sensitive data
- II. Integrity Protection evaluates how well the system protects data from unauthorized modification or tampering.
- III. Availability: The system should provide continuous service in terms of accessing the data without degradation or denial of service.

IV. Authentication Strength: Measures how strong the identity confirmation is and its resistance capabilities from impersonation attacks.

$$DSL(\%) = \frac{\text{Confidentiality} + \text{Integrity} + \text{Availability} + \text{Authentication}}{4} \quad (7)$$

Table 1. Performance comparison of encryption schemes for 50 MB data.

Configuration	Encryption Time (ms)	Decryption Time (ms)	Throughput (MB/s)
AES-256 + SHA-256	420	400	119
RSA-2048 + AES-256 + SHA-256	460	430	109
RSA-2048 + ChaCha20 + SHA3-256	310	305	161

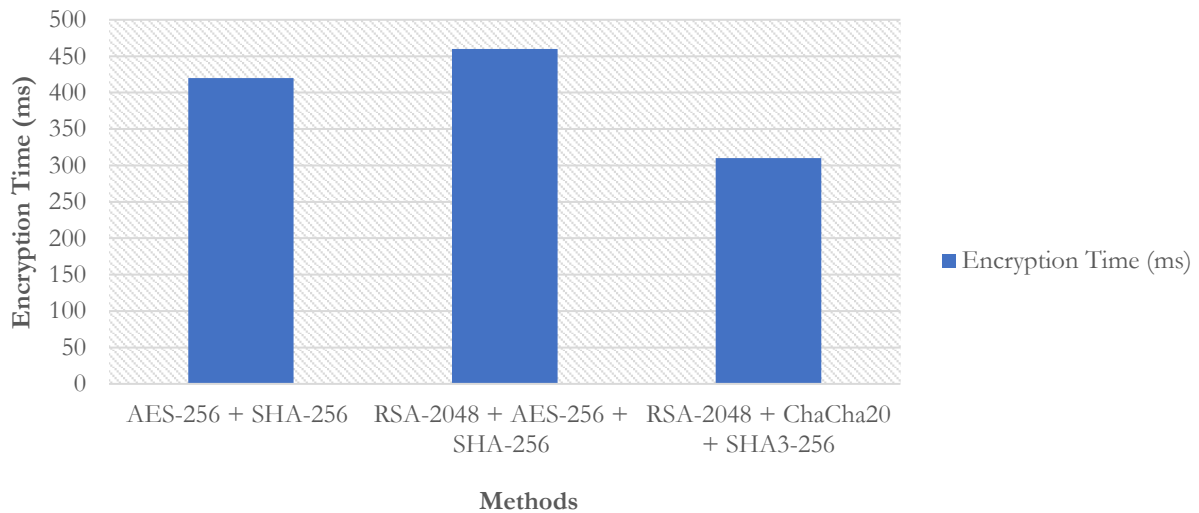


Fig. 4. Comparison of encryption time for different cryptographic methods.

Encryption times (in milliseconds) are depicted in *Fig. 4* for three different models of cryptosystem: RSA-2048 + AES-256 + SHA-256, AES-256 + SHA-256 and the proposed RSA2048 + ChaCha20+SHA3-256. The RSA-2048 + AES-256 + SHA-256 algorithm has the longest encryption time (around 460 ms), followed by AES-256 + SHA-256 (about 420 ms). For all configurations, the cryptographic answer RSA-2048 + ChaCha20 + SHA3-256 has the lowest encryption time (around 310 ms). And the researches show that, (x) ChaCha20 combined with SHA-3 and RSA key exchange provides faster and more efficient encryption/decryption than traditional AES.

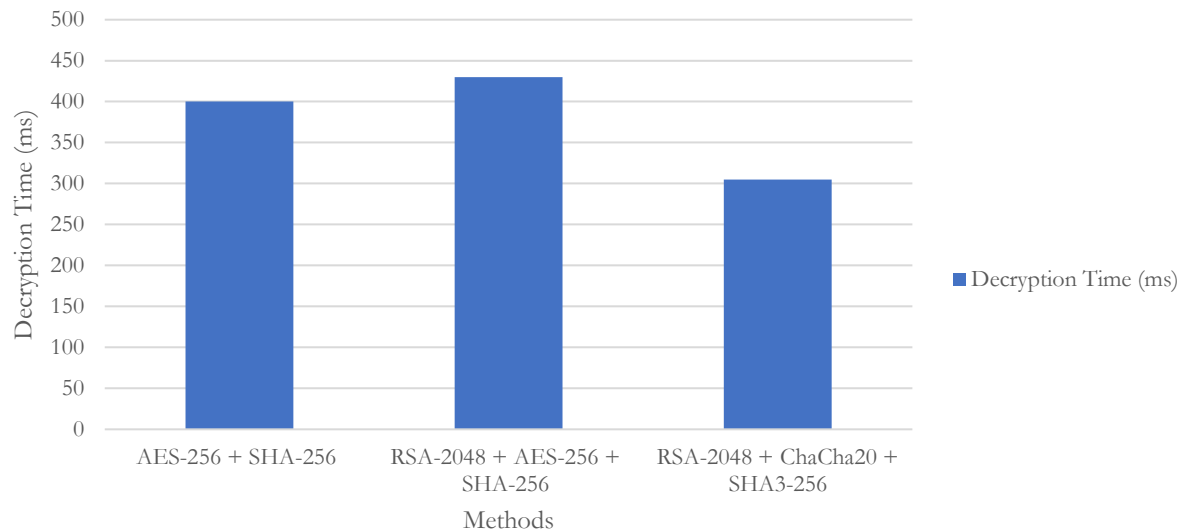


Fig. 5. Comparison of decryption time for different cryptographic methods.

The decryption times in milliseconds for three cryptographic configurations, namely, RSA-2048 + AES-256 + SHA-256, AES-256 + SHA-256 and the proposed RSA-2048+ChaCha20+SHA3-256 are shown Figure 5. The RSA-AES hybrid method has the longest decryption time at (approximately 430 ms), while AES-256 + SHA-256 follows closely at around 400 ms. The preferred approach (RSA-ChaCha20-SHA3) enhances the system in that it has by far, the shortest decryption time of approximately 305 ms demonstrating its great efficiency. Overall, the graph evidences the much better decryption results of the proposed encryption scheme compared to that of AES based techniques.

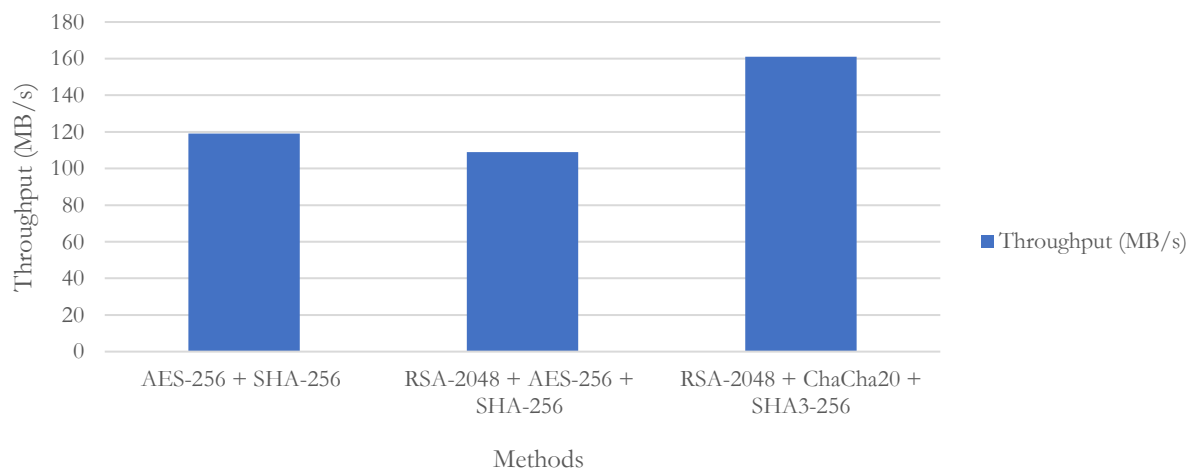
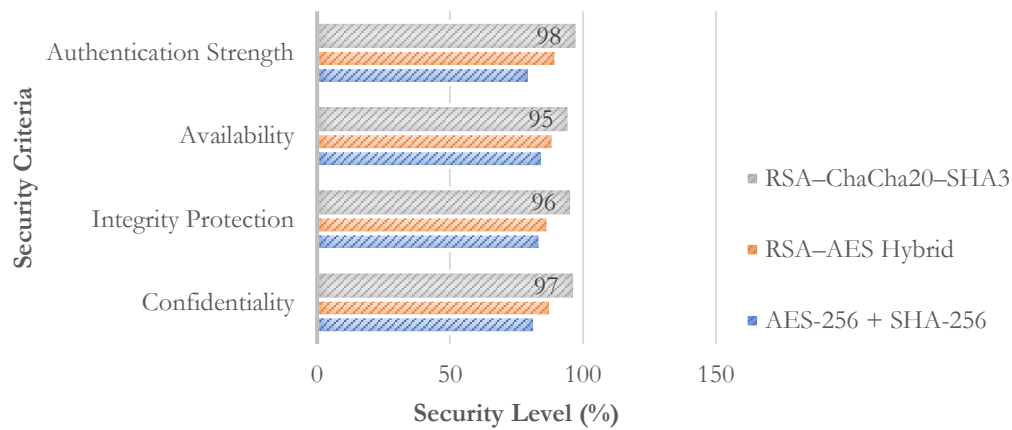


Fig. 6. Throughput performance comparison for different cryptographic methods.

Fig. 6 displays the throughput (in MB/s) for three different cryptographic setups: RSA-2048 + AES-256 + SHA-256, AES-256 + SHA-256 and the recommended RSA-2048 + ChaCha20 + SHA3-256 algorithm. The recommended solution has the maximum throughput around 160 MB/s, followed by AES-256 + SHA-256 at 118 MB/s, and RSA-AES hybrid with a maximum of approximately 110 MB/s. The test results demonstrate that compared to other encryption standards considered, the proposed RSA-ChaCha20-SHA3 system produces faster throughput which is preferable for engineering work performed in real-time environments on the cloud.

Table 3. Data security level comparison for different encryption frameworks.

Security Criteria (0–100%)	AES-256 + SHA-256	RSA–AES Hybrid	RSA–ChaCha20–SHA3
Confidentiality	82	88	97
Integrity Protection	84	87	96
Availability	85	89	95
Authentication Strength	80	90	98
Overall Security Score	82.75%	88.5%	96.5%

**Fig. 7. Security level comparison across different cryptographic frameworks.**

AES-256 + SHA-256, RSA-AES Hybrid and proposed RSA-ChaCha20-SHA3-based on four critical to security attributes; Confidentiality, Integrity protection, Availability and Authentication strength. In all four measures, RSA-ChaCha20-SHA3 is strongest; with 97% in confidentiality and 96%, 95%, and the maximum strength for integrity protection, availability and authentication. The RSA-AES Hybrid pattern is second and AES-256 + SHA-256 appears again at the bottom in each category. This demonstrates that the designed pattern offers the most secure environment cloud-based engineering solutions can have.

5 | Conclusion

This paper satisfied the demand of enhancing security in cloud-based engineering systems by analysing serious security threats and performance limitations of current cryptography methods. To address these challenges, this paper presented an all-in-one security protocol that takes advantages of 1) RSA for secure key management, 2) ChaCha20 for fast symmetric encryption, and 3) SHA-3 for robust integrity verification. The proposed architecture enhanced cloud security by the use of a performant key exchange protocol in combination with lightweight ARX based encryption and collision resistant hashing to achieve secure confidentiality, integrity, availability and robust authentication. The proposed RSA-ChaCha20-SHA3 framework is better than traditional AES-based methodologies in both encryption-decryption speed, throughput, and security trade-off as the experimental results demonstrate. These findings suggest that the presented approach performs well for engineering processes in active time and large cloud deployments. In addition, we plan to consider 1.

Data Availability

All the data are available in this paper.

Funding

This research received no specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest

The authors declare no conflict of interest.

Consent for Publication

The authors consent to the publication of this manuscript.

Ethics Approval and Consent to Participate

The authors confirm that this research did not involve human participants or animal subjects.

References

- [1] Soveizi, N., Turkmen, F., & Karastoyanova, D. (2023). Security and privacy concerns in cloud-based scientific and business workflows: A systematic review. *Future generation computer systems*, 148, 184–200. <https://doi.org/10.1016/j.future.2023.05.015>
- [2] Omer, M. A., Yazdeen, A. A., Malallah, H. S., & Abdulrahman, L. M. (2022). A survey on cloud security: concepts, types, limitations, and challenges. *Journal of applied science and technology trends*, 3(02), 101–111. <https://doi.org/10.38094/jastt301137>
- [3] Rana, M., Mamun, Q., & Islam, R. (2022). Lightweight cryptography in IoT networks: A survey. *Future generation computer systems*, 129, 77–89. <https://doi.org/10.1016/j.future.2021.11.011>
- [4] Pothireddy, S., Peddisetty, N., Yellamma, P., Botta, G., & Gottipati, K. N. (2024). Data security in cloud environment by using hybrid encryption technique: A comprehensive study on enhancing confidentiality and reliability. *International journal of intelligent engineering & systems*, 17(2), 159–170. <https://doi.org/10.22266/ijies2024.0430.14>
- [5] Anjana, A., & Singh, A. (2024). An enhanced three layer cryptographic algorithm for cloud information security. *International journal of intelligent systems and applications in engineering (IJISAE)*, 12(17s), 615–627. <https://ijisae.org/index.php/IJISAE/article/view/4927/5613>
- [6] Amaithi Rajan, A., & V, V. (2024). Systematic survey: secure and privacy-preserving big data analytics in cloud. *Journal of computer information systems*, 64(1), 136–156. <https://doi.org/10.1080/08874417.2023.2176946>
- [7] Abrera, J., & others. (2024). Data privacy and security in cloud computing: a comprehensive review. *Journal of computer science and information technology*, 1(1), 1–9. <https://doi.org/10.61424/jcsit.v1i1.58>
- [8] Begum, T., & Hossain, M. E. (2021). Enhancing the security of cloud computing by building hybrid cryptography algorithms. *International journal of computer applications*, 975, 8887. <https://www.academia.edu/download/115894406/ijca2021921849.pdf>
- [9] Abroshan, H. (2021). A hybrid encryption solution to improve cloud computing security using symmetric and asymmetric cryptography algorithms. *International journal of advanced computer science and applications*, 12(6). <https://doi.org/10.14569/IJACSA.2021.0120604>
- [10] Virushabadoss, S., & Anithaashri, T. P. (2022). Enhancing data security in mobile cloud using novel key generation. *Procedia computer science*, 215, 567–576. <https://doi.org/10.1016/j.procs.2022.12.059>
- [11] Ramachandran, D., Albathan, M., Hussain, A., & Abbas, Q. (2023). Enhancing cloud-based security: a novel approach for efficient cyber-threat detection using GSCSO-IHNN model. *Systems*, 11(10), 518. <https://doi.org/10.3390/systems11100518>
- [12] Rao, B. R., & Sujatha, B. (2023). A hybrid elliptic curve cryptography (HECC) technique for fast encryption of data for public cloud security. *Measurement: sensors*, 29, 100870. <https://doi.org/10.1016/j.measen.2023.100870>
- [13] Singh, J., Reddy, A. M., Bande, V., Lakshmanarao, A., Rao, G. S., & Samunnisa, K. (2023). Enhancing Cloud Data Privacy with a Scalable Hybrid Approach: HE-DPSMC. *Journal of electrical systems*, 19(4). <https://doi.org/10.52783/jes.643>

- [14] Reddy, S. M., Sana, M., Sushma, M., Sri, M. D., & others. (2024). Enhanced data privacy and security in cloud storage: A robust authentication scheme for Cyber-Physical-Social systems. <https://hdl.handle.net/10481/103751>
- [15] Hossain, M. A., & Al Hasan, M. A. (2022). Improving cloud data security through hybrid verification technique based on biometrics and encryption system. *International journal of computers and applications*, 44(5), 455–464. <https://doi.org/10.1080/1206212X.2020.1809177>
- [16] He, Q., & He, H. (2020). A novel method to enhance sustainable systems security in cloud computing based on the combination of encryption and data mining. *Sustainability*, 13(1), 101. <https://doi.org/10.3390/su13010101>
- [17] Verma, G., & Kanrar, S. (2022). A novel model to enhance the data security in cloud environment. *Multiagent and grid systems*, 18(1), 45–63. <https://doi.org/10.3233/MGS-220361>
- [18] Silambarasan, E., Suryawanshi, R., & Reshma, S. (2024). Enhanced cloud security: a novel intrusion detection system using ARSO algorithm and Bi-LSTM classifier. *International journal of information technology*, 16(6), 3837–3845. <https://doi.org/10.1007/s41870-024-01887-x>
- [19] Kumar, S., & Kumar, D. (2023). Securing of cloud storage data using hybrid AES-ECC cryptographic approach. *Journal of mobile multimedia*, 19(2), 363–388. <https://doi.org/10.13052/jmm1550-4646.1921>
- [20] Li, X., Zhu, J., Meng, F., Wang, L., & Qin, J. (2024). Security and privacy protection using rivest shamir adleman with bit insertion technique based on big data in cloud. In *2024 third international conference on distributed computing and electrical circuits and electronics (ICDCECE)* (pp. 1-4). <https://doi.org/10.1109/ICDCECE60827.2024.10549358>
- [21] Muhammed, R. K., Rashid, Z. N., & Saydah, 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. <https://doi.org/10.24017/science.2025.1.5>
- [22] Soni, A., & Sahay, S. K. (2024). SEASHA3: Secure and efficient encryption for the iot data by replacing subkeys of AES with SHA3. In *2024 IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 1998-2003). <https://doi.org/10.1109/SMC54092.2024.10831459>
- [23] Ghosh, S., Verma, S. K., Ghosh, U., & Al-Numay, M. (2023). Improved End-to-End data security approach for cloud computing. *Sustainability*, 15(22), 16010. <https://doi.org/10.3390/su152216010>