International Journal of Recent Innovations in Academic Research
E-ISSN: 2635-3040; P-ISSN: 2659-1561
Homepage: https://www.ijriar.com/
Volume-10, Issue-1, January-March-2026: 1-5

Research Article

Mathematical Foundations of RSA Encryption and Its Computational Implications

Doheon Kim

Gwacheon Jungang High School, Korea
Email: leodoheonkim@gmail.com

Article History
Received: December 06, 2025
Accepted: December 29, 2025
Published: January 05, 2026

Abstract
RSA is the most widely deployed public-key cryptosystem for key establishment and digital signatures. Its appeal comes from an unusually clean mathematical structure: modular exponentiation in the ring Z_N combined with Euler's theorem (or, more precisely, the Chinese Remainder Theorem over prime factors of N). This paper reviews the number-theoretic foundations that make RSA correct, then connects those foundations to computational realities: the asymptotic and practical costs of key generation and exponentiation, the security reductions and pitfalls introduced by padding and protocol composition, and the algorithmic landscape of integer factorization that governs recommended key sizes. We emphasize the gap between the mathematical idealization (a trapdoor permutation) and real implementations that must confront randomness failures, side channels, and evolving computational capabilities including quantum algorithms. The goal is to provide a unified view of RSA as both a mathematical object and an engineered system.
Keywords: RSA, Modular Arithmetic, Euler's Theorem, Chinese Remainder Theorem, Integer Factorization, OAEP, Computational Complexity, Side-Channel Security.

1. Introduction

Public-key cryptography resolves a key-distribution bottleneck that is intrinsic to symmetric encryption: two parties who have never met must somehow share a secret key before they can communicate privately. The conceptual breakthrough of public-key cryptography is that secrecy can be achieved even when an encryption key is public, provided that a corresponding decryption key remains private and computationally infeasible to derive. Diffie and Hellman articulated this paradigm in 1976 and identified digital signatures as a parallel application of the same idea [1]. RSA, introduced shortly thereafter, was the first practical public-key cryptosystem to see broad deployment and remains a cornerstone of modern security protocols [2]. At a high level, RSA defines a function f(x) = x^e mod N over integers modulo a composite N = pq. The public key is (N, e), while the private key is an exponent d that (approximately) inverts f. Correctness depends on classic number theory: if ed ≡ 1 (mod φ(N)), then (x^e)^d ≡ x (mod N) for all x in the appropriate message space. Security is intended to follow from the difficulty of inverting modular exponentiation without the trapdoor information (the factorization of N), which is closely related to the integer factorization problem.

However, RSA as deployed is not a single algorithm but a family of primitives wrapped into protocols. The computational implications of RSA therefore span multiple layers: (i) number-theoretic assumptions about factoring; (ii) algorithmic choices for fast exponentiation and key generation; (iii) padding and encoding schemes required to achieve semantic security and robust signatures; and (iv) implementation hazards such as timing leakage and fault attacks. This paper surveys each layer and highlights how mathematical structure shapes performance and risk.

2. Number-Theoretic Preliminaries

RSA operates in the multiplicative structure of the residue class ring Z_N = {0, 1, …, N−1} with arithmetic modulo N. For an integer N ≥ 2, define the set of units Z_N* = {x ∈ Z_N : gcd(x, N) = 1}. This set forms a finite

abelian group under multiplication modulo N, with order φ(N), Euler's totient function. When N is a product of two distinct odd primes p and q, φ(N) = (p−1)(q−1).

Euler's theorem states that for any x ∈ Z_N*, x^{φ(N)} ≡ 1 (mod N). A direct corollary is that if integers e and d satisfy ed ≡ 1 (mod φ(N)), then for all x ∈ Z_N*, x^{ed} ≡ x (mod N). This provides the core correctness identity for RSA on invertible residues.

The Chinese Remainder Theorem (CRT) strengthens this picture by decomposing arithmetic modulo N into arithmetic modulo its prime factors. For N = pq with gcd(p, q) = 1, CRT gives an isomorphism Z_N ≅ Z_p × Z_q. Computationally, CRT enables faster private-key operations: one can compute x^d mod p and x^d mod q and then recombine the results to obtain x^d mod N. Because exponentiation modulo a k-bit modulus is substantially slower than exponentiation modulo a k/2-bit modulus, CRT typically yields a 3–4× speedup in practice.

Finally, the extended Euclidean algorithm provides the constructive bridge between the public and private exponents. Given e and φ(N) with gcd(e, φ(N)) = 1, it computes d = e^{-1} mod φ(N). This algorithm runs in time polynomial in log N and is negligible compared with prime generation.

3. The RSA Construction and Correctness

RSA key generation proceeds as follows: (1) sample two large random primes p and q; (2) compute N = pq; (3) compute φ(N) = (p−1)(q−1) or, more commonly, λ(N) = lcm(p−1, q−1); (4) choose a public exponent e with gcd(e, λ(N)) = 1; and (5) compute d = e^{-1} mod λ(N). The public key is (N, e) and the private key is d (often stored with p, q, and CRT coefficients).

Encryption of a message representative m ∈ Z_N is c = m^e mod N. Decryption is m = c^d mod N. For signatures, the roles are reversed: a signer computes s = m^d mod N and a verifier checks m ≡ s^e (mod N). In practice, m is not a raw message but an encoded value produced by a padding scheme (Section 5).

Correctness can be shown in two steps. First, if m ∈ Z_N* then Euler's theorem implies m^{λ(N)} ≡ 1 (mod N). Since ed = 1 + k·λ(N) for some integer k, we have m^{ed} = m·(m^{λ(N)})^k ≡ m (mod N). Second, if m is not in Z_N* (i.e., shares a factor with N), Euler's theorem does not apply directly. Nonetheless, the RSA decryption identity still holds for all m ∈ Z_N when d is defined modulo λ(N) and the computation is interpreted via CRT: modulo p and q, Fermat's little theorem ensures the exponentiation reduces appropriately, and the congruences recombine uniquely modulo N. This is why many standards recommend using λ(N) rather than φ(N) when defining d.

Mathematically, RSA induces a permutation on Z_N* (and a bijection on suitably restricted message representatives) as long as e is coprime to λ(N). The trapdoor is the ability to compute d, which is easy if one knows p and q but is believed hard otherwise.

4. Security Assumptions and the Role of Factoring

RSA's security is often summarized as: 'breaking RSA is as hard as factoring N.' The relationship is subtler. Knowing the factorization of N immediately yields φ(N) (or λ(N)) and therefore d, so factoring implies the ability to invert RSA. The converse-whether inverting RSA implies factoring-is not known in full generality. There are reductions in restricted settings, but in general RSA inversion is assumed hard as an independent problem (the RSA problem). Boneh surveys two decades of attacks and emphasizes that many practical breaks exploit misuse rather than the underlying mathematics [3]. Still, the best-known general attacks against properly generated RSA keys reduce to factoring N. The dominant classical algorithm for large, random semiprimes is the Number Field Sieve (NFS), whose heuristic runtime is sub-exponential in log N and asymptotically faster than earlier methods [4]. This sub-exponential complexity is why key sizes must grow over time as computational power improves.

Security also depends on parameter choices. A small public exponent e (commonly 65537) improves performance, but if paired with deterministic or structured message representatives it can enable 'low-exponent' attacks. Håstad's broadcast attack shows that if the same plaintext is sent to multiple recipients with the same small e and insufficient randomization, the message can be recovered using the Chinese Remainder Theorem and integer root extraction [5]. More generally, Coppersmith's lattice methods can recover small roots of modular polynomials and underpin attacks when parts of the plaintext or structure are known [6]. Finally, RSA is threatened by quantum computation. Shor's algorithm can factor integers in time polynomial in

the bit-length of N, which would render RSA insecure against sufficiently large fault-tolerant quantum computers [7]. As of early 2026, such machines do not exist at the required scale, but the algorithm shifts RSA's long-term viability and motivates migration plans to post-quantum cryptography.

5. Padding, Encodings, and Provable Security

Raw RSA (textbook RSA) is deterministic: encrypting the same message representative twice yields the same ciphertext. Determinism violates modern notions of confidentiality such as indistinguishability under chosen-plaintext attack (IND-CPA). Moreover, textbook RSA is malleable: given c = m^e mod N, an attacker can form c' = (c·r^e) mod N, which decrypts to m·r mod N. Therefore, secure RSA encryption requires randomized padding.

The most widely standardized approach is RSAES-OAEP (Optimal Asymmetric Encryption Padding), introduced by Bellare and Rogaway and later standardized in PKCS #1 [8]. OAEP wraps the message with randomness via hash-based masking, turning a trapdoor permutation into an encryption scheme with strong security guarantees in the random oracle model. In practice, OAEP is used in 'hybrid encryption': RSA-OAEP encrypts a random symmetric key, and a symmetric cipher encrypts the bulk data.

For signatures, a parallel issue arises: textbook RSA signatures are vulnerable to existential forgery because RSA is multiplicative. Modern standards therefore hash the message and apply an encoding before exponentiation. PKCS #1 v1.5 and RSASSA-PSS are common schemes standardized in RFC 8017 [9]. PSS is designed to admit tighter security arguments (again in the random oracle model) and incorporates per-signature salt.

History shows why these layers matter. Bleichenbacher's 1998 adaptive chosen-ciphertext attack exploited PKCS #1 v1.5 encryption padding validation to create an oracle, leading to practical decryption of RSA ciphertexts under certain protocols such as SSL [10]. The lesson is that security depends not only on mathematics but also on how errors are handled and how protocol responses leak information. Contemporary guidance therefore emphasizes constant-time implementations, strict conformance to standards, and preference for schemes with strong security proofs (e.g., OAEP, PSS).

6. Computational Implications: Cost Models and Optimizations

RSA's computational cost is dominated by modular exponentiation. Using square-and-multiply, exponentiation to an exponent of t bits requires O(t) modular multiplications. For a k-bit modulus N, naive multiplication is O(k^2), while asymptotically faster methods exist; in practice, high-performance libraries use variants of Karatsuba, Toom-Cook, and FFT-based multiplication depending on size. The practical takeaway is that RSA encryption and signature verification (public-key operations) are relatively fast because the public exponent e is chosen small (commonly 65537). Decryption and signing (private-key operations) are slower because d is large and roughly k bits.

CRT optimization reduces private-key exponentiation cost. By computing modulo p and q separately, the multiplications involve k/2-bit numbers, reducing the overall work by about a factor of 4, plus overhead for recombination. Most implementations store dp = d mod (p−1), dq = d mod (q−1), and q^{-1} mod p to perform this recombination efficiently.

Key generation is also computationally significant, especially for constrained devices. It requires sampling candidate odd integers of the target bit-length and testing primality. Modern implementations use probabilistic primality tests (e.g., Miller–Rabin) and may incorporate additional checks for strong primes or constraints on p−1 and q−1 depending on policy. The expected time is dominated by the density of primes near 2^{k/2}, approximately 1/ln(2^{k/2}) = 2/(k ln 2). Thus, on average, O(k) random odd candidates are tested to find each prime.

These costs motivate design tradeoffs. Small e improves verification speed but can interact dangerously with poor padding. Large e mitigates certain low-exponent attacks but slows verification. Similarly, using a larger modulus improves security but increases latency, bandwidth (for certificates and signatures), and server load. Standards attempt to balance these concerns by recommending minimum key sizes and approved padding schemes; for example, NIST SP 800-56B Rev. 2 provides guidance for RSA-based key establishment, including security strength categories tied to modulus size [11].

Computational implications extend beyond raw time. Side-channel resistance can increase cost because constant-time exponentiation avoids data-dependent branches and memory access patterns. Blinding-

multiplying inputs by a random r^e before private-key exponentiation and then unblinding the result-adds extra exponentiations but is often necessary to mitigate timing and power analysis attacks. Thus, the mathematically 'same' RSA operation can have very different real-world cost depending on the threat model.

7. Broader Implications: Deployment, Key Sizes, and the Future

RSA's mathematical simplicity helped it become a de facto standard, but simplicity does not imply uniform safety. Large-scale incidents frequently stem from implementation mistakes: reuse of primes across devices, inadequate randomness, or protocol oracles. A recurring theme is that RSA inherits fragility from its reliance on prime generation. If p or q is biased or partially predictable, the effective security can collapse.

Key sizes are driven by the state of factoring algorithms and available computational resources. Sub-exponential algorithms like NFS imply that security does not scale linearly with key length. As a rule of thumb, doubling the bit-length of N yields more than a doubling of security, but the exact mapping depends on modeling assumptions and empirical records of factorization efforts. Operational guidance therefore tends to be conservative and periodically updated (e.g., by NIST and other standards bodies) [11].

RSA also competes with elliptic-curve cryptography (ECC), which typically offers comparable security with smaller keys and faster operations. Yet RSA remains entrenched because of its broad support in legacy systems, certificate infrastructures, and hardware security modules. In the near term, RSA is likely to persist in signatures (especially where verification speed matters), even as key exchange migrates toward elliptic-curve and post-quantum mechanisms.

In the longer term, Shor's algorithm establishes a clear endgame: if scalable quantum computers become practical, RSA must be replaced. This prospect has already reshaped cryptographic engineering, shifting attention toward hybrid protocols and crypto-agility, and toward post-quantum candidates whose security rests on different mathematical problems. In this sense, RSA's enduring value is not only as a deployed primitive but also as a canonical example of how deep mathematics and computational constraints co-determine what 'secure' means in practice.

8. Conclusion

RSA's core idea-using modular exponentiation with a trapdoor derived from the factorization of N-remains one of the most elegant constructions in applied mathematics. Its correctness is a direct consequence of Euler's theorem and CRT, and its efficiency is shaped by fast arithmetic and careful parameter selection. At the same time, RSA's real-world security depends critically on layers that are not present in the textbook definition: randomized padding, strict protocol behavior, side-channel countermeasures, and continuous alignment with the evolving cost of factoring. The computational implications are therefore twofold. RSA is computationally practical because it enables fast public-key operations and admits optimizations like CRT; but it is also computationally constrained because private-key operations are heavy and because key sizes must grow to stay ahead of classical factoring algorithms. Looking forward, quantum algorithms place a fundamental bound on RSA's long-term security. Understanding RSA as both a mathematical object and a computational artifact is essential for using it responsibly today and for transitioning away from it when necessary.

Declarations

Acknowledgments: The author acknowledges that this study was conducted independently and did not receive support from any institution or external organization.

Artificial Intelligence (AI) Use Statement: The author declares that no artificial intelligence tools were used for the generation of scientific content, analysis, or conclusions in this manuscript.

Author Contribution: The author confirms sole responsibility for all aspects of the study, including conception and design, data collection, data analysis and interpretation, and manuscript preparation.

Conflict of Interest: The author declares that no competing interests exist.

Consent to Publish: The author agrees to the publication of this article in International Journal of Recent Innovations in Academic Research.

Data Availability Statement: The data supporting the findings of this study are fully available within the manuscript.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Institutional Review Board Statement: This study did not require approval from an Institutional Review Board.

Informed Consent Statement: Informed consent was not applicable to this research.

Research Content Statement: The manuscript represents original research and has not been published previously nor is it under consideration for publication elsewhere.

References

1. Diffie, W. and Hellman, M.E. 1976. New directions in cryptography. IEEE Transactions on Information Theory, 22(6): 644–654.
2. Rivest, R.L., Shamir, A. and Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2): 120–126.
3. Boneh, D. 1999. Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society, 46(2): 203–213.
4. Lenstra, A.K., Lenstra, H.W., Jr., Manasse, M.S. and Pollard, J.M. 1990. The number field sieve. In: Proceedings of the twenty-second annual ACM symposium on theory of computing (pp. 564–572). Association for Computing Machinery. https://doi.org/10.1145/100216.100295
5. Håstad, J. 1988. Solving simultaneous modular equations of low degree. SIAM Journal on Computing, 17(2): 336–341.
6. Coppersmith, D. 1996. Finding a small root of a univariate modular equation. In: Maurer, U., (Ed.), Advances in cryptology—EUROCRYPT 1996 (Lecture Notes in Computer Science, Vol. 1070, pp. 155–165). Springer.
7. Shor, P.W. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5): 1484–1509.
8. Bellare, M. and Rogaway, P. 1995. Optimal asymmetric encryption-How to encrypt with RSA. In: De Santis, A., (Ed.), Advances in cryptology-EUROCRYPT 1994 (Lecture Notes in Computer Science, Vol. 950, pp. 92–111). Springer.
9. Moriarty, K. (Ed.). 2016. PKCS #1: RSA cryptography specifications version 2.2 (RFC 8017). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc8017
10. Bleichenbacher, D. 1998. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H., (Ed.), Advances in cryptology-CRYPTO 1998 (Lecture Notes in Computer Science, Vol. 1462, pp. 1–12). Springer. https://doi.org/10.1007/BFb0055716
11. Barker, E., Chen, L., Roginsky, A., Vassilev, A., Davis, R. and Simon, S. 2019. Recommendation for pair-wise key establishment using integer factorization cryptography (NIST Special Publication 800-56B Rev. 2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-56Br2

Citation: Doheon Kim. 2026. Mathematical Foundations of RSA Encryption and Its Computational Implications. International Journal of Recent Innovations in Academic Research, 10(1): 1-5.