International Journal of Safety and Security Engineering
Vol. 16, No. 9, September, 2026, pp. 2057-2072
Journal homepage: http://iieta.org/journals/ijsse

DEE-RSA: A Structured Framework for the Design, Evaluation, and Experimentation of RSA Variants

Dian Rachmawati[1*], Maya Silvi Lydia[1], Mohammad Andri Budiman[1], Romi Fadillah Rahmat[2]

[1] Department of Computer Science, Universitas Sumatera Utara, Medan 20115, Indonesia
[2] Department of Information Technology, Universitas Sumatera Utara, Medan 20115, Indonesia

Corresponding Author Email: dian.rachmawati@usu.ac.id

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ijsse.160911

Received: 19 July 2026
Revised: 23 August 2026
Accepted: 4 September 2026
Available online: 30 September 2026

Keywords:
Rivest-Shamir-Adleman variants, structured evaluation framework, cryptanalysis checklist, benchmarking protocol, reproducibility, retrospective evidence audit

ABSTRACT

Rivest-Shamir-Adleman (RSA) research has produced many structural and implementation variants, yet published studies often use incompatible security, performance, and reproducibility criteria. This paper proposes Design, Evaluation, and Experimentation of RSA variants (DEE-RSA), a structured three-phase framework for the Design, Evaluation, and Experimentation of RSA variants. To make framework construction traceable, we document a targeted structured source-selection procedure over a 52-item corpus, including the represented databases and repositories, search concepts, eligibility rules, source screening, and criterion-derivation steps. The resulting framework combines complete mathematical specification, correctness verification, explicit hardness assumptions, an eight-category cryptanalysis checklist, a five-metric benchmarking protocol, and a ten-item reproducibility checklist. We retrospectively apply the framework to eight representative families—Classical RSA, Chinese Remainder Theorem-based RSA (CRT-RSA), Multi-Prime RSA, Takagi Multi-Power RSA, Rebalanced RSA, Batch RSA, R-Prime RSA, and XRSA (an XOR-based four-prime RSA variant)—using explicit Present, Partial, and Absent coding rules. Because this coding was an author-team evidence audit without independent inter-rater assessment, it is presented as an illustrative retrospective evaluation rather than an external assessment or consensus. We also report a reproducible benchmark of five implementations at 1024, 2048, and 4096 bits, with 100 independent trials per variant and size. Results include key-generation, encryption, and decryption time, decryption throughput, peak traced Python memory, standard deviations, and 95% confidence intervals. The findings show that performance rankings depend on both operation and modulus size; they also demonstrate how DEE-RSA exposes incomplete claims while preserving evidence-qualified outcomes. DEE-RSA is therefore offered as a transparent research framework and candidate reporting guideline, not as an externally structured methodology.

1. INTRODUCTION

The Rivest-Shamir-Adleman (RSA) cryptosystem, introduced in 1978 [1], has remained one of the most widely deployed public-key cryptographic mechanisms. A two-decade survey of RSA attacks [2] classified them into four categories: elementary attacks exploiting system misuse, low private exponent attacks, low public exponent attacks, and implementation attacks. Despite the emergence of post-quantum algorithms standardized by the National Institute of Standards and Technology (NIST) [3], RSA is projected to remain relevant throughout a 10–20 year transition period [4, 5].

Research on RSA variants has expanded substantially in both breadth and depth. A systematic literature review [6] classified RSA-based schemes into eleven distinct domains: Hybrid, Parallel, Cloud, Image, Multiple-Keys, Chinese-Remainder-Theorem-based, Digital-Signatures, K-Nearest-Theorem-based, Batch, Wireless, and Core-Modifications. The Core-Modifications domain was further subdivided into Fast Variants, Integrated Schemes, Security-Focused, and Mathematical Modifications [6]. Well-established fast variants include Chinese Remainder Theorem-based RSA (CRT-RSA) [7], Multi-Prime RSA [8], Multi-Power RSA [9], Rebalanced RSA [10], Batch RSA [11], and R-Prime RSA [12]. A bibliometric analysis [13] further identified RSA variants as a distinct co-citation cluster in the lattice-based cryptanalysis literature.

However, RSA variant research lacks a structured evaluation methodology. A systematic review [6] concluded that a consistent taxonomy is needed to align execution time, security, and performance, while its quality assessment of reviewed papers revealed that many studies omit practical implementation, comparison with alternatives, or evaluation of all algorithmic phases. A survey of fast RSA variants [14] emphasized that their security remained an open problem and urged cautious use. In the post-quantum domain, an evaluation study [15] likewise identified inconsistent comparisons that

hinder scheme selection for specific use cases. Similarly, the reproducibility literature reveals systemic deficiencies: Olszewski et al. [16] reported that only 10% of published applied security research provides runnable artifacts and only 3.9% of papers produce reproducible results, while Olszewski et al. [17] found that 60% of security papers lack accompanying code and 56% of provided artifacts fail to execute entirely.

The absence of a structured methodology for developing, evaluating, cryptanalyzing, benchmarking, and reporting novel RSA variants produces several undesirable consequences: (i) results across studies are incomparable due to inconsistent metrics and evaluation protocols; (ii) security claims are difficult to verify independently; (iii) replication is hindered by incomplete documentation; and (iv) reviewers lack a systematic framework for assessing the completeness of proposed variants. To the best of our knowledge, the RSA variant literature has not yet addressed all of these dimensions within a single unified framework.

To address this gap, we propose Design, Evaluation, and Experimentation of RSA variants (DEE-RSA), a three-phase structured framework for the Design, Evaluation, and Experimentation of novel RSA variants. The contributions of this work are:

(1) DEE-RSA Framework—A structured three-phase methodology (Design–Evaluation–Experimentation) that guides the systematic development and evaluation of RSA variants.

(2) Cryptanalysis Checklist—A comprehensive checklist of known RSA attacks, derived from the attack taxonomy of Boneh [2] and extended with modern attack vectors, against which every novel variant must be evaluated.

(3) Benchmarking Protocol—A structured experimental protocol specifying metrics, measurement procedures, and statistical reporting requirements adapted from the generic benchmark model of Damaj and Kasbah [18] and the comprehensive evaluation framework of Pote and Bansode [15].

(4) Reporting Guideline—An explicit template for reporting RSA variant research, ensuring completeness and consistency across studies.

(5) Reproducibility Package—A structured reproducibility checklist based on the SciRep framework [19], ACM artifact evaluation badging criteria [16], and established reproducibility recommendations [17, 20].

The remainder of this paper is organized as follows. Section 2 reviews related work on RSA variants and existing evaluation practices, highlighting research gaps through a structured gap analysis. Section 3 presents the DEE-RSA framework in detail. Section 4 evaluates the framework through retrospective application to representative RSA variants and an illustrative case study on XRSA. Section 5 describes the reproducibility package. Section 6 provides a step-by-step usage guide with a workflow diagram, instructions, and accompanying templates. Section 7 discusses strengths, limitations, and future work. Section 8 concludes.

2. RELATED WORKS AND RESEARCH GAPS

2.1 Framework construction methodology

Review status and corpus. DEE-RSA was constructed through a targeted structured review, not a formal systematic review or a consensus-standard process. The working corpus comprised the 52 unique sources cited in this article and was finalized with a 30 June 2026 cutoff. The corpus represents IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, USENIX, NIST, IETF, patent, thesis, and cross-reference sources. Seminal syntheses on RSA attacks and fast variants [2, 14] and the systematic RSA review [6] were used as seed studies for backward and forward citation tracing.

Search concepts and eligibility. The source-identification concepts combined (RSA variant OR fast RSA OR CRT-RSA OR Multi-Prime RSA OR Multi-Power RSA OR rebalanced RSA OR batch RSA OR R-Prime RSA OR XRSA) with (correctness OR cryptanalysis OR attack OR benchmark OR performance OR reproducibility). Eligible sources were English-language primary constructions, standards, seminal cryptanalysis, empirical evaluations, or reproducibility frameworks with directly inspectable methods. Duplicates, nontechnical commentary, and application papers that used unchanged RSA without evaluating a variant were excluded.

Study selection and data extraction. Titles and abstracts were screened for direct relevance to one or more framework dimensions; the full text or complete bibliographic record was then checked for an explicit algorithm, attack model, measurement method, reporting requirement, or artifact criterion. For every retained source, evidence was extracted into five provisional domains: design specification, correctness, cryptanalysis, experimentation, and reproducibility. The bibliography is the auditable record of included sources; no claim of exhaustive database coverage is made.

Criterion derivation. Overlapping requirements were merged, source-specific wording was generalized only when the same requirement was supported by multiple sources, and the stricter contemporary requirement was retained where guidance conflicted. The resulting criteria were mapped to Design, Evaluation, and Experimentation, and each checklist item was linked back to at least one cited source. This process supports transparency and repeatability, but it does not establish external validity, expert consensus, or community standardization.

2.2 Development of Rivest-Shamir-Adleman variants

The original RSA cryptosystem [1] uses a modulus $n = pq$, where p and q are distinct large primes, with public exponent e and private exponent d satisfying $ed \equiv 1 \pmod{\phi(n)}$. Since its publication, numerous variants have been proposed to improve decryption speed, reduce computational cost, or adapt RSA to constrained environments. Imam et al. [6] classified all RSA-based schemes into eleven domains and identified Core-Modifications as the category most relevant to algorithmic improvement, further subdivided into Fast Variants, Integrated Schemes, Security Focused, and Mathematical Modifications.

2.2.1 Multi-Prime Rivest-Shamir-Adleman

Multi-Prime RSA generalizes standard RSA by using $r \geq 3$ distinct primes p_1, ..., p_r in the modulus $n = \prod_{i=1}^{r} p_i$ [8, 14]. Decryption is accelerated by splitting the exponentiation into r smaller modular exponentiations modulo each prime factor, combined via the Chinese Remainder Theorem (CRT). An analysis of fast RSA variants [14] derived the theoretical speedup as r^2 (approximately 2.25 × for $r = 3$ with a 1024-bit

modulus) and confirmed through OpenSSL benchmarking on a 750 MHz Pentium III that batching further improves throughput. A peer-reviewed implementation study [12] evaluated R-Prime RSA as a hybrid combining a Multi-Prime structure with Rebalanced exponents.

2.2.2 Multi-Power Rivest-Shamir-Adleman

The Multi-Power RSA construction [9] uses an RSA-type cryptosystem modulo $n = p^k q$, differing from Multi-Prime RSA by using repeated prime factors rather than distinct ones. For k = 2 (modulus $p^2 q$), Takagi reported decryption approximately 3 times faster than CRT-RSA using the Quisquater–Couvreur method for a 768-bit modulus with 256-bit primes. Boneh and Shacham [14] derived the theoretical speedup as $\frac{r^3}{2}$ (approximately 3.38× for r = 3 at 1024 bits). The fast decryption uses Hensel lifting to compute $M \bmod p^k$ without requiring the full secret exponent [9].

2.2.3 Rebalanced Rivest-Shamir-Adleman

Rebalanced RSA [10] shifts computational work from the decryptor to the encryptor by making the private CRT exponents d_p, d_q small (typically w = 160 bits) while keeping the full private exponent d large enough to resist Wiener's continued fraction attack [2, 14]. This produces a large public exponent e (on the order of n), making encryption slower but decryption significantly faster. The theoretical speedup is $\frac{2n}{w}$ [12].

2.2.4 Chinese Remainder Theorem-based Rivest-Shamir-Adleman

CRT-RSA [7] accelerates decryption by computing $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$ separately (where $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$), then combining results using the CRT. The theoretical speedup is approximately 4 × over standard RSA [12, 14]. CRT-RSA is the most widely deployed variant and is supported by virtually all cryptographic libraries. Buhrow et al. [21] demonstrated CRT-RSA acceleration on Intel Xeon processors using AVX-512 vector instructions, achieving 1.9 × improvement in decryption throughput compared to OpenSSL.

2.3 Existing evaluation practices

A systematic analysis of the RSA variant literature reveals significant inconsistencies in evaluation practices. The systematic review [6] applied five quality assessment criteria to surveyed papers: (A) practical implementation, (B) comparison with other algorithms, (C) extension from conference to journal versions, (D) evaluation of all algorithm phases, and (E) alignment of objectives with conclusions. Their analysis demonstrated that many studies fail to meet these criteria, with a substantial proportion lacking practical implementation or comprehensive comparison.

Table 1 presents a gap analysis of representative RSA variant publications across five evaluation dimensions: Correctness Proof, Cryptanalysis, Benchmarking, Reporting Completeness, and Reproducibility. The analysis follows the quality-assessment and systematization methods reported in [6, 14].

As shown in Table 1, no single publication in the surveyed literature provides complete coverage across all five evaluation dimensions. Correctness proofs or algorithmic descriptions are nearly universal. However, cryptanalysis is often partial—limited to informal claims of factoring resistance without systematic evaluation against known attacks. Benchmarking, when present, typically reports only execution time without consistent metrics or statistical rigor. Few papers provide source code or reproducibility packages. This fragmentation mirrors broader trends in the security literature: only 28.9% of applied security papers make artifacts available, and the introduction of artifact evaluation committees alone has not significantly improved this rate [16, 17]. These findings motivate the need for a structured framework that systematically addresses all dimensions of RSA variant evaluation.

Table 1. Gap analysis of evaluation practices in Rivest-Shamir-Adleman (RSA) variant literature

Study	Correctness	Cryptanalysis	Benchmark	Reporting	Reproducibility
Classical RSA [1]	Present	Partial	Partial	Present	Absent
Chinese Remainder Theorem-based RSA (CRT-RSA) [7]	Present	Partial	Partial	Partial	Absent
Multi-Prime RSA [8, 14]	Present	Partial	Partial	Partial	Absent
Takagi RSA [9]	Present	Partial	Partial	Present	Absent
Rebalanced RSA [10, 14]	Present	Partial	Partial	Partial	Absent
Batch RSA [11]	Present	Partial	Partial	Present	Absent
R-Prime RSA [12]	Present	Partial	Partial	Partial	Absent

3. THE DEE-RSA FRAMEWORK

The DEE-RSA framework is structured into three sequential phases: Design, Evaluation, and Experimentation. Each phase comprises specific sub-phases with clearly defined inputs, activities, and outputs. The framework draws methodological inspiration from the generic benchmark model of Damaj and Kasbah [18], which identifies six elements of any evaluation framework (Goal, Input, Activities, Output, Outcomes, Performance), and from the systematic review methodology of Imam et al. [6], which demonstrated the need for structured quality assessment in RSA research.

3.1 Phase I: Design

The Design phase establishes the theoretical foundation of the RSA variant and produces a Candidate RSA Variant. This phase is guided by the principle that every variant should be specified with sufficient mathematical rigor to enable independent verification [18].

3.1.1 Motivation and problem statement

The designer must articulate the specific limitation of existing RSA constructions that the proposed variant addresses. As documented by Boneh [2] and Boneh and Shacham [14], well-motivated directions include decryption

speed optimization, ciphertext size reduction, resistance to specific attack classes, and adaptation to resource-constrained environments [12].

3.1.2 Mathematical formulation

The variant must be expressed through complete mathematical definitions including: the modulus structure (e.g., $n = pq$, $n = p^k q$, $n = \prod_{i=1}^{r} p_i$), the key relationship (e.g., $ed \equiv 1 \ (mod\ \phi(n))$, $ed \equiv 1 \ (mod\ \lambda(n))$, or $ed \equiv 1 \ (mod\ lcm(p-1, q-1))$ as in Takagi [9] or the recent XRSA construction [22]), the encryption function, and the decryption function [1, 23]. All algebraic assumptions (e.g., $gcd(e, \phi(n)) = 1$) must be stated explicitly.

3.1.3 Parameter selection

Parameters must be specified with concrete bounds justified by cryptanalytic results. For the modulus size, Boneh [2] recommended at least 1024 bits (in 1999); current practice requires ≥2048 bits [24]. For the public exponent, Boneh [2] noted that the minimum value $e = 3$ is vulnerable to broadcast attacks, and $e = 65537$ (i.e., $2^{16r} + 1$) is the recommended standard. For the private exponent, Wiener [10] proved that $d < \frac{1}{3}N^{\frac{1}{4}}$ is insecure using continued fractions, and Boneh and Durfee [25] improved this bound to $d < N^{0.292}$ using Coppersmith's lattice-based method. The Fermat condition $|p - q| > 2N^{\frac{1}{4}}$ must be enforced to prevent Fermat factorization [2, 26].

3.1.4 Key generation design

The key generation algorithm must specify the complete procedure for deriving public and private key material from the selected parameters. Where randomness is required, the entropy source must be identified [27]. Primality testing should follow established procedures such as Miller–Rabin with sufficient rounds for the desired security level, and improper primality testing in adversarial settings can lead to catastrophic key compromise [26].

Output: A complete specification of the Candidate RSA Variant, including all algorithms, parameter bounds, and design rationale.

3.2 Phase II: Evaluation

The Evaluation phase assesses theoretical validity, stated security properties, and unresolved risks, producing an Evaluated RSA Variant with an evidence-qualified security report. This output is not a certification or external assessment.

3.2.1 Correctness verification

A formal proof of correctness must establish that $Dec(sk, Enc(pk, m)) = m$ for all valid messages m in the designated message space. The proof should follow the structure of established RSA correctness proofs, which rely on Euler's theorem or Carmichael's theorem [23]. Variants with non-standard modulus structures (e.g., $n = p^k q$) require careful treatment of the group structure to avoid correctness failures [9]. For CRT-based variants, the correctness of the CRT recombination must be verified for all possible exponent combinations.

3.2.2 Security assessment

The security assessment formalizes the hardness assumptions on which the variant's security relies:

•Standard RSA Assumption: Given $(n, e, y = x^e \ mod\ n)$, it is computationally infeasible to recover x without knowledge of the factorization of n [1, 2]. The RSA problem is random self-reducible: an algorithm inverting RSA on a non-negligible fraction of inputs can be transformed to invert RSA on all inputs [2, 23].

•Factoring Assumption: Given n, it is computationally infeasible to recover p and q [2, 28]. The most effective classical algorithm is the Number Field Sieve (NFS) with sub-exponential complexity [28], while the Elliptic Curve Method (ECM) is more efficient for finding small prime factors [9]. Multi-Prime variants with smaller individual primes face increased ECM vulnerability [14].

•Security Properties: The designer must state which standard security definitions the variant achieves—indistinguishability under chosen-plaintext attack (IND-CPA) [29], indistinguishability under adaptive chosen-ciphertext attack (IND-CCA2) [30, 31], or neither—and provide appropriate reductions.

3.2.3 Cryptanalysis checklist

Every RSA variant must be evaluated against the following structured checklist of known attacks, derived from the comprehensive taxonomies of Boneh [2], Nitaj [32], and subsequent cryptanalytic literature. We categorize attacks following Boneh's four-category classification: elementary attacks, low private exponent attacks, low public exponent attacks, and implementation attacks [2].

(1) Wiener's continued fraction attack. Wiener [10] proved that if $d < \frac{1}{3}N^{\frac{1}{4}}$, the private exponent d can be efficiently recovered from the continued fraction expansion of $\frac{e}{N}$. Verheul and van Tilborg [33] extended the exhaustive search range, and de Weger [34] further improved the bound for small $|p - q|$. Recent work by Bunder and Tonien [35] extended Wiener's attack to work for $d^2 e < 8N^{\frac{3}{2}}$ using a modified continued fraction approach, and multicore systems were leveraged to further extend practical attack boundaries [36].

(2) Boneh–Durfee lattice attack. Boneh and Durfee [25] improved Wiener's bound to $d < N^{0.292}$ using Coppersmith's lattice-based method for finding small roots of modular polynomial equations. Nitaj [32] subsequently introduced the concept of F-constrained public exponents, where the public exponent e satisfies a specific linear relation, and showed that at least $O(n^{\frac{3}{4}-\varepsilon})$ Weak exponents exist for the function $F_1(u) = p(q - u)$. Blömer and May [37] further extended the attack to cases where $ex + y \equiv 0 \ (mod\ \phi(N))$ with small x and y.

(3) Factorization attacks. The NFS is the most powerful general-purpose factoring algorithm [28], while ECM is more efficient when the modulus contains relatively small prime factors [14]. Fermat's factorization method exploits primes that are too close together (when $|p - q| < 2N^{\frac{1}{4}}$) [2, 26]. The greatest common divisor (GCD) attack exploits shared prime factors across different RSA keys due to insufficient entropy during key generation [26, 38]. Adversarial primality testing implementations can produce keys that pass Miller–Rabin but are vulnerable to specialized factoring methods [26].

(4) Timing attacks. Measuring decryption time can leak information about the private exponent [39]. Aldaya et al. [40] presented the first practical single-trace cache-timing attack against the binary GCD step during RSA key generation, achieving approximately a 27% success rate for complete

RSA-2048 private key recovery via Flush + Reload.

(5) Chosen ciphertext attacks. Bleichenbacher [41] introduced an adaptive chosen ciphertext attack against RSA Public-Key Cryptography Standards (PKCS) #1 v1.5, showing that an oracle returning a single bit indicating PKCS conformance enables full plaintext recovery. The ROBOT attack demonstrated that Bleichenbacher-style vulnerabilities persisted in widely deployed Transport Layer Security (TLS) implementations two decades later [42], and Meyer et al. [43] identified additional Bleichenbacher side channels in Secure Sockets Layer/Transport Layer Security (SSL/TLS) implementations.

(6) Partial key exposure attacks. Boneh, Durfee, and Frankel [44] showed that knowledge of the $\left\lceil \frac{n}{4} \right\rceil$ least significant bits of d enables efficient recovery of the complete private key, and that partial information about p, q, or d can be exploited through lattice-based methods.

(7) Fault injection attacks. Boneh, DeMillo, and Lipton [45] demonstrated that a single random fault during CRT-RSA decryption can expose the factorization of n. Chiu and Xiong [46] systematized the knowledge on fault injection attacks, and Erata et al. [47] proposed Random Self-Reducibility (RSR) as a systematic countermeasure.

(8) Side-channel attacks. Aljuffri et al. [48] demonstrated thermal side-channel key recovery from both protected and unprotected RSA implementations, while Socha et al. [49] identified power side-channel analysis as the most researched threat category in non-invasive passive attacks.

Output: An Evaluated RSA Variant accompanied by a cryptanalysis report that distinguishes supported resistance, identified susceptibility, and open or untested attack categories.

3.3 Phase III: Experimentation

The Experimentation phase subjects the evaluated variant to empirical performance evaluation, producing an Experimental Evaluation Report. This phase follows the generic benchmark model of Damaj and Kasbah [18] and adopts the performance metric framework of Pote and Bansode [15].

3.3.1 Benchmarking protocol

The benchmarking protocol specifies five mandatory metric categories:

1) Key Generation Time—Time required to produce a complete key pair (pk, sk), measured in milliseconds (ms) and reported as mean $\pm$ standard deviation over a minimum of 100 independent trials [12].

2) Encryption Time—Time to compute $c = Enc(pk, m)$, measured per operation and reported with confidence intervals [15].

3) Decryption Time—Time to compute $m = Dec(sk, c)$, measured per operation and reported with confidence intervals [12].

4) Throughput—Number of cryptographic operations completed per second, measured across key sizes (1024, 2048, and 4096 bits as applicable) [15].

5) Memory Usage—Peak memory consumption during key generation, encryption, and decryption, reported in kilobytes (KB) or megabytes (MB) [15, 18].

All benchmarks must be conducted on a documented hardware platform (CPU model, clock speed, RAM) and software environment (operating system, programming language, compiler/interpreter version, cryptographic library) [17]. The protocol recommends SUPERCOP (System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives) as an established benchmarking tool where applicable [15].

3.3.2 Comparative evaluation

The proposed variant must be compared against:

•Classical (textbook) RSA as the baseline reference [1];

•At least one established fast variant from the literature (e.g., CRT-RSA [7] or Multi-Prime RSA [8, 14]);

•At least one recent variant representative of the same design objective (e.g., XRSA [22] for Multi-Prime constructions).

Comparison tables should include: key size, encryption time, decryption time, throughput, ciphertext size, key generation time, and memory usage [6, 14].

3.3.3 Statistical reporting

All experimental results must be reported with complete descriptive and inferential statistics:

•Mean (μ) as the measure of central tendency.

•Standard Deviation (σ) as the measure of dispersion.

•95% Confidence Intervals for the mean, computed as $\mu \pm 1.96 \cdot \frac{\sigma}{\sqrt{n}}$ where n is the number of trials.

•Minimum of 100 independent trials per measurement to obtain stable empirical estimates.

Results should be presented in tabular form with uncertainty estimates and, where informative, visualized through box plots or histograms [15, 22].

Output: An Experimental Evaluation Report containing all measurement data, analysis, and comparison tables.

4. RETROSPECTIVE FRAMEWORK EVALUATION

To evaluate the coverage and practical interpretability of DEE-RSA, we retrospectively applied its criteria to eight representative RSA families. This exercise is an evidence audit of the source publications; it is not an external assessment of the framework or a claim that the underlying variants are cryptographically approved.

4.1 Variant selection and coding procedure

(1) Selection rationale. The sample includes the classical baseline [1]; established acceleration mechanisms based on CRT, distinct primes, repeated prime powers, short CRT exponents, and batching [7-11]; the hybrid R-Prime construction [12]; and the more recent four-prime XRSA proposal [22]. These eight families span the principal structural and performance-oriented categories identified by the RSA synthesis literature [6, 14].

(2) Operational definitions. Present means that the source explicitly specifies or performs the criterion and reports inspectable evidence for the relevant algorithmic stages. Partial means that the criterion is mentioned or partly evaluated but omits required attacks, parameters, baselines, metrics, statistics, or artifacts. Absent means that no inspectable evidence for the criterion was found in the publication or its identified supplement.

(3) Assessment procedure. Each source was coded against the same definitions and then discussed within the author team to resolve ambiguous evidence. The study did not use blinded independent assessors and does not report an inter-rater coefficient; consequently, the ratings are explicitly treated as

illustrative, evidence-qualified judgments. A future external replication should preregister the rubric and use at least two independent coders.

(4) Interpretation rule. Ratings concern completeness of reported evidence, not the intrinsic security or practical value of a construction. An Absent score therefore means not reported, rather than proven insecure or ineffective.

4.2 Retrospective evaluation results

Table 2 summarizes the eight-family audit at the phase level. All ratings use the operational definitions above, and the principal evidence gap is stated to make each judgment traceable rather than presenting an unexplained symbol.

The retrospective results indicate recurring reporting gaps rather than proving universal deficiencies across the RSA literature. Modern attack coverage, complete multi-metric statistics, and reusable artifacts were the least consistently reported dimensions. Because the sample was purposive and the coding was not independently replicated, the table should be read as a transparent demonstration of framework use, not as a population estimate or consensus assessment.

Table 2. Retrospective evaluation of eight RSA families against DEE-RSA phase criteria

Variant	Design	Evaluation	Experiment	Reproduce	Principal Evidence Gap
Classical RSA [1]	Present	Partial	Partial	Absent	Original evidence predates modern attack, statistical, and artifact criteria.
Chinese Remainder Theorem-based RSA (CRT-RSA) [7]	Present	Partial	Partial	Absent	Correct CRT method; no systematic modern implementation-attack evaluation.
Multi-Prime RSA [8, 14]	Present	Partial	Partial	Absent	Speed/security trade-off discussed, but metric and artifact reporting is incomplete.
Takagi Multi-Power RSA [9]	Present	Partial	Partial	Absent	Formal construction and speed analysis; limited modern attack and reproducibility evidence.
Rebalanced RSA [10, 14]	Present	Partial	Partial	Absent	Short-exponent bounds are central, but evaluation coverage remains incomplete.
Batch RSA [11]	Present	Partial	Partial	Absent	Batch objective is clear; evidence does not cover the complete DEE-RSA criteria.
R-Prime RSA [12]	Present	Partial	Partial	Absent	Hybrid construction is specified; empirical and reproducibility evidence is incomplete.
XRSA [22]	Partial	Partial	Partial	Absent	Some algorithms and timing data are given, but claims, attacks, statistics, and artifacts are incomplete.

Note: Rivest-Shamir-Adleman (RSA); Design, Evaluation, and Experimentation of RSA variants (DEE-RSA).

4.3 Illustrative case study: Applying DEE-RSA to XRSA

To demonstrate the practical applicability of the DEE-RSA framework, this subsection presents a detailed walkthrough of applying each phase of the framework to the XRSA variant proposed by Imam et al. [22]. XRSA was selected because it represents a contemporary Multi-Prime RSA variant with four distinct prime factors and XOR-based intermediate operations, making it an instructive case for illustrating the completeness that DEE-RSA mandates.

4.3.1 Phase I: Design—Specifying XRSA under DEE-RSA

Motivation and Problem Statement. Imam et al. [22] proposed XRSA to "enhance the security of the RSA algorithm" through three mechanisms: (i) employing four random primes instead of two to increase brute-force resistance, (ii) introducing intermediate variables E_1, E_2 and E' for public exponent derivation so that the public key exponent E is not computed directly, and (iii) applying exclusive OR (XOR) operations with modular arithmetic in the final encryption and decryption steps to increase algebraic complexity. The design objective was to increase brute-force resistance while maintaining acceptable encryption and decryption performance compared to standard RSA [6, 22].

Mathematical Formulation. Under DEE-RSA, XRSA must be specified through complete mathematical definitions following the original exposition in Imam et al. [22]:

•Modulus construction: Select four distinct random primes p_1, p_2, p_3, p_4. Compute intermediate products $x = p_1 p_2$ and $y = p_3 p_4$, then form the modulus $N = x \cdot y = p_1 p_2 p_3 p_4$. The Euler totient is $\phi(N) = (p_1 - 1)(p_2 - 1)(p_3 - 1)(p_4 - 1)$.

•Key generation:

a. Select E_1 such that $1 < E_1 < \phi(x)$ and $gcd(E_1, \phi(x)) = 1$.
b. Select E_2 such that $1 < E_2 < \phi(y)$ and $gcd(E_2, \phi(y)) = 1$.
c. Verify $\gcd(E_1 \cdot E_2,\ \phi(N)) = 1$.
d. Compute intermediate public exponent $E' = (E_1 \times E_2)\ mod\ N$.
e. Compute intermediate private exponent D' such that $E' \cdot D' \equiv 1\ (mod\ \phi(N))$.
f. Compute final public key exponent $E = E' \oplus N$ (bitwise XOR).
g. Compute final private key exponent $D = D' \oplus N$ (bitwise XOR).
h. Public key: (E, N), Private key: (D, N).

•Encryption function: $E'' = E \oplus N$ (recovers E' since XOR twice with N cancels), then $C = M^{E''} \bmod N$.

•Decryption function: $D'' = D \oplus N$ (recovers D'), then $M = C^{D''} \bmod N$.

•Algebraic assumptions: $\gcd(E_1, \phi(x)) = 1$, $\gcd(E_2, \phi(y)) = 1$, $\gcd(E_1 \cdot E_2, \phi(N)) = 1$, and $M < N$. The XOR operation serves as an obfuscation layer: since $A \oplus N \oplus N = A$, the effective public/private exponent pair used in modular exponentiation is (E', D'), which satisfies $E' \cdot D' \equiv 1\ (mod\ \phi(N))$ [22].

Parameter Selection. Under DEE-RSA guidelines, XRSA parameters must be justified with cryptanalytic bounds:

•Modulus size: ≥2048 bits total, with each of the four prime factors contributing approximately 512 bits [22].

•Intermediate exponents: E_1 and E_2 must be selected independently such that each is coprime to its respective totient. The product $E_1 \cdot E_2$ must additionally be coprime to

$\phi(N)$.

•Final exponents: The effective public exponent is $E' = (E_1 \cdot E_2) \bmod N$ and the effective private exponent D' satisfies $E' \cdot D' \equiv 1\ (mod\ \phi(N))$. Wiener and Boneh–Durfee bounds must be evaluated with respect to D', not D, since the XOR obfuscation does not alter the effective private exponent size.

•Wiener bound: Ensure $D' > \frac{1}{3}N^{\frac{1}{4}}$ to resist continued fraction attacks [10, 35].

•Boneh–Durfee bound: Ensure $D' > N^{0.292}$ for lattice attack resistance [25, 32].

•Fermat condition: For all prime pairs (p_i, p_j), $|p_i - p_j| > 2N^{\frac{1}{4}}$ to prevent Fermat factorization [2].

•ECM vulnerability: Four 512-bit primes are individually smaller than the two 1024-bit primes of classical RSA, increasing vulnerability to Elliptic Curve Method factoring [9, 14]. This trade-off must be explicitly acknowledged.

Key Generation Design. The key generation procedure must specify:

•Entropy source for prime generation (e.g., os.urandom or hardware random number generator (RNG) [27]).

•Miller–Rabin primality testing with at least 40 rounds for 512-bit candidate primes [26].

•Rejection loops for $gcd(E_1, \phi(x)) = 1$, $gcd(E_2, \phi(y)) = 1$, and $gcd(E_1 \cdot E_2, \phi(N)) = 1$ with documented worst-case behavior.

•Explicit documentation of the mapping from intermediate exponents (E', D') to final keys (E, D) via XOR with N.

Phase I Output: A Candidate XRSA Variant specification document containing all algorithms, parameter bounds, and design rationale.

4.3.2 Phase II: Evaluation—Cryptanalyzing XRSA under DEE-RSA

Correctness Verification. The correctness of XRSA decryption relies on the XOR-cancellation property and the standard RSA modular exponentiation relationship. The proof follows the structure established by Imam et al. [22]:

Step 1—XOR recovery of intermediate exponents. During encryption, the effective exponent is recovered from the public key:

$$E'' = E \oplus N = (E' \oplus N) \oplus N = E'$$

During decryption, the effective exponent is recovered from the private key:

$$D'' = D \oplus N = (D' \oplus N) \oplus N = D'$$

Both equalities hold because XORing twice with the same value *N* cancels out (bitwise identity: $A \oplus N \oplus N = A$).

Step 2—Decryption inverts encryption. Using the recovered exponents:

$$\begin{aligned} Dec\big(sk, Enc(pk, M)\big) &= Dec(sk, C) = C^{D''}\ mod\ N \\ &= C^{D'}\ mod\ N = (M^{E''})^{D'}\ mod\ N \\ &= (M^{E'})^{D'}\ mod\ N = M^{E' \cdot D'}\ mod\ N \end{aligned}$$

Since $E' \cdot D' \equiv 1\ (mod\ \phi(N))$ by construction, Euler's theorem yields $M^{E' \cdot D'} \equiv M^1 \equiv M\ (mod\ N)$ for all M with $gcd(M, N) = 1$. For the general case where $gcd(M, N) \neq 1$, the Chinese Remainder Theorem extends correctness to all $M \in \mathbb{Z}_N$ [23].

This proof differs fundamentally from standard CRT-accelerated RSA decryption. XRSA does not employ CRT during decryption—the correctness relies on the XOR obfuscation layer mapping (E, D) back to the valid RSA key pair (E', D'), after which standard RSA decryption applies directly [22].

Security Assessment. XRSA's security must be formalized through explicit hardness assumptions that account for the layered exponent structure and XOR obfuscation:

•RSA Assumption: Given $(N, E'', C = M^{E''}\ mod\ N)$, recovering M is computationally infeasible without knowledge of D''. Since $E'' = E \oplus N = E'$ and $D'' = D \oplus N = D'$, the attacker must either factor N to compute $\phi(N)$ and derive D', or invert RSA modulo N directly. The XOR obfuscation does not alter the underlying RSA hardness: an adversary who obtains the public key (E, N) can trivially recover $E' = E \oplus N$ and then faces the standard RSA problem with public exponent E'.

•Factoring Assumption: The 4-prime composite N must resist all known factoring algorithms. While NFS complexity depends on the total modulus size (unchanged), ECM complexity depends on the size of the smallest prime factor—reduced from approximately 1024 bits to approximately 512 bits [9, 14]. Imam et al. [22] acknowledged this trade-off, noting that the brute-force attack resistance gained through the XOR-layered exponent structure compensates for the increased ECM vulnerability.

•Security Level: The variant provides OW-CPA (one-way under chosen plaintext attack) security—identical to textbook RSA—as the encryption function $C = M^{E \oplus N}\ mod\ N$ remains deterministic. The XOR obfuscation of the exponent does not add semantic security; it only introduces an additional layer of algebraic complexity that increases brute-force search cost [22].

•Key Recovery Resistance: Imam et al. [22] claimed that knowledge of the public key (E, N) does not directly reveal D' because E is the XOR-obfuscated version of E', and D is the XOR-obfuscated version of D'. However, since $E' = E \oplus N$ is trivially computable from public information, the XOR layer does not provide cryptographic hiding—it serves as a deterrent against naive key recovery rather than a mathematically hard obfuscation.

Cryptanalysis Checklist Application. Table 3 presents the completed 8-point DEE-RSA cryptanalysis checklist applied to XRSA.

The checklist reveals that XRSA, as originally published [22], provided security evaluation solely through brute-force attack time comparison against 4–8 bit primes—addressing none of the eight defined attack categories.

Notably, the XOR obfuscation layer $E = E' \oplus N$, $D = D' \oplus N$ does not fundamentally alter the underlying RSA security model: an adversary can trivially recover $E' = E \oplus N$ from public information and then apply standard RSA attacks against the effective exponent pair (E', D'). Under DEE-RSA, items 1–2 can be addressed through parameter constraints (enforcing lower bounds on D' rather than D), while items 3–8 require additional cryptanalytic analysis and, in several cases, implementation-level countermeasures. The XOR obfuscation does not protect against mathematical attacks (Wiener, Boneh–Durfee, NFS/ECM) since the effective exponents are trivially recoverable [22].

Phase II Output: An XRSA Evaluation Report documenting conditional resistance to the small-private-exponent categories,

partial concern for factorization because of smaller prime factors, and open implementation evidence for timing, chosen-ciphertext, partial-exposure, fault, and side-channel categories. This output does not claim XRSA as fully evaluated.

4.3.3 Phase III: Experimentation—Benchmarking XRSA under DEE-RSA

Benchmarking protocol. We implemented Classical RSA, three-prime RSA, Rebalanced RSA, four-prime XRSA, and Takagi p^{2q} RSA in Python and executed the same harness for every variant and modulus size. Each trial generated fresh key material, verified decrypt(encrypt(m)) = m, and then recorded all required measurements.

(1) Trial count: 100 independent fresh-key trials were executed for every variant at every modulus size (1,500 complete trials).

(2) Key-size range: 1024 bits was retained as a legacy comparison point, while 2048 and 4096 bits evaluate contemporary scaling behavior.

(3) Metric completeness: key generation, encryption, decryption, decryption throughput, and peak traced Python memory were recorded for each trial.

(4) Statistical reporting: Table 4 summarizes the remedied XRSA gaps; Table 5 reports the mean, sample standard deviation, and two-sided 95% confidence-interval half-width for every metric and combination.

(5) Environment documentation: AMD Ryzen 7 5800H with Radeon Graphics; 31.86 GiB RAM; Windows-11-10.0.26200-SP0; Python 3.12.10; cryptography 49.0.0; OpenSSL 4.0.1 9 Jun 2026. Each timed encryption and decryption value is the mean of five repetitions within a trial.

Comparative Implementations:

•Classical RSA uses two primes and full private-exponent decryption, providing the non-CRT baseline [1].

•Multi-Prime RSA, Rebalanced RSA, and Takagi RSA represent distinct-prime CRT, short private CRT exponents, and repeated-prime-power acceleration [9, 10, 14].

•XRSA represents the recent four-prime construction and performs full-width exponentiation after reversible XOR recovery [22].

Compliance and scope. The current experiment remedies the missing metrics, uncertainty estimates, and key-size range. It is a comparative reference implementation, not a constant-time production library or a claim of universal performance.

Phase III Output: The revised manuscript now includes the complete numerical summary, a multi-size figure, raw per-trial CSV data, aggregate JSON/CSV files, the executable benchmark, and an environment record.

The case study demonstrates how DEE-RSA separates evidence from claims. For XRSA, reversible XOR does not change the underlying RSA hardness, the four-prime structure creates a smaller-factor trade-off, and implementation-level resistance remains open until measured. The revised output is therefore an evaluation report with qualified findings.

Table 3. DEE-RSA cryptanalysis checklist applied to XRSA

#	Attack	XRSA Status	Required Mitigation
1	Wiener's Continued Fraction	Resistant if $D' > \frac{1}{3}N^{\frac{1}{4}}$ is enforced during key generation; otherwise vulnerable [10, 35]	Hard bound on D' during keygen
2	Boneh–Durfee Lattice	Resistant if $D' > N^{0.292}$; the 4-prime structure does not increase vulnerability [25, 32]	Hard bound on D' during keygen
3	Factorization (NFS/ECM)	Partially vulnerable — ECM more effective against smaller primes (512-bit vs. 1024-bit) [14, 28]	Increase prime sizes or accept trade-off
4	Timing Attacks	Not evaluated by Imam et al. [22]; standard modular exponentiation $M = C^{D \oplus N} \bmod N$ is susceptible [39, 40]	Constant-time exponentiation or blinding
5	Chosen Ciphertext Attacks	Not evaluated; deterministic encryption implies IND-CPA insecurity [41, 42]	Use hybrid encryption or add padding
6	Partial Key Exposure	Not evaluated; lattice methods may exploit partial d or p_i information [44]	Assess with $\left\lceil \frac{n}{4} \right\rceil$ bits of d
7	Fault Injection	Not evaluated; standard modular exponentiation using XOR-recovered exponent D'' remains vulnerable to fault injection [45, 46]	Apply CRT fault countermeasures
8	Side-Channel Attacks	Not evaluated; the XOR recovery step (E ⊕ N) and subsequent modular exponentiation create potential leakage points [48, 49]	Assess with differential power analysis (DPA) and correlation power analysis (CPA) methodology

Note: Rivest-Shamir-Adleman (RSA); Design, Evaluation, and Experimentation of RSA variants (DEE-RSA); XOR-based four-prime RSA variant (XRSA).

Table 4. Original XRSA evaluation and current-study response to DEE-RSA requirements

DEE-RSA Requirement	Original XRSA [22]	Current Study Response
Formal correctness proof	Partial (algorithmic)	Reassessed; proof scope and assumptions stated
Eight-category cryptanalysis	Brute-force comparison only	Completed as an evidence-qualified checklist
100 trials per metric	Single-run reporting	Yes; 100 trials at each of three key sizes
95% confidence intervals	Not reported	Yes; mean, SD, and 95% CI reported
Throughput metric	Not reported	Yes; decryption operations per second
Memory-usage metric	Not reported	Yes; peak traced Python allocation, with scope disclosed
Hardware specification	CPU/RAM only	Yes; CPU, RAM, OS, and architecture recorded
Software environment	SageMath version unstated	Yes; Python, cryptography, and OpenSSL versions recorded
Source code and raw data	Not provided	Yes; benchmark code and raw CSV/JSON retained
Randomness documentation	Not provided	OS-seeded OpenSSL cryptographically secure pseudorandom number generator (CSPRNG) documented; no deterministic seed used for this benchmark

Note: Rivest-Shamir-Adleman (RSA); Design, Evaluation, and Experimentation of RSA variants (DEE-RSA); XOR-based four-prime RSA variant (XRSA).

Table 5. Complete multi-key-size benchmark results (100 trials per variant and size)

Variant	Bits	Key Generation (ms)	Encryption (ms)	Decryption (ms)	Throughput (ops/s)	Peak Memory (KiB)
Classical RSA	1024	12.82 ± 3.55 (CI ± 0.70)	0.077 ± 0.023 (CI ± 0.004)	6.30 ± 0.78 (CI ± 0.15)	160.94 ± 18.75 (CI ± 3.72)	4.40 ± 0.04 (CI ± 0.01)
Classical RSA	2048	90.07 ± 34.39 (CI ± 6.82)	0.185 ± 0.064 (CI ± 0.013)	34.54 ± 3.51 (CI ± 0.70)	29.24 ± 2.81 (CI ± 0.56)	7.70 ± 0.00 (CI ± 0.00)
Classical RSA	4096	786.18 ± 332.54 (CI ± 65.98)	0.520 ± 0.067 (CI ± 0.013)	224.79 ± 15.43 (CI ± 3.06)	4.47 ± 0.30 (CI ± 0.06)	14.97 ± 0.00 (CI ± 0.00)
Multi-Prime RSA (3)	1024	11.37 ± 2.52 (CI ± 0.50)	0.075 ± 0.018 (CI ± 0.004)	5.12 ± 0.92 (CI ± 0.18)	201.07 ± 32.11 (CI ± 6.37)	3.54 ± 0.05 (CI ± 0.01)
Multi-Prime RSA (3)	2048	67.27 ± 16.03 (CI ± 3.18)	0.183 ± 0.036 (CI ± 0.007)	13.95 ± 2.42 (CI ± 0.48)	73.71 ± 12.02 (CI ± 2.39)	5.48 ± 0.00 (CI ± 0.00)
Multi-Prime RSA (3)	4096	572.34 ± 134.06 (CI ± 26.60)	0.546 ± 0.270 (CI ± 0.053)	46.08 ± 5.20 (CI ± 1.03)	21.96 ± 2.38 (CI ± 0.47)	9.50 ± 0.00 (CI ± 0.00)
Rebalanced RSA	1024	25.24 ± 15.02 (CI ± 2.98)	5.946 ± 0.598 (CI ± 0.119)	1.96 ± 0.31 (CI ± 0.06)	519.84 ± 67.23 (CI ± 13.34)	4.67 ± 0.04 (CI ± 0.01)
Rebalanced RSA	2048	200.75 ± 138.20 (CI ± 27.42)	32.284 ± 4.085 (CI ± 0.810)	3.96 ± 0.84 (CI ± 0.17)	261.81 ± 45.80 (CI ± 9.09)	7.71 ± 0.00 (CI ± 0.00)
Rebalanced RSA	4096	1755.42 ± 1710.90 (CI ± 339.44)	206.908 ± 11.858 (CI ± 2.353)	9.32 ± 1.32 (CI ± 0.26)	109.28 ± 14.37 (CI ± 2.85)	13.10 ± 0.00 (CI ± 0.00)
XRSA (4-Prime)	1024	21.36 ± 3.80 (CI ± 0.75)	6.295 ± 0.641 (CI ± 0.127)	6.55 ± 0.86 (CI ± 0.17)	154.84 ± 17.51 (CI ± 3.47)	5.06 ± 0.05 (CI ± 0.01)
XRSA (4-Prime)	2048	114.46 ± 25.98 (CI ± 5.15)	34.102 ± 3.071 (CI ± 0.609)	35.38 ± 3.01 (CI ± 0.60)	28.48 ± 2.53 (CI ± 0.50)	8.90 ± 0.00 (CI ± 0.00)
XRSA (4-Prime)	4096	951.23 ± 398.46 (CI ± 79.05)	202.734 ± 10.847 (CI ± 2.152)	218.99 ± 11.82 (CI ± 2.34)	4.58 ± 0.24 (CI ± 0.05)	17.23 ± 0.00 (CI ± 0.00)
Takagi RSA (p^2q)	1024	7.51 ± 1.69 (CI ± 0.33)	0.075 ± 0.018 (CI ± 0.004)	3.99 ± 0.59 (CI ± 0.12)	254.67 ± 29.94 (CI ± 5.94)	2.91 ± 0.05 (CI ± 0.01)
Takagi RSA (p^2q)	2048	44.72 ± 14.51 (CI ± 2.88)	0.189 ± 0.041 (CI ± 0.008)	11.32 ± 1.76 (CI ± 0.35)	90.55 ± 14.64 (CI ± 2.90)	4.54 ± 0.00 (CI ± 0.00)
Takagi RSA (p^2q)	4096	383.27 ± 123.64 (CI ± 24.53)	0.520 ± 0.095 (CI ± 0.019)	31.84 ± 2.62 (CI ± 0.52)	31.60 ± 2.44 (CI ± 0.48)	7.94 ± 0.00 (CI ± 0.00)

Note: Entries are mean ± standard deviation, with 95% confidence interval half-width in parentheses. Throughput is defined as completed decryptions per second. Peak memory is the maximum Python-managed allocation observed by tracemalloc during the complete trial; native OpenSSL allocations are excluded. Rivest-Shamir-Adleman (RSA).

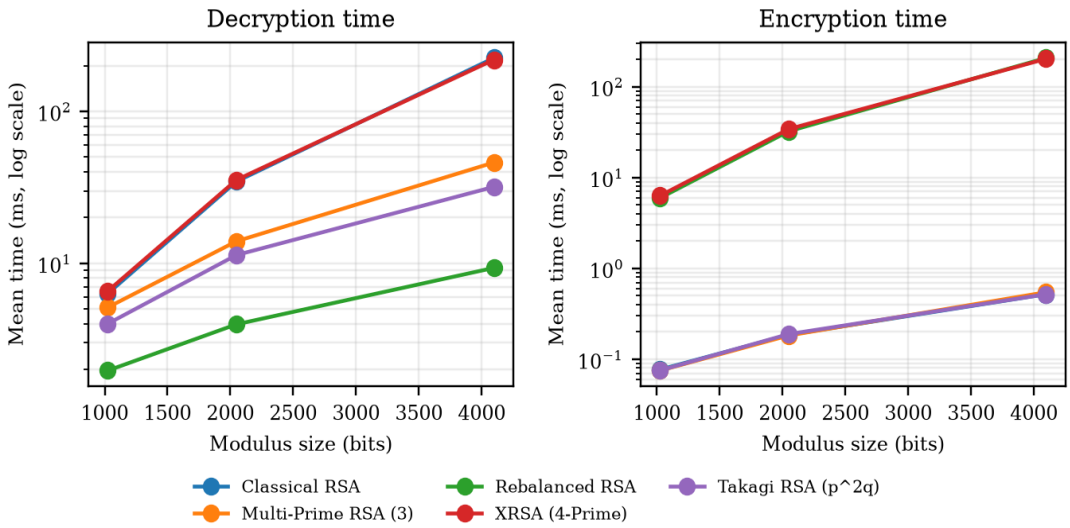


Figure 1. Mean encryption and decryption time across 100 trials at 1024, 2048, and 4096 bits (logarithmic time scale)

4.4 Benchmarking results

As a practical demonstration of Phase III, five implementations were evaluated at 1024, 2048, and 4096 bits using 100 independent fresh-key trials for every variant–size combination. Table 5 reports all five required metrics with mean, standard deviation, and 95% confidence-interval half-widths. The raw per-trial data and environment record are retained with the reproducibility package.

The multi-size results show that the ranking is operation-dependent and changes in magnitude as the modulus grows. The lowest mean decryption time was obtained by Rebalanced RSA at 1024 bits, Rebalanced RSA at 2048 bits, and Rebalanced RSA at 4096 bits. Rebalanced RSA consistently shifts cost toward encryption, while XRSA performs full-width exponentiation without CRT acceleration. These observations are implementation-specific rather than universal algorithmic rankings; they support the framework requirement to report multiple operations, uncertainty, and key sizes. Figure 1 visualizes these timing results.

5. REPRODUCIBILITY PACKAGE

In alignment with established reproducible research practices in computational science [16, 19], DEE-RSA includes a structured reproducibility package. The package is designed to enable independent researchers to replicate every computational and experimental result reported in an RSA variant study.

All materials comprising the DEE-RSA reproducibility package are publicly available at: https://github.com/dianrachmawatiE/dee-rsa-framework

5.1 Hardware specifications

Every benchmark report must document the complete hardware configuration:

•CPU model, architecture, base clock frequency, and number of physical cores.

•Total RAM capacity, type (DDR4/DDR5), and speed.

•Operating system name, version, and build number.

•Any relevant hardware acceleration features (e.g., AES-NI, AVX-512).

5.2 Software environment

The complete software stack must be specified with exact version numbers:

•Programming language and version (e.g., Python 3.12, Java 17, C++ with GCC 13.2).

•Cryptographic library and version (e.g., OpenSSL 3.1, Bouncy Castle 1.77, SageMath 10.2).

•All dependencies with explicit version identifiers, ideally through a package specification file (requirements.txt, Cargo. toml, CMakeLists.txt) [17].

•Compiler/interpreter flags used, if any non-default settings are employed.

5.3 Parameter settings

All configurable parameters must be documented:

•Key sizes tested (e.g., 1024, 2048, 4096 bits).

•Public exponent value(s) used.

•Number of independent trials per measurement.

•Randomness and seeding policy (provide a deterministic seed only when deterministic reproduction is intended; otherwise document the OS-seeded CSPRNG source).

•Primality testing parameters (e.g., Miller–Rabin round count).

5.4 Source code availability

Following the recommendations of Olszewski et al. [17] and Costa et al. [19]:

•Complete source code for key generation, encryption, decryption, and benchmarking must be deposited in a public, persistent repository (e.g., GitHub, Zenodo, OSF).

•Repository must include a README.md with step-by-step instructions, including exact commands for installation and execution [17].

•A requirements file with explicit version control must accompany every repository [17].

•Where feasible, Docker images or virtual machine images should be provided to mitigate dependency conflicts and ensure consistent execution across systems [19, 20].

5.5 Reproducibility checklist

Table 6 presents the DEE-RSA reproducibility checklist, synthesized from the ten principles of the SciRep framework [19], the ACM artifact evaluation badging criteria [16], and the five recommendations of Olszewski et al. [17].

This checklist operationalizes the definition of computational reproducibility articulated by the National Academies of Science and adopted by the computer security community: "obtaining consistent results using the same data and code" [16]. Following DEE-RSA, every RSA variant publication should satisfy all checklist items.

Table 6. DEE-RSA reproducibility checklist

#	Checklist Item
1	Source code is publicly available in a persistent repository
2	README.md provides step-by-step execution instructions
3	All dependencies are listed with explicit version numbers
4	Parameter settings are completely documented
5	Hardware specifications are reported
6	Software environment (OS, compiler, libraries) is fully specified
7	Randomness source and seeding policy are documented; deterministic seeds are provided only when used
8	Raw experimental data (CSV/JSON) is included in the repository
9	Output from running the code matches tables/figures in the paper
10	Docker container or VM image is provided (recommended)

Note: Design, Evaluation, and Experimentation of RSA variants (DEE-RSA); Rivest-Shamir-Adleman (RSA).

6. DEE-RSA USAGE GUIDE AND WORKFLOW

This section provides a step-by-step practical guide for researchers to apply the DEE-RSA framework when developing and evaluating a new RSA variant. Figure 2 presents a detailed decision-flow diagram that operationalizes the three-phase framework into an actionable workflow.

6.1 Step-by-step usage instructions

The following step-by-step instructions operationalize the workflow diagram (Figure 2) for researchers seeking to develop and evaluate an RSA variant using DEE-RSA:

Step 1 — Define Motivation and Problem Statement.

Articulate the specific limitation of existing RSA constructions that the proposed variant addresses. Reference the classification framework of Imam et al. [6] to position the variant within established RSA research domains. Document the design objective (e.g., faster decryption, ciphertext size reduction, improved side-channel resistance).

Step 2 — Write Mathematical Formulation.

Specify the complete mathematical definition including: modulus structure ($n = pq$, $n = p^k q$, $n = \prod p_i$), key relationship ($ed \equiv 1\ (mod\ \phi(n))$ or $(mod\ \lambda(n))$), encryption function, and decryption function [1, 23]. State all algebraic assumptions explicitly (e.g., $\gcd(e, \phi(n)) = 1$).

Step 3 — Specify Parameter Bounds.

Define concrete bounds for all parameters, justified by cryptanalytic results: modulus size (≥ 2048 bits), public exponent ($e = 65537$), Wiener bound ($d > \frac{1}{3}N^{\frac{1}{4}}$), Boneh–Durfee bound ($d > N^{0.292}$), and Fermat condition ($|p - q| > 2N^{\frac{1}{4}}$) [2, 25].

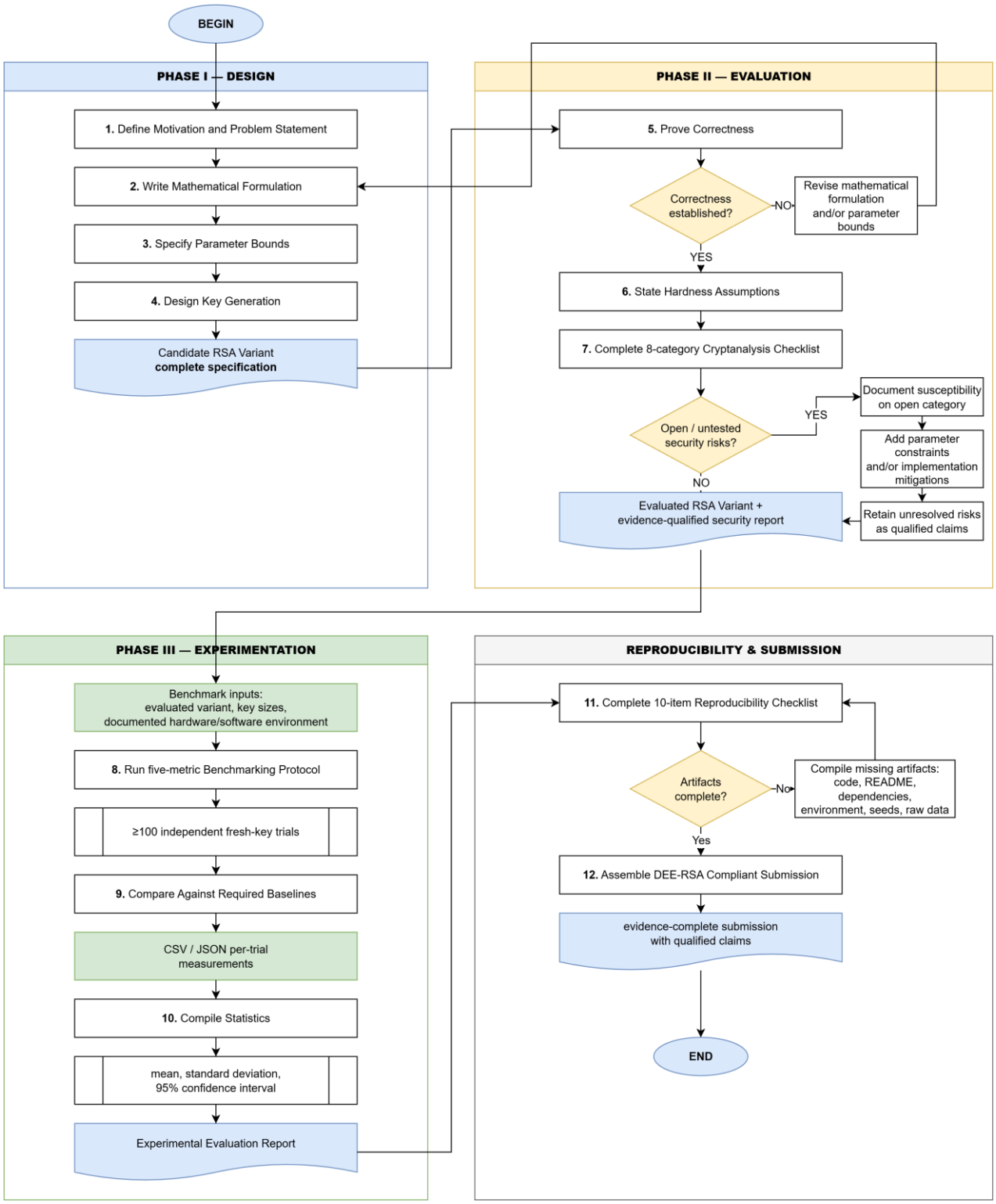


Figure 2. Design, Evaluation, and Experimentation of RSA variants (DEE-RSA) workflow from design through evaluation and experimentation, with evidence-qualified outputs and explicit open-risk reporting

Note: Rivest-Shamir-Adleman (RSA).

Step 4 — Design Key Generation.

Specify the complete key generation procedure including entropy source [27], primality testing method with round count [26], and CRT pre-computation where applicable.

Step 5 — Prove Correctness.

Provide a formal proof that decryption inverts encryption for all valid messages. For non-standard modulus structures, explicitly address the group-theoretic properties that guarantee correctness.

Step 6 — State Hardness Assumptions.

Formalize the security assumptions: RSA Assumption, Factoring Assumption, and achieved security properties (IND-CPA, IND-CCA2, or OW-CPA). Reference standard definitions from Goldwasser and Micali [29] and Rackoff and Simon [30].

Step 7 — Complete Cryptanalysis Checklist.

Evaluate the variant against all eight attack categories in the DEE-RSA checklist (Section 3.2). For each attack, document whether the variant is resistant, vulnerable, or not yet evaluated, with appropriate citations and justification. Attacks

that cannot be definitively resolved at the specification level should be acknowledged with recommended implementation mitigations.

Step 8 — Run Benchmarking Protocol.

Execute the five-metric benchmarking protocol (key generation time, encryption time, decryption time, throughput, memory usage) on a documented hardware and software platform. Conduct a minimum of 100 independent trials per metric with fresh key generation each trial. Record all raw data in CSV or JSON format.

Step 9 — Compare Against Baselines.

Benchmark the proposed variant alongside: (a) classical (textbook) RSA as baseline [1], (b) at least one established fast variant (e.g., CRT-RSA [7] or Multi-Prime RSA [8]), and (c) at least one recent variant from the literature.

Step 10 — Compute Statistics.

Report mean (μ), standard deviation (σ), and 95% confidence intervals ($\mu \pm 1.96 \cdot \frac{\sigma}{\sqrt{n}}$) for each metric, following the statistical reporting standards in Section 3.3.

Step 11 — Complete Reproducibility Checklist.

Verify all 10 items of the DEE-RSA reproducibility checklist (Table 6), including source code publication, dependency specification, hardware/software documentation, the documented randomness and seeding policy, and raw data inclusion. Provide a Docker image or VM image where feasible [19, 20].

Step 12 — Prepare DEE-RSA Compliant Submission.

Assemble the complete DEE-RSA package: (i) candidate variant specification (Phase I output), (ii) cryptanalysis report (Phase II output), (iii) experimental evaluation report (Phase III output), and (iv) reproducibility package.

6.2 Templates and checklists

To facilitate adoption, DEE-RSA provides ready-to-use supporting artifacts that accompany this publication:

•DEE-RSA Checklist (checklist_dee_rsa.md) — A comprehensive 30-item Markdown checklist covering all three phases, the 8-point cryptanalysis evaluation, the 5-metric benchmarking protocol, and the 10-item reproducibility criteria. Researchers can use this checklist during development and reviewers can use it during manuscript evaluation.

•Benchmarking Template (template_benchmarking.md) — A pre-formatted template for recording hardware specifications, software environment, parameter settings, measurement procedures, and statistical results in tabular form, aligned with the DEE-RSA Phase III requirements.

•Reporting Template (template_reporting.md) — A section-level template for organizing RSA variant manuscripts in DEE-RSA-compliant format, including placeholders for motivation, mathematical formulation, correctness proof, cryptanalysis results, benchmarking data, and reproducibility documentation.

All templates and checklists are available at https://github.com/dianrachmawatiE/dee-rsa-framework and can be freely adapted by researchers for their specific variant requirements. These artifacts operationalize the principles articulated by Damaj and Kasbah [18] regarding the necessity of "combining multiple measurements and characteristics in unified indicators" and by Costa et al. [19] regarding the importance of structured templates for computational reproducibility.

7. DISCUSSION

7.1 Comparison of conventional vs. DEE-RSA evaluation approaches

To clarify the advantages of adopting the DEE-RSA framework, Table 7 contrasts the conventional approach to RSA variant evaluation—as documented by Imam et al. [6] and evidenced in the gap analysis of Table 1—with the structured methodology prescribed by DEE-RSA.

Table 7. Comparison of conventional RSA variant evaluation practices and DEE-RSA framework

Evaluation Dimension	Conventional Practice	DEE-RSA Framework
Security Assessment	Informal claims of factoring resistance; no systematic attack evaluation [6]	Structured 8-point cryptanalysis checklist derived from Boneh [2] and Nitaj [32], covering Wiener, Boneh–Durfee, NFS/ECM, timing, CCA, partial key exposure, fault injection, and side-channel attacks
Correctness Verification	Often implicit or omitted for non-standard modulus structures [14]	Mandatory formal proof that Dec(sk, Enc(pk, m)) = m for all valid messages, with explicit treatment of CRT recombination
Benchmarking Protocol	Ad hoc execution time measurement; inconsistent metrics; small trial counts (often single runs) [12, 22]	Five mandatory metrics (keygen, encrypt, decrypt, throughput, memory) with minimum 100 independent trials, mean ± std. dev., and 95% confidence intervals [15, 18]
Comparative Baselines	Often absent or compared against only textbook RSA [6]	Mandatory comparison against textbook RSA baseline plus at least one established fast variant and one recent variant from the literature
Statistical Reporting	Raw numbers without statistical analysis [22]	Complete descriptive and inferential statistics: μ, σ, 95% CI $\mu \pm 1.96 \cdot \frac{\sigma}{\sqrt{n}}$
Reproducibility	Source code rarely provided; insufficient environment documentation; only 3.9% of applied security papers produce reproducible results [16, 17]	10-item reproducibility checklist (source code, README, dependencies, parameters, hardware, software, randomness and seeding policy, raw data, output matching, container image) adapted from SciRep [19] and ACM badging criteria [16]
Reporting Completeness	Inconsistent coverage; critical dimensions often omitted [6]	Structured reporting template covering all three phases with explicit sub-phase outputs (Candidate Variant, Evaluated Variant, Experimental Evaluation Report)
Reviewer Guidance	No structured evaluation instrument for assessing submission completeness	Objective checklist-based review instrument enabling systematic assessment of RSA variant submissions

Note: Rivest-Shamir-Adleman (RSA); Design, Evaluation, and Experimentation of RSA variants (DEE-RSA).

As shown in Table 7, the DEE-RSA framework addresses each dimension that is inconsistently covered or entirely absent in conventional RSA variant evaluation practices. The most pronounced improvements are in security assessment (from informal claims to a structured 8-point checklist), statistical rigor (from raw numbers to 95% confidence intervals with 100 trials), and reproducibility (from rare code availability to a mandatory 10-item checklist).

7.2 Strengths of the DEE-RSA framework

The DEE-RSA framework provides several advantages over current ad hoc evaluation practices in RSA variant research:

Consistency. By structuring the evaluation dimensions, metrics, and reporting format, DEE-RSA enables meaningful cross-study comparison that is currently absent from the literature. Imam et al. [6] identified the need for "a consistent taxonomy to align various concepts such as execution time, security, and performance," which the DEE-RSA framework directly addresses through its structured three-phase methodology.

Completeness. The cryptanalysis checklist derived from Boneh's taxonomy [2] and Nitaj's constrained key framework [32] ensures that no major attack class is overlooked during variant evaluation. The gap analysis (Table 1) demonstrates that existing publications systematically omit entire evaluation dimensions—particularly timing attacks, chosen ciphertext attacks, and fault injection analysis—that the DEE-RSA checklist mandates.

Reproducibility. The reproducibility package, grounded in established frameworks [16, 17, 19], addresses the systemic deficiencies documented in the security literature. Only 10% of applied security research provides runnable artifacts [16], and 60% of security papers lack accompanying code [17]. DEE-RSA's mandatory reproducibility checklist directly targets this gap.

Reviewer utility. The framework provides a structured, evidence-traceable instrument for checking whether RSA variant submissions report the expected dimensions. It supports consistent review questions but does not replace expert judgment, cryptographic certification, or venue-specific policy.

Facilitation of Comparison. The benchmarking protocol specifies uniform metrics (key generation time, encryption/decryption time, throughput, memory usage), statistical reporting standards (mean, standard deviation, 95% confidence intervals, minimum trial counts), and comparative baselines, enabling direct numerical comparison between variants evaluated under DEE-RSA.

7.3 Limitations

Several limitations of the current DEE-RSA proposal should be acknowledged. These limitations define the boundary of the claims made in this paper and motivate independent follow-up work:

Domain Scope. DEE-RSA is designed specifically for the RSA cryptosystem family. While the general three-phase structure (Design–Evaluation–Experimentation) may be adaptable to other public-key families, the cryptanalysis checklist is tailored to RSA-specific attacks. Adaptation to elliptic curve cryptography (ECC), lattice-based cryptography, or other post-quantum schemes would require domain-specific attack checklists.

Computational Cost. The framework mandates extensive benchmarking (minimum 100 trials across multiple key sizes) and comprehensive cryptanalysis, which increases the computational burden of variant evaluation. However, this cost is justified by the resulting increase in confidence, comparability, and reproducibility.

Methodological status and rating reliability. DEE-RSA is an author-synthesized research framework, not an externally assessed or community-structured methodology. The retrospective ratings were resolved within the author team without blinded independent coding or an inter-rater coefficient. External expert review, preregistered source selection, and independent replication are required before stronger acceptance claims can be made.

Benchmark implementation and memory scope. The benchmark uses transparent Python reference implementations rather than constant-time production cryptographic code. Peak memory is measured with tracemalloc and therefore excludes native OpenSSL allocations. Results support within-harness comparison and scaling analysis, but should not be generalized to all libraries, processors, compilers, or deployment environments.

Evolving Attack Landscape. The cryptanalysis checklist represents the state of knowledge at the time of publication. As new attacks are discovered—as demonstrated by the emergence of cache-timing attacks [40], thermal side-channel attacks [48], and fault injection systematization [46]—the checklist must be updated. DEE-RSA should be treated as a living document that evolves with the cryptanalytic literature.

Implementation vs. Specification. The DEE-RSA framework evaluates RSA variants at the specification level. While it includes implementation-level attacks (timing, side-channel, fault injection) in the cryptanalysis checklist, it does not prescribe specific implementation-level countermeasures, which remain the responsibility of the implementer [39, 49].

7.4 Future work

Further work should evaluate strengthened RSA key-generation schemes [50] and post-quantum extensions [51]:

Adaptation to ECC. The three-phase structure could be adapted to standardize the evaluation of elliptic curve cryptography variants, with a new cryptanalysis checklist addressing ECC-specific attacks (e.g., Pollard's rho, MOV attack, invalid curve attacks).

Adaptation to Lattice-Based Cryptography. As post-quantum cryptography transitions to standardization, a DEE framework for lattice-based schemes (DEE-Lattice) could provide structured evaluation criteria for the emerging generation of post-quantum algorithms [15].

Automated Tooling. Developing software tools that automatically execute the DEE-RSA benchmarking protocol and generate the required statistical reports would further reduce barriers to adoption. Integration with SUPERCOP [15] or similar benchmarking frameworks [18] would enhance usability.

Community Adoption. The long-term impact of DEE-RSA depends on adoption by the RSA research community, conference program committees, and journal reviewers. Establishing DEE-RSA compliance as a review criterion at cryptography venues would support broader methodological convergence.

8. CONCLUSION

We have presented DEE-RSA, a three-phase structured framework for the Design, Evaluation, and Experimentation of novel RSA variants. Its construction is documented through a traceable 52-source procedure, and its scope is explicitly limited: DEE-RSA is a candidate research and reporting framework rather than an externally assessed standard. It integrates established RSA attack taxonomies, benchmark guidance, and reproducibility practices.

(1) A structured Design phase guiding systematic variant specification with complete mathematical formulation and parameter justification.
(2) A comprehensive Evaluation phase comprising correctness verification, security assessment, and an eight-point cryptanalysis checklist covering Wiener's attack, Boneh–Durfee lattice attack, factorization attacks, timing attacks, chosen ciphertext attacks, partial key exposure attacks, fault injection attacks, and side-channel attacks.
(3) A structured Experimentation phase with a five-metric benchmarking protocol (key generation time, encryption time, decryption time, throughput, memory usage), comparative evaluation requirements, and statistical reporting standards (mean, standard deviation, 95% confidence intervals, minimum 100 trials).
(4) A reproducibility package with hardware/software documentation requirements, source code availability mandates, and a ten-item reproducibility checklist.

Retrospective evaluation across eight representative RSA families illustrated recurring evidence gaps while making the coding rules and assessor limitations explicit. A 1,500-trial benchmark across 1024, 2048, and 4096 bits now demonstrates the five-metric protocol with complete uncertainty reporting. Future work should obtain external expert assessment, independent coding, additional optimized implementations, and community feedback before any standardization claim is considered.

REFERENCES

[1] Rivest, R.L., Shamir, A., Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2): 120-126. https://doi.org/10.1145/359340.359342

[2] Boneh, D. (1999). Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society, 46(2): 203-213. https://www.ams.org/journals/notices/199902/boneh.pdf?trk=199902boneh&cat=collection.

[3] National Institute of Standards and Technology (NIST). (2024). Module-lattice-based key-encapsulation mechanism standard. Federal Information Processing Standards Publication (FIPS) 203. https://doi.org/10.6028/NIST.FIPS.203

[4] Moody, D., Perlner, R., Regenscheid, A., Robinson, A., Cooper, D. (2024). Transition to post-quantum cryptography standards. National Institute of Standards and Technology, NIST IR 8547 Initial Public Draft. https://doi.org/10.6028/NIST.IR.8547.ipd

[5] Campbell, R. (2025). Enterprise migration to post-quantum cryptography: Timeline analysis and strategic frameworks. Computers, 15(1): 9. https://doi.org/10.3390/computers15010009

[6] Imam, R., Areeb, Q.M., Alturki, A., Anwer, F. (2021). Systematic and critical review of RSA-based public-key cryptographic schemes: Past and present status. IEEE Access, 9: 155949-155976. https://doi.org/10.1109/ACCESS.2021.3129224

[7] Quisquater, J.-J., Couvreur, C. (1982). Fast decipherment algorithm for RSA public-key cryptosystem. Electronics Letters, 18(21): 905-907. https://doi.org/10.1049/el:19820617

[8] Collins, T., Hopkins, D., Langford, S., Sabin, M. (1998). Public key cryptographic apparatus and method. U.S. Patent 5,848,159, Compaq Computer Corporation.

[9] Takagi, T. (1998). Fast RSA-type cryptosystem modulo p^kq. In Advances in Cryptology—CRYPTO '98, Lecture Notes in Computer Science, pp. 318-326. https://doi.org/10.1007/BFb0055738

[10] Wiener, M.J. (1990). Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, 36(3): 553-558. https://doi.org/10.1109/18.54902

[11] Fiat, A. (1989). Batch RSA. In Advances in Cryptology—CRYPTO '89, Lecture Notes in Computer Science, pp. 175-185. https://doi.org/10.1007/0-387-34805-0_17

[12] Hansen, K., Larsen, T., Olsen, K. (2009). On the efficiency of fast RSA variants in modern mobile phones. Int. J. Comput. Sci. Inf. Secur., 6(3): 136-140. https://doi.org/10.48550/arXiv.1001.2249

[13] Zheng, M., Kang, H. (2024). Lattice-based cryptanalysis of RSA-type cryptosystems: A bibliometric analysis. Cybersecurity, 7: 74. https://doi.org/10.1186/s42400-024-00289-7

[14] Boneh, D., Shacham, H. (2002). Fast variants of RSA. CryptoBytes, 5(1): 1-9.

[15] Pote, P., Bansode, R. (2025). Performance evaluation of post-quantum cryptography: A comprehensive framework for experimental analysis. Journal of Information Systems Engineering and Management, 10(9):548-556. https://doi.org/10.52783/jisem.v10i9s.1253

[16] Olszewski, D., Lu, A., Crowder, A., et al. (2025). Reproducibility in applied security conferences: An 11-year review on artifacts and evaluation committees. In Proceedings of the 3rd ACM Conference on Reproducibility and Replicability, Vancouver, Canada, pp. 96-107. https://doi.org/10.1145/3736731.3746151

[17] Olszewski, D., Lu, A., Stillman, C., et al. (2023). "Get in Researchers; We're Measuring Reproducibility": A Reproducibility Study of Machine Learning Papers in Tier 1 Security Conferences. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark, pp. 3433-3459. https://doi.org/10.1145/3576915.3623130

[18] Damaj, I., Kasbah, S. (2018). An analysis framework for hardware and software implementations with applications from cryptography. Computers & Electrical Engineering, 69: 572-584. https://doi.org/10.1016/j.compeleceng.2017.06.008

[19] Costa, L., Barbosa, S., Cunha, J. (2026). A framework for supporting the reproducibility of computational experiments in multiple scientific domains. Future Generation Computer Systems, 174: 107924. https://doi.org/10.1016/j.future.2025.107924

[20] Sandve, G.K., Nekrutenko, A., Taylor, J., Hovig, E.

(2013). Ten simple rules for reproducible computational research. PLOS Comput. Biol., 9(10): e1003285. https://doi.org/10.1371/journal.pcbi.1003285

[21] Buhrow, B., Gilbert, B., Haider, C. (2022). Parallel modular multiplication using 512-bit advanced vector instructions: RSA fault-injection countermeasure via interleaved parallel multiplication. Journal of Cryptographic Engineering, 12(1): 95-105. https://doi.org/10.1007/s13389-021-00256-9

[22] Imam, R., Anwer, F., Nadeem, M. (2022). An effective and enhanced RSA based public key encryption scheme (XRSA). International Journal of Information Technology, 14(5): 2645-2656. https://doi.org/10.1007/s41870-022-00993-y

[23] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A. (1996). Handbook of applied cryptography. CRC Press.

[24] Moriarty, K., Kaliski, B., Jonsson, J., Rusch, A. (2016). PKCS #1: RSA cryptography specifications version 2.2. Internet Engineering Task Force, RFC 8017. https://www.rfc-editor.org/info/rfc8017

[25] Boneh, D., Durfee, G. (2000). Cryptanalysis of RSA with private key d less than N^0.292. IEEE Transactions on Information Theory, 46(4): 1339-1349. https://doi.org/10.1109/18.850673

[26] Albrecht, M.R., Massimo, J., Paterson, K.G., Somorovsky, J. (2018). Prime and prejudice: Primality testing under adversarial conditions. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), Toronto, Canada, pp. 281-298. https://doi.org/10.1145/3243734.3243787

[27] Rukhin, A., Soto, J., Nechvatal, J., et al. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology, Special Publication 800-22 Rev. 1a. https://doi.org/10.6028/NIST.SP.800-22r1a

[28] Lenstra, A.K., Lenstra, H.W.Jr., Manasse, M.S., Pollard, J.M. (1990). The number field sieve. In The Development of the Number Field Sieve, Lecture Notes in Mathematics, Springer, pp. 11-42. https://doi.org/10.1007/BFb0091537

[29] Goldwasser, S., Micali, S. (1984). Probabilistic encryption. Journal of Computer and System Sciences, 28(2): 270-299. https://doi.org/10.1016/0022-0000(84)90070-9

[30] Rackoff, C., Simon, D.R. (1992). Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Advances in Cryptology—CRYPTO '91, Lecture Notes in Computer Science, Springer, pp. 433-444. https://doi.org/10.1007/3-540-46766-1_35

[31] Naor, M., Yung, M. (1990). Public-key cryptosystems provably secure against chosen ciphertext attacks. In Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC), Baltimore, Maryland, USA, pp. 427-437.

[32] Nitaj, A. (2009). Cryptanalysis of RSA with constrained keys. International Journal of Number Theory, 5(2): 311-325. https://doi.org/10.1142/S1793042109002122

[33] Verheul, E.R., van Tilborg, H.C.A. (1997). Cryptanalysis of 'less short' RSA secret exponents. Applicable Algebra in Engineering, Communication and Computing, 8(5): 425-435. https://doi.org/10.1007/s002000050082

[34] De Weger, B. (2002). Cryptanalysis of RSA with small prime difference. Applicable Algebra in Engineering, Communication and Computing, 13: 17-28. https://doi.org/10.1007/s002000100088

[35] Bunder, M., Tonien, J. (2017). A new attack on the RSA cryptosystem based on continued fractions. Malaysian Journal of Mathematical Sciences, 11(S): 45-57.

[36] Bahig, H.M., Nassr, D.I., Mahdi, M.A., Bahig, H.M. (2022). Small private exponent attacks on RSA using continued fractions and multicore systems. Symmetry, 14(9): 1897. https://doi.org/10.3390/sym14091897

[37] Blömer, J., May, A. (2001). Low secret exponent RSA revisited. In Cryptography and Lattices—CaLC 2001, Lecture Notes in Computer Science, pp. 4-19. https://doi.org/10.1007/3-540-44670-2_2

[38] Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A. (2012). Mining your Ps and Qs: Detection of widespread weak keys in network devices. In Proceedings of the 21st USENIX Security Symposium, USENIX Association, Bellevue, WA, USA, pp. 205-220.

[39] Kocher, P.C. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Advances in Cryptology—CRYPTO'96, Lecture Notes in Computer Science, Springer, pp. 104-113. https://doi.org/10.1007/3-540-68697-5_9

[40] Aldaya, A.C., García, C.P., Tapia, L.M.A., Brumley, B.B. (2019). Cache-timing attacks on RSA key generation. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(4): 213-242. https://doi.org/10.13154/tches.v2019.i4.213-242

[41] Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Advances in Cryptology—CRYPTO '98, Lecture Notes in Computer Science, pp. 1-12. https://doi.org/10.1007/BFb0055716

[42] Böck, H., Somorovsky, J., Young, C. (2018). Return of Bleichenbacher's oracle threat (ROBOT). In Proceedings of the 27th USENIX Security Symposium, USENIX Association, Baltimore, MD, USA, pp. 817-832.

[43] Meyer, C., Somorovsky, J., Weiss, E., Schwenk, J., Schinzel, S., Tews, E. (2014). Revisiting SSL/TLS implementations: New Bleichenbacher side channels and attacks. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), San Diego, CA, USA, pp. 733-748.

[44] Boneh, D., Durfee, G., Frankel, Y. (1998). An attack on RSA given a small fraction of the private key bits. In Advances in Cryptology—ASIACRYPT '98, Lecture Notes in Computer Science, pp. 25-34. https://doi.org/10.1007/3-540-49649-1_3

[45] Boneh, D., DeMillo, R.A., Lipton, R.J. (1997). On the importance of checking cryptographic protocols for faults. In Advances in Cryptology—EUROCRYPT '97, Lecture Notes in Computer Science, Springer, pp. 37-51. https://doi.org/10.1007/3-540-69053-0_4

[46] Chiu, T., Xiong, W. (2023). SoK: Fault injection attacks on cryptosystems. In Proceedings of the 12th International Workshop on Hardware and Architectural Support for Security and Privacy (HASP), Toronto, Canada, pp. 64-72. https://doi.org/10.1145/3623652.3623671

[47] Erata, F., Chiu, T., Etim, A., et al. (2024). Systematic use of random self-reducibility in cryptographic code against physical attacks. In Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Newark, NJ, USA, pp. 1-9.

https://doi.org/10.1145/3676536.3689920
[48] Aljuffri, A., Zwalua, M., Reinbrecht, C.R.W., Hamdioui, S., Taouil, M. (2021). Applying thermal side-channel attacks on asymmetric cryptography. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(11): 1930-1942. https://doi.org/10.1109/TVLSI.2021.3111407
[49] Socha, P., Miškovský, V., Novotný, M. (2022). A comprehensive survey on the non-invasive passive side-channel analysis. Sensors, 22(21): 8096. https://doi.org/10.3390/s22218096
[50] Thangavel, M., Varalakshmi, P., Murrali, M., Nithya, K. (2015). An enhanced and secured RSA key generation scheme (ESRKGS). Journal of Information Security and Applications, 20: 3-10. https://doi.org/10.1016/j.jisa.2014.10.004
[51] Dam, D.T., Tran, T.H., Hoang, V.P., Pham, C.K., Hoang, T.T. (2023). A survey of post-quantum cryptography: Start of a new race. Cryptography, 7(3): 40. https://doi.org/10.3390/cryptography7030040