Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 2-A, pp. 445–452
DOI : 10.47974/JDMSC-2476

Enhancing RSA algorithm security through advanced combinatorial n! permutations in cryptographic protocols

Amol Sapatnekar [†]
Symbiosis Centre for Advanced Legal Studies and Research (SCALSAR)
Symbiosis Law School
Symbiosis International (Deemed University)
Pune 411014
Maharashtra
India

Deepika Sharma*
Department of Computer Sciences
Noida International University
Greater Noida 203201
Uttar Pradesh
India

Rita Thakare [§]
Department of Electronics and Telecommunication
Dr. D. Y. Patil Institute of Technology
Pimpri
Pune 411018
Maharashtra
India

Anita Dombale [‡]
Department of Computer Science & Engineering
Vishwakarma Institute of Technology
Pune 411037
Maharashtra
India

[†] *E-mail:* amol.sapatnekar@symlaw.ac.in
* *E-mail:* deepika.sharma@niu.edu.in (Corresponding Author)
[§] *E-mail:* rita.thakare@dypvp.edu.in
[‡] *E-mail:* anita.dombale@vit.edu

© Taru Publications

Vikas Nandgaonkar [@]
Indira College of Engineering and Management
Pune 410506
Maharashtra
India

S. Radhakrishnan [#]
Department of Artificial Intelligence
KKR and KSR Institute of Technology and Sciences
Guntur 522017
Andhra Pradesh
India

Abstract

Quantum computing and improved cryptanalysis have rendered the RSA algorithm less safe, but it remains a vital aspect of public-key encryption. Newly, this work enhances the RSA algorithm's secure protocol by adding sophisticated combinatorial n! Permutations. To create a vast and surprising keyspace, factorial-based permutations of key components like ciphertext and plaintext are used. These n! variants are carefully exploited for key creation, encryption, and decoding. They make illegal file decoding tougher without changing the process. This reduces brute-force and factorisation risks. The design additionally employs these padding and message encoding variants to improve spread and confusion, following Shannon's cryptography principles. The entropy and statistical unpredictability of protected data have improved, and theoretical study supports the assumption that the number of key configurations may expand exponentially without affecting data processing speed.

Subject Classification: *11T71.*

Keywords: *RSA algorithm, Factorial-based permutation, Public-key security, Combinatorial cryptography.*

1. Introduction

The RSA technique is one of the maximum popular sorts of public-key cryptography, that's a key part of preserving digital interactions secure. Given that the beginning, RSA has been recognised for its mathematical beauty and use of the computational trouble of integer factorisation [1][2]. RSA is on the whole safe because it's challenging to element big composite numbers, which might be typically the sum of 2 huge primes [3]. There are massive issues about how long-lasting well known RSA implementations may be because quantum computing is getting better and factorisation methods like the overall variety discipline Sieve are becoming higher [4][5]. Responding to these difficulties,

[@] *E-mail:* **vikas.nandgaonkar@indiraicem.ac.in**
[#] *E-mail:* **dr.srkrishnan@gmail.com**

latest cryptography research has attempted to improve established strategies by using adding extra degrees of complexity without reducing their usability. Combinatorial arithmetic, especially n! Factorial diversifications, is probably used to substantially increase the range of viable encryption keys. This is a hopeful course [6]. The factorial characteristic, which suggests how many ways n extraordinary parts can be put together, grows at a fantastic-exponential fee, giving us a huge wide variety of approaches to change things [7]. Adding these n! versions to the stairs of making an RSA key, encrypting it, and decrypting it provides another non-linear trade that makes it more difficult to break using brute pressure or sample-based attacks. Including extra complicated combinations no longer solely makes ciphertext much less predictable, but it additionally fits with simple cryptography thoughts like uncertainty and unfold [8]. Claude Shannon came up with those guidelines to cover the relationship between plaintext, ciphertext, and key [9]. This way, it would be impossible for a person to decode data without permission. This development thinking keeps RSA's uneven nature and can be used barring changing its simple modular math form.

Incorporating n! permutations into RSA leverages the vast combinatorial possibilities of factorial arrangements to exponentially expand the keyspace. This enhancement introduces an additional non-linear transformation layer, making brute-force and pattern-based attacks significantly more difficult. n! permutations provide a factorial-scale expansion of possible key configurations, drastically increasing RSA's cryptographic strength. By introducing such combinatorial diversity, the encryption process becomes far less predictable, effectively countering brute-force and statistical attacks.

Because of the need to protect current encryption systems from new threats while still using well-known mathematical structures, this method was created. Using factorial-based combinatorial logic adds a scalable and flexible improvement that can be tweaked to meet the security needs of different uses, such as protecting personal healthcare data, encrypting financial data, and ensuring secure military communications [10]. As digital environments change, one important area of study that will stay the same is how to make cryptographic methods stronger using techniques that are both mathematically sound and logically possible.

2. Key Generation Using Traditional RSA Approach:

The process starts with picking two big prime numbers, which are shown as p and q. The modulus n=p×q is found using these prime numbers. It is an important part of both encrypting and decrypting. To find the totient function (n), use the following formula:

$$\phi(n) = (m - 1)(q - 1) \quad (1)$$

Then, an encryption exponent e is chosen so that $1<e<\phi(n)$ and $\gcd(e, \phi(n)) = 1$, making sure it is the same as $\phi(n)$(n). To find the decryption exponent d, it's necessary to solve the following equation:

$$e.d \equiv 1 \left(mod\ \phi\ (n)\right) \tag{2}$$

This means that d is the modular inverse of e modulo (n). The pair (e, n) makes up the public key. The pair (d, n) makes up the private key. These factors support the RSA algorithm's asymmetric structure and allow safe communication by using modular exponentiation during encryption and decoding. System architecture process block diagram illustrate in figure 1.

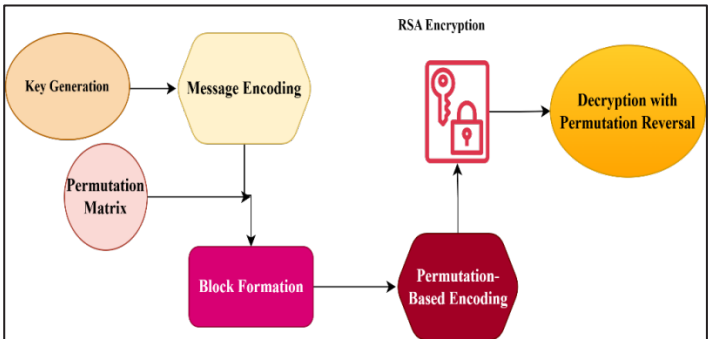


Figure 1

System architecture to enhancing RSA algorithm security

3. Generation of n! Permutation Matrix

A permutation matrix $P_p \in S_k$ Every message block of length k has a unique key $P_k \in S_k$. The symmetric group with all k possible combinations is shown by the set S_k. A safe pseudo-random number generator with a cryptography hash function is used to choose the permutation:

$$Seed = H(e \mid\mid n \mid\mid t) \tag{3}$$

The eq. (1) has H which is a safe hash method (like SHA-256) and t is a timestamp or session nonce. Because of this seed, a random permutation $\pi \in S_k$ k is picked. The appropriate permutation matrix MkP k is made in a way that:

$$(P_k)_{i,j} = \begin{cases} 1 & if\ j = \pi\ (i) \\ 0 & otherwise \end{cases}$$

This grid lets key-specific data drive fixed but unexpected changes, which makes recovery safe and consistent during decryption. The matrix has the factorial rearrangement qualities, which make sure that each message block can be put together in one of the k! possible ways. This makes the cryptographic space much bigger.

4. Permutation-Based Encoding

The permutation matrix P_k is then applied to each message block Mi, where Mi is a number in the range Znk. The following describes the permutation transformation:

$$M_i' = P_k \cdot M_i \tag{4}$$

The eq. (1) have P_k which is a k x k matrix and Mi is a column vector. This matrix multiplication rearranges the message block's parts according to the permutation π. This makes sure that the decoded block M_i' keeps all of the original data but is jumbled. The total number of unique results is given by k!, which makes things much less predictable. Because permutation matrices are orthogonal, the process can be turned around:

$$P_k^{-1} = P_k^T \tag{5}$$

This feature makes sure that decryption can correctly put things back in the way it was before. By adding this variation before encryption, the structure adds to the confusion and makes it harder to use frequency analysis and pattern recognition, which could help with the analysis.

5. RSA Encryption with Permuted Message

The permuted block Mi' M i ′ is encrypted with normal RSA encryption, which is described by

$$C_i = (M_i')^e mod\ n \tag{6}$$

In the permuted block shown in the eq. (1), the public exponent e with modulus n is used to raise each value by one. Prior randomisation makes the input more varied and makes the ciphertext very hard to attack with either chosen-plaintext or known-plaintext. Randomness added by k! permutations makes sure that even if two similar blocks of data are used, they will produce different ciphertexts. The discrete logarithm and factorisation problems are both hard to solve, which supports the mathematical strength of this stage. These problems are also at the heart of RSA's encryption toughness. The encrypted result is a version of the original block that has been jumbled and multiplied, which makes it safer through multiple changes.

6. Decryption with Permutation Reversal:

The RSA decryption method is used to start the decryption phase:

$$M_i' = C_i^d\ mod\ n \tag{7}$$

This gets back the permuted message block M_i'. Then, the inverse permutation matrix P_k^{-1} is used to put together the original message:

$$M_i = P_k^{-1} \cdot M_i' \tag{8}$$

Because permutation matrices are orthogonal, $P_k^{-1} = P_k^T$. This means that the process of reordering can be done exactly backwards. This makes sure that each message block goes back to the order it was in before it was permuted. The two-step decryption process, modular exponentiation followed by permutation inversion keeps the data's coherence and purity. The method follows the reversibility principle, which is important for symmetric recovery processes, and it only needs the hash-seeded permutation key as extra information. This step finishes the change process and makes sure that the original information can be retrieved safely and without any loss.

7. Performance Analysis

The table 1 shows how well four well-known encryption methods perform: Classical RSA, Enhanced RSA with combinatorial (n!) permutations, ECC (Elliptic Curve Cryptography), and AES (Advanced Encryption Standard). The key size is 2048 bits for both types of RSA. This is followed by 256 bits for ECC and 128 bits for AES. This shows how small current symmetric and elliptic curve systems are. The encryption and decoding times show that AES works the fastest (0.09 ms and 0.11 ms), while Enhanced RSA is the slowest (2.89 ms and 5.07 ms), which is because it needs more work to do because it has more permutation layers.

Table 1

Performance Comparison of traditional encryption strategies with enhanced RSA+n!

Parameter	Classical RSA	Enhanced RSA + n!	ECC (256-bit)	AES (128-bit)
Key Size (bits)	2048	2048	256	128
Encryption Time (ms/block)	2.15	2.89	1.10	0.09
Decryption Time (ms/block)	4.30	5.07	1.40	0.11
Keyspace Complexity	2^{2048}	$2^{2048} \times n!$	2^{256}	2^{128}
Block Permutation Entropy (bits)	6.23	9.84	6.50	7.20
Avalanche Effect (Bit Change %)	53.1%	88.7%	78.5%	92.3%

One important difference is the keyspace complexity. With Enhanced RSA, the standard RSA keyspace grows from 2^{2048} to (2^{2048} x n!), which makes brute-force attacks almost impossible. The Enhanced RSA (9.84 bits) is much more resistant to statistical cryptanalysis than the Classical RSA (6.23 bits) when it comes to Block permutation entropy, which is a measure of randomness. The avalanche effect, which shows how much the output changes when one bit of the input changes, also shows that Enhanced RSA works better than Classical RSA and even ECC (88.7%). AES is better at preventing avalanches (92.3%), but Enhanced RSA strikes a good mix between being hard to use and being strong. This makes it especially useful in high-security settings that need strong asymmetric encryption protocols.

Because of the extra work that goes into combinatorial processing, the Enhanced RSA algorithm takes a little longer to encrypt (2.89 ms) than Classical RSA (2.15 ms). ECC and AES are much faster at encryption than both types of RSA (1.10 ms vs. 0.09 ms), which shows how useful they are in time-sensitive security applications, especially when resources are limited. At 0.11 ms, AES has the fastest decoding time, followed by ECC at 1.40 ms. It takes 4.30 ms for

classical RSA to work, but 5.07 ms for Enhanced RSA with $n!$ permutations. The graph shows the trade-off in Enhanced RSA: higher security through combinatorial complexity means a little slower speed.

8. Conclusion

The RSA method is much stronger now that it uses advanced combinatorial n! Permutations. The integration of n! permutations into RSA fortifies encryption strength by amplifying entropy and reducing predictability in ciphertext. This combinatorial complexity ensures higher resilience against modern cryptanalytic techniques while preserving the algorithm's core mathematical framework. This makes standard public-key infrastructure much safer. This improvement slightly raises the amount of work that needs to be done, but it has big benefits in terms of keyspace growth, entropy generation, and protection against traditional cryptographic threats. The factorial-based randomisation layer makes the process of making ciphertext more complicated, which means that brute-force and known-plaintext attacks can't be used. Higher entropy and a better cascade effect make sure that even small changes in input lead to outputs that are very hard to predict. This makes the encryption more resistant to differential cryptanalysis. It is different from other modern encryption methods like ECC and AES. It's still true that symmetric algorithms like AES are faster, but the improved RSA method is clearly better for apps that need better security over faster processing time. This method shows a lot of promise for keeping private messages safe in defence, banking, and distributed cloud settings.

References

[1] G. Barbu, "FA-LLLing for RSA: Lattice-based fault attacks against RSA encryption and signature," in Proc. 2022 Workshop on Fault Detection and Tolerance in Cryptography (FDTC), Italy, pp. 30–37 (2022).

[2] W. Cao, H. Shi, H. Chen, J. Chen, L. Fan, and W. Wu, "Lattice-based fault attacks on deterministic signature schemes of ECDSA and EdDSA," in Topics in Cryptology – CT-RSA 2022, Springer (2022).

[3] P. Basaligheh and S. Kothawade, "Biometric authentication systems: Advances in security and usability," *Int. J. Recent Adv. Eng. Technol.*, vol. 13, no. 1, pp. 20–25 (2025).

[4] A. Godbole, R. Dhabliya, V. Deshpande, S. A. Sivakumar, B. M. Shankar, and V. Khetani, "Ethical hacking and penetration testing strengthening cybersecurity posture through offensive security measures," *J. Discrete Math. Sci. Cryptogr.*, vol. 27, no. 4, pp. 1295–1305 (2024).

[5] Z. Liu, C. Cao, R. Xu, C. Liu, and P. K. M. Bi, "Analysis of two papers based on RSA algorithm: Plagiarism investigation of HRSA algorithm," in Proc. 2024 14th Int. Conf. Inf. Technol. Med. Educ. (ITME), Guiyang, China, pp. 461–465 (2024).

[6] N. D. Garisto, "This month in physics history September 2002: Schön scandal report is released," *Amer. Phys. Soc.* (Aug. 2022).

[7] S. Kumar, H. Singh, I. Gupta, and A. J. Gupta, "Symmetric encryption scheme based on quasigroup using chained mode of operation," *J. Discrete Math. Sci. Cryptogr.,* vol. 27, no. 8, pp. 2337–2364 (2024).

[8] C.-H. Hsia, S.-J. Lou, H.-H. Chang, and D. Xuan, "Novel hybrid public/private key cryptography based on perfect Gaussian integer sequences," *IEEE Access*, vol. 9, pp. 145045–145059 (2021).

[9] B. Sankhyan, S. Sharma, and M. Singh, "Exploring modern cryptographic algorithms: An experimental analysis," in Proc. 2024 15th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT), Kamand, India, pp. 1–7 (2024).

[10] S. Harsha, K. Aswini, G. Sahith, G. V. N. Abhiram, and D. B. Kumar, "Data integrity verification in images through SHA-based security," *Int. J. Adv. Comput. Eng. Commun. Technol.*, vol. 14, no. 1, pp. 92–102 (2025).

Received April, 2025