INTERNATIONAL JOURNAL OF LATEST TECHNOLOGY IN ENGINEERING, MANAGEMENT & APPLIED SCIENCE (IJLTEMAS)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

The Unseen Engine of Trust A Deep Dive into the RSA Algorithm

Dr Partha Majumdar

Department of Computer Science, Swiss School of Business and Management (SSBM), Geneva, Switzerland

DOI : https://doi.org/10.51583/IJLTEMAS.2025.1412000084

Received: 21 December 2025; Accepted: 27 December 2025; Published: 07 January 2026

ABSTRACT

This analysis offers a thorough overview of the RSA algorithm, a key element of modern digital security that addresses the historic challenge of secure communication over insecure public channels. Its innovation is based on number theory, using a "trapdoor" one-way function reliant on the significant computational difference between multiplying large prime numbers and factoring their product. Paired with modular arithmetic and Euler's Totient Theorem, this enables the practical use of public-key cryptography, where a public key encrypts data that only the matching private key can decrypt. The text highlights RSA's vital role in internet security, especially in securing web connections via SSL/TLS and ensuring authenticity and integrity with digital signatures. It discusses the cybersecurity arms race, stressing the importance of sufficient key lengths and secure practices to fend off classical attacks. The analysis also examines RSA's main vulnerability: the threat from quantum computing. Shor's algorithm can efficiently solve the integer factorisation problem, making RSA obsolete and prompting a global shift to Post-Quantum Cryptography (PQC). Despite its eventual replacement, RSA's legacy as the first framework that enabled trust in the digital world remains a monumental achievement in computer science.

Keywords: RSA algorithm, Public-key cryptography, Integer factorisation, Digital signatures, Quantum threat

INTRODUCTION:

The Paradox of Public Secrecy

In the digital realm, we perform acts of profound trust with a simple click. When we see the small padlock icon in a web browser's address bar, we instinctively trust that our credit card numbers, private messages, and sensitive data are safe from prying eyes. This trust is not based on faith, but on a revolutionary mathematical concept that solved a paradox as old as communication itself: how can two parties, who have never met, establish a secret conversation over a public channel that is being actively monitored by adversaries? For centuries, the answer was that they could not, at least not without first securely exchanging a secret key through a trusted courier or a prior clandestine meeting.

The algorithm that broke this century-old impasse and became the unseen engine of modern digital trust is known as RSA, named for its public inventors, Ron Rivest, Adi Shamir, and Leonard Adleman. Introduced in 1977, RSA was the first practical and widely adopted system to realise the dream of public-key cryptography. This paradigm fundamentally reshaped our ability to create privacy and verify identity in an increasingly connected world. Its impact has been monumental, forming the cryptographic bedrock for everything from the rise of e-commerce and online banking to the security of national intelligence communications.

This report embarks on a comprehensive journey into the world of RSA, structured to align with the principles of "Data, Dharma, and Decision." We will first delve into the **Data**—the elegant and profound principles of number theory that provide the algorithm's unshakeable mathematical foundation. Next, we will explore the **Dharma**—the core principles of asymmetric cryptography and the fascinating dual history of RSA's invention, a story of both secret government research and open academic collaboration. Finally, we will examine the critical **Decisions** that RSA enables and forces upon us, from its practical applications in securing the internet to the urgent, ongoing arms race against new threats, most notably the dawn of quantum computing. This exploration

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

will deconstruct the algorithm from its 18th-century mathematical roots to its potential obsolescence on the quantum horizon, revealing the intricate mechanics of the system that secures our digital lives.

The Elegant Mathematics of Asymmetry (The Data)

The genius of the RSA algorithm lies not in a complex new invention, but in the clever application of mathematical principles that have been understood for centuries. It leverages a profound imbalance in the computational difficulty of certain number-theoretic problems, creating a system where locking a message is easy for anyone, but unlocking it is practically impossible for everyone except the intended recipient. This asymmetry is the source of its power.

The Primacy of Primes and the One-Way Function

At the heart of RSA is the prime number—an integer greater than 1 that is divisible only by 1 and itself. These numbers are the fundamental building blocks of all integers, a concept formalised in the **Fundamental Theorem of Arithmetic**, which states that every integer greater than 1 can be represented as a product of prime numbers in exactly one way (ignoring the order of the factors). This unique factorisation property is a cornerstone of number theory and, as it turns out, of modern cryptography.

The security of RSA is built upon a specific computational task related to primes: integer factorisation. [7] The core principle is a "one-way function," a mathematical operation that is easy to perform in one direction but exceedingly difficult to reverse. In the case of RSA, this function is the multiplication of two very large prime numbers. [6]

The "Easy" Direction: If one chooses two distinct, large prime numbers, let's call them p and q, calculating their product, $n = p * q$, is computationally trivial. Even for primes that are hundreds of digits long, a modern computer can perform this multiplication in a fraction of a second. [1]

The "Hard" Direction: However, if one is given only the product (known as a semiprime), the reverse operation—finding the original prime factors p and q—is a monumentally difficult task for classical computers. The best-known classical algorithms for factoring large numbers take a prohibitively long time, potentially thousands of years, for the key sizes used in modern applications.

This vast and predictable gap in computational effort is not merely a mathematical curiosity; it is the essential security primitive upon which RSA is built. Historically, security was achieved through physical means or shared secrets. The breakthrough of asymmetric cryptography was the realisation that a purely mathematical imbalance could serve as a lock. Security could be derived not from hiding information, but from the inherent computational limits of even the most powerful machines. This function is not perfectly "one-way," because a reverse path does exist. Instead, it is a **trapdoor function**: the reverse path is computationally infeasible *unless* one possesses a special piece of information—the "trapdoor". [1] In RSA, as we will see, that trapdoor is the knowledge of the prime factors themselves.

The World of Modular Arithmetic

The operational environment where RSA's mathematics unfolds is known as modular arithmetic, a system of arithmetic for integers that "wraps around" upon reaching a certain value—the modulus. It is often intuitively explained as "clock arithmetic." On a 12-hour clock, 4 hours after 10 o'clock is not 14 o'clock, but 2 o'clock. In mathematical terms, this is expressed as $10 + 4 \equiv 2 \pmod{12}$. The symbol ≡ denotes congruence, meaning that 14 and 2 have the same remainder when divided by the modulus 12.

All encryption and decryption operations in RSA are modular exponentiations, taking the form of m^e (mod n) or c^d (mod n). This mathematical framework is crucial for two reasons. First, it ensures that the results of cryptographic operations—the ciphertext and the decrypted plaintext—always remain within a manageable, fixed-size range ($from\ 0\ to\ (n-1)$), regardless of how large the intermediate values of the exponentiation become. Second, this finite, cyclical universe of numbers possesses special properties, described by group theory, that

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Euler's Theorem exploits to guarantee that decryption perfectly and reliably reverses encryption.

Euler's Totient Theorem: The Key to the Trapdoor

The mathematical linchpin that makes the RSA trapdoor function work is a theorem from 1763 by Leonhard Euler. To understand it, one must first be familiar with the **totient function**, denoted as ϕ(n) (phi of n).

Euler's totient function counts the number of positive integers less than or equal to *n* that are relatively prime to *n*. Two numbers are relatively prime (or coprime) if their greatest common divisor (GCD) is 1. [12] For example, $\phi(n) = 4$ because only four numbers *(1, 5, 7, 11)* are less than 12 and share no factors with 12 other than 1.

The totient function has several properties that are critical for RSA:

- If *p* is a prime number, then all numbers from *1* to *(p – 1)* are relatively prime to it, so $\phi(p) = (p - 1)$.
- If *m* and *n* are coprime, the function is multiplicative: $\phi(mn) = \phi(m)\,\phi(n)$.

Combining these, for the RSA modulus $n=p*q$, where p and q are distinct primes, the totient is easily calculated as:

$$\phi(n) = \phi(pq) = \phi(p)\,\phi(q) = (p - 1)(q - 1)$$

With this function defined, we can now state **Euler's Theorem** (also known as the Fermat-Euler theorem or Euler's totient theorem). [4] If an integer *m* and the modulus *n* are relatively prime, then raising *m* to the power of the totient of *n* is congruent to 1, modulo *n*. [1]

$$m^{\phi(n)} \equiv 1 \pmod{n}$$

This theorem provides the mathematical proof for why RSA's encryption and decryption are perfect inverses. In RSA, the public exponent *e* and private exponent *d* are specifically chosen to be modular multiplicative inverses with respect to the totient ϕ(n). This means they satisfy the relationship:

$$ed \equiv 1 \pmod{\phi(n)}$$

This congruence implies that there exists some integer k such that $ed = k\phi(n) + 1$. When a ciphertext $c \equiv m^e \pmod n$ is decrypted, the operation is:

$$c^d \equiv (m^e)^d \pmod n \equiv m^{ed} \pmod n$$

Substituting the relationship for *ed*, we get:

$$m^{ed} \pmod n \equiv m^{(k\phi(n) + 1)} \pmod n \equiv m^{k\phi(n)}\, m^1 \pmod n \equiv (m^{\phi(n)})^k\, m \pmod n$$

Applying Euler's Theorem, $m^{\phi(n)}$ becomes 1:

$$(m^{\phi(n)})^k\, m \pmod n \equiv 1^k\, m \pmod n \equiv m \pmod n$$

The original message *m* is recovered. This elegant proof demonstrates that the decryption process will always succeed.

The critical link between all these concepts is that the entire system hinges on the secrecy of the totient, $\phi(n)$. The public key consists of the pair *(n, e)*. To find the corresponding private key *d*, an attacker would need to solve the congruence $ed \equiv 1 \pmod{\phi(n)}$. This is impossible without knowing $\phi(n)$. And to calculate $\phi(n) = (p - 1)(q - 1)$, one must know the secret prime factors *p* and *q*. Therefore, the "trapdoor" that allows the owner to reverse the one-way function easily is precisely the knowledge of the prime factors of *n*. An attacker who can factor *n* can break the system; an attacker who cannot is left with an intractable problem.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

A Secret History: The Invention and Re-Invention of Public-Key Cryptography

The story of RSA is not a single, linear narrative of invention but a remarkable tale of convergent evolution, where the same revolutionary idea was born twice, independently, in two vastly different worlds: the clandestine corridors of a government intelligence agency and the open, collaborative environment of a university.

The Precursors: Diffie, Hellman, and the Public-Key Revolution

For most of history, cryptography operated under a single paradigm: symmetric encryption. This method uses the same secret key to both encrypt and decrypt a message. While effective, it suffers from a fundamental logistical challenge known as the key distribution problem: for two parties to communicate securely, they must first find a secure way to share the secret key. This often required a trusted courier or a face-to-face meeting, a process that was slow, expensive, and impractical for the burgeoning digital age. [1]

In 1976, this ancient paradigm was shattered. In a groundbreaking paper titled "New Directions in Cryptography," researchers Whitfield Diffie and Martin Hellman introduced the world to the theoretical concept of **public-key cryptography**. They envisioned a system with two mathematically related keys: a **public key**, which could be shared openly with the world, and a **private key**, which would be kept secret by its owner. A message encrypted with the public key could only be decrypted by the corresponding private key. This asymmetric approach elegantly solved the key distribution problem. They also conceptualised digital signatures, but a practical implementation of a "trapdoor one-way function" to make it all work remained an open problem. Their paper, however, lit a fire in the academic community and directly inspired the team at MIT that would ultimately give the world RSA.

The Secret Cypher: Clifford Cocks and GCHQ's Classified Discovery

Unbeknownst to the public academic community, the puzzle posed by Diffie and Hellman had already been solved inside the secretive world of British intelligence. In 1973, four years before RSA was publicly announced, a young mathematician named Clifford Cocks at the UK's Government Communications Headquarters (GCHQ) independently invented an identical algorithm.

Cocks had recently joined GCHQ after studying number theory at Cambridge and Oxford. He was presented with a theoretical paper written in 1969 by his colleague James Ellis, which outlined the concept of "non-secret encryption"—what the world would later call public-key cryptography. Ellis had conceived of the possibility but had been unable to find a mathematical function to implement it. Cocks, with his background in number theory, immediately saw a potential solution in prime factorisation and, within hours, developed the algorithm that would later be known as RSA. Shortly after, another GCHQ mathematician, Malcolm Williamson, independently developed a method for key exchange equivalent to what is now known as the Diffie-Hellman protocol.

However, this monumental British achievement was immediately classified as top-secret. At the time, with the high cost of computing power, the invention was seen more as a theoretical curiosity than a practical tool for military use and was never deployed. The work remained hidden from the world for 24 years, finally being declassified and revealed to the public in 1997.

The Public Breakthrough: Rivest, Shamir, and Adleman at MIT

Back in the public sphere, inspired by Diffie and Hellman's 1976 paper, three colleagues at the Massachusetts Institute of Technology (MIT)—computer scientists Ron Rivest and Adi Shamir, and mathematician Leonard Adleman—embarked on a quest to find the elusive trapdoor one-way function. Their process was a rigorous cycle of creation and destruction: Rivest and Shamir would propose candidate functions, and Adleman's role was to find their weaknesses and break them.

For nearly a year, they struggled, trying and discarding 42 different approaches until they began to believe the task was impossible. The breakthrough came in April 1977. After a long day of work followed by a Passover

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Seder, Rivest lay on a couch, unable to sleep, his mind racing through number theory concepts. Suddenly, the idea coalesced. He realised that a function based on modular exponentiation and the difficulty of factoring a large semiprime number could provide the properties they needed. He spent the rest of the night formalising the algorithm, and by morning, the core of the paper was written.

The algorithm was named RSA, using the initials of their surnames in the order they appeared on the paper. [1] They published their work in 1977 and were granted a U.S. patent in 1983. The parallel invention of the same algorithm in two isolated environments is a striking example of convergent discovery. It suggests that the idea's time had come, as the necessary mathematical tools and the pressing need for secure digital communication had finally aligned. The contrasting fates of these two inventions—one locked away by state secrecy, the other shared openly to become a global standard—serve as a powerful illustration of how institutional culture can shape the destiny of a technology.

Deconstructing the RSA Algorithm

At its core, the RSA cryptosystem involves three distinct processes: the initial generation of a public and private key pair, the encryption of a message using the public key, and the decryption of the ciphertext using the private key. While the underlying mathematics is profound, the procedural steps are remarkably straightforward.

Step-by-Step: Key Generation, Encryption, and Decryption

Key Generation

The foundation of RSA security is the careful and random generation of a unique key pair. This process is performed by any individual or system that wishes to receive encrypted messages. The steps are as follows:

Choose two distinct large prime numbers, *p* and *q*. These primes must be chosen at random from a vast space of possibilities and must be kept secret. The security of the entire system rests on the infeasibility of an attacker guessing or deriving these two numbers. For a modern 2048-bit key, each prime would be roughly 1024 bits long.

Compute the modulus $n = p * q$. This product, *n*, serves as the modulus for all cryptographic operations. Its length in bits (e.g., 2048) defines the key size. The value of *n* is made public as part of the public key.

Compute the totient $\phi(n) = (p - 1)(q - 1)$. This value, derived from the secret primes, is essential for calculating the private key and must also be kept secret. Modern implementations often use a refinement called Carmichael's totient function, $\lambda(n) = lcm(p - 1, q - 1)$, which can sometimes be a smaller value than $\phi(n)$ and works just as well. However, the original paper and the core concept rely on Euler's totient.

Choose the public exponent e. This integer must satisfy two conditions: it must be greater than 1 and less than $\phi(n)$ ($1 < e < \phi(n)$), and it must be coprime to $\phi(n)$ (i.e., $gcd(e, \phi(n)) = 1$). To improve encryption efficiency, *e* is often chosen to be a small number with a low count of '1's in its binary representation. The most common choice in modern applications is the prime number 65537, which is $2^{16} + 1$. The value of *e* is released as the second part of the public key.

Compute the private exponent *d*. This is the crucial final step. The value *d* is the modular multiplicative inverse of *e modulo* $\phi(n)$. It is the unique integer that satisfies the congruence $ed \equiv 1 \mod(\phi(n))$. This value can be calculated efficiently using the Extended Euclidean Algorithm. [12] The resulting number, *d*, is the core component of the private key and must be guarded with absolute secrecy.

At the end of this process, the **public key** is the pair (n, e), which can be shared with anyone. The **private key** is the pair (n, d), which must never be revealed. The original primes *p* and *q*, and the totient $\phi(n)$, can be discarded after *d* is computed, as they are no longer needed for operations but could be used to compromise the key if discovered.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Encryption

To send a secure message to a recipient, the sender performs the following steps:

1. Obtain the recipient's authentic public key *(n, e)*.
2. Convert the plaintext message, *m*, into an integer such that $0 \leq m < n$. If the message is too long, it is broken into smaller blocks, each satisfying this condition. [1]
3. Compute the ciphertext, c, using the recipient's public key with the formula:

4. $c = m^e \pmod n$

5. Send the resulting ciphertext *c* to the recipient.

Decryption

Upon receiving the ciphertext *c*, the recipient uses their own private key *(n, d)* to recover the original message:

1. Compute the original message integer, m, using the formula:

2. $m = c^d \pmod n$

3. Convert the integer *m* back into its original plaintext format.

A Practical Walkthrough with Numerical Example

To make these abstract steps concrete, consider a simplified example using small prime numbers that can be calculated by hand. In a real-world scenario, the primes would have hundreds of digits.

Let's follow the key generation process using the primes p = 61 and q = 53.

Step	Action	Example Calculation (p = 61, q = 53)	Result
1	Choose Primes	p = 61, q = 53	*p* and *q* are secret.
2	Compute Modulus	n = p * q = 61 * 53	*n = 3233* (Public)
3	Compute Totient	$\phi(n) = (p-1)(q-1) = 60 * 52$	$\phi(n) = 3120$ (Secret)
4	Choose Public Exponent	Choose e coprime to 3120. A common prime choice is suitable.	*e = 17* (Public)
5	Compute Private Exponent	Solve for d in 17 * d ≡ 1 mod(3120). Using the Extended Euclidean Algorithm, we find the inverse.	*d = 2753* (Secret)
Result	**Key Pairs**	**Public Key: (3233, 17)** **Private Key: (3233, 2753)**	

Now, suppose a sender wants to encrypt the message "A", which has an ASCII value of 65. So, the plaintext message is *m = 65*.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Encryption:

The sender uses the recipient's public key (3233, 17):

$$c = m^e \pmod{n} = 65^{17} \pmod{3233}$$

This calculation yields:

$$c = 2790$$

The sender transmits the ciphertext 2790.

Decryption:

The recipient uses their private key (3233, 2753) to decrypt the ciphertext:

$$m = c^d \pmod{n} = 2790^{2753} \pmod{3233}$$

This calculation, while enormous, correctly yields:

$$m = 65$$

The recipient converts the number 65 back to its ASCII representation, recovering the original message "A".

The Digital Architect: RSA's Modern Applications (The Decision)

The RSA algorithm is not merely a theoretical construct; it is a foundational technology woven into the very fabric of the internet and modern digital security. Its ability to solve the key exchange problem and provide a basis for digital identity has made it an indispensable tool in a wide array of applications, forcing organisations and individuals to make critical decisions about how they implement trust and privacy online.

Securing the Web: The Role of RSA in SSL/TLS

Perhaps the most ubiquitous application of RSA is in the Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS) protocols—the cryptographic standards that provide the "S" in HTTPS. Every time a user connects to a secure website, RSA often plays a crucial role in the initial "handshake" process that establishes a secure session.

The process works through a hybrid cryptosystem, a pragmatic engineering decision that leverages the strengths of both asymmetric (RSA) and symmetric (e.g., AES) cryptography. Asymmetric algorithms like RSA are computationally intensive and thus too slow for encrypting large volumes of data, such as an entire web browsing session. [10] Symmetric algorithms are incredibly fast but require a shared secret key. The TLS handshake uses RSA to solve this dilemma: [15]

1. When a client (your browser) connects to a server (e.g., yourbank.com), the server presents its digital certificate.
2. This certificate, issued by a trusted Certificate Authority (CA), contains the server's public key *(n, e)* and verifies its identity.
3. The client's browser verifies the certificate's authenticity. It then generates a random symmetric key (called a "pre-master secret") that will be used for the rest of the session.
4. The client encrypts this new symmetric key using the server's RSA public key and sends it to the server.
5. Because only the server possesses the corresponding private key, it is the only entity that can decrypt the message and recover the symmetric key.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

At this point, both the client and server share the same secret symmetric key. They switch from the slow RSA algorithm to a fast symmetric algorithm like AES to encrypt all subsequent data exchanged during the session. [11] This hybrid approach provides a best-of-both-worlds solution: the security of public-key cryptography for key exchange and the speed of symmetric cryptography for bulk data encryption.

The Unforgeable Signature: Authentication and Integrity

Beyond confidentiality, RSA provides a powerful mechanism for ensuring authenticity, integrity, and non-repudiation through **digital signatures**. This application cleverly inverts the normal use of the public and private keys to prove identity rather than to conceal information.

The standard encryption model is for confidentiality: anyone can use a public key to lock a message, but only the private key holder can unlock it. For a digital signature, the logic is reversed: only the private key holder can "lock" (sign) a document, but anyone with the public key can "unlock" (verify) it. This creates mathematical proof of origin. The process is as follows: [15]

1. The sender (signer) takes the message or file to be signed and creates a fixed-size cryptographic hash of it (e.g., using SHA-256). A hash is a unique digital fingerprint of the data.

2. The sender then encrypts this hash value using their own **private key**. The result is the digital signature, which is appended to the original message.

3. The recipient receives the message and the signature. To verify it, they first compute their own hash of the received message.

4. Then, they use the sender's **public key** to decrypt the signature, revealing the original hash computed by the sender.

5. If the two hashes match, the signature is valid. This proves two things:

Authenticity: Since only the sender has the private key, the signature must have originated from them.

Integrity: Even a one-bit change to the message in transit would result in a completely different hash, causing the verification to fail. This proves the message has not been altered.

This mechanism is fundamental to securing software downloads (code signing), legal documents, and ensuring the authenticity of digital certificates themselves.

Beyond the Browser: VPNs, Secure Email, and More

The foundational capabilities of RSA extend far beyond web browsing.

Virtual Private Networks (VPNs): When a user connects to a corporate network via a VPN, RSA is often used during the initial authentication and connection setup phase. Similar to the TLS handshake, it helps authenticate the client and server to each other and securely negotiates the symmetric keys that will be used to encrypt all the data flowing through the VPN tunnel.

Secure Email (PGP and S/MIME): Protocols like Pretty Good Privacy (PGP) and Secure/Multipurpose Internet Mail Extensions (S/MIME) rely heavily on RSA. To send a confidential email, the message is encrypted with the recipient's public key, ensuring that only they can decrypt it with their private key. To prove the sender's identity, emails can be digitally signed using the sender's private key. [13]

The Arms Race: RSA Security and the Quantum Threat (The Dharma)

The invention of RSA initiated an ongoing arms race between cryptographers and cryptanalysts. The security of the algorithm is not absolute; it is a function of contemporary technology, implementation choices, and the

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

discovery of new mathematical attacks. This dynamic forces a continuous re-evaluation of security practices, embodying the principle that digital security requires perpetual vigilance.

The Strength in Numbers: Key Length and Vulnerabilities

The primary defence of the RSA algorithm is the sheer size of its numbers. The security of an RSA key is directly proportional to the difficulty of factoring its modulus, *n*. This difficulty grows exponentially with the number of bits in the key.

In the early days of RSA, key sizes of 512 or 768 bits were considered secure. Today, advances in computing power and factorisation algorithms have rendered such keys obsolete. A 768-bit RSA key was publicly factored in 2009, and a 829-bit key was broken in 2020. Consequently, 1024-bit keys, once a common standard, are now widely deprecated as they are believed to be within reach of well-funded state-level adversaries.

Current industry and government standards, such as those from the National Institute of Standards and Technology (NIST), mandate a minimum key length of **2048 bits** for most applications. For data requiring long-term protection (into the 2030s and beyond), **3072-bit** or **4096-bit** keys are recommended.

It is crucial to understand that the security strength of asymmetric keys does not scale linearly like that of symmetric keys. A 2048-bit RSA key is not "twice as strong" as a 1024-bit key; it is many orders of magnitude stronger. The following table compares the equivalent strengths of different cryptographic systems, providing a more accurate context for making security decisions.

Symmetric Key Strength (bits)	Equivalent RSA Key Length (bits)	Equivalent ECC Key Length (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Beyond brute-force factorisation, RSA implementations can be vulnerable to other, more subtle attacks: [8]

Low Public Exponent Attacks: If a small public exponent (like $e = 3$) is used without proper cryptographic padding, and the same short message is encrypted and sent to multiple recipients, an attacker can use the Chinese Remainder Theorem to recover the original message. This is why modern systems use standardised padding schemes like Optimal Asymmetric Encryption Padding (OAEP).

Timing Attacks: This is a type of side-channel attack where an adversary meticulously measures the time a cryptographic device takes to perform a private key operation. Variations in computation time, which can depend on the bits of the secret key *d*, can leak enough information over many measurements to allow the attacker to reconstruct the key.

Poor Random Number Generation: The security of RSA depends on the unpredictability of its prime factors, *p* and *q*. If the random number generator used to create them is flawed or predictable, an attacker might be able to guess the primes. A more catastrophic failure occurs if two different RSA keys are generated using the same faulty random number generator and happen to share one of their prime factors. An attacker can then find this common factor simply by computing the greatest common divisor (GCD) of the two public moduli, instantly breaking both keys. [6]

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

A final important nuance lies in computational complexity theory. While the integer factorisation problem is known to be in the class **NP** (its solutions can be verified quickly), it is *not* known to be **NP-complete** (the hardest problems in NP). It is widely suspected to belong to a rare intermediate class known as **NP-intermediate**. [7] This means that even a definitive proof that P≠NP would not guarantee that factorisation remains a fundamentally hard problem for classical computers.

The Quantum Menace: Shor's Algorithm

The most profound threat to RSA does not come from classical computers but from an entirely new paradigm of computation: quantum computing. In 1994, mathematician Peter Shor developed a quantum algorithm that fundamentally changed the nature of the integer factorisation problem. [16]

For a classical computer, the difficulty of factoring grows exponentially with the size of the number. For a quantum computer running Shor's algorithm, the problem can be solved in **polynomial time**. [14] This represents a true paradigm shift. While a classical supercomputer might take thousands of years to factor a 2048-bit RSA key, a sufficiently large and stable quantum computer could theoretically do so in a matter of hours or days. [3]

Shor's algorithm does not simply speed up classical factorisation methods. It attacks the problem in a completely different way by reframing it. It transforms the task of factoring *n* into the task of finding the *period* of a specific modular function. While period-finding is also difficult for classical computers, it is a problem that is perfectly suited to the unique capabilities of quantum machines, which can use principles like superposition and the Quantum Fourier Transform to find the period efficiently. [14]

This algorithm is not just an incremental improvement; it is a paradigm-killing discovery. It invalidates the core assumption of computational difficulty upon which RSA and other public-key systems like Elliptic Curve Cryptography (ECC) are built. The existence of Shor's algorithm has forced the entire cryptographic community to confront the mortality of its most trusted tools and to begin the urgent search for a new generation of cryptography.

The Next Generation: An Introduction to Post-Quantum Cryptography

In response to the quantum threat, the field of **Post-Quantum Cryptography (PQC)** has emerged. Its goal is to develop new public-key algorithms that are secure against attacks from both classical and quantum computers. [2] These algorithms are not based on integer factorisation or the discrete logarithm problem (which also falls to Shor's algorithm), but on different mathematical problems believed to be hard even for quantum computers. [9]

NIST has been leading a multi-year process to standardise PQC algorithms. The main categories of candidates include [3]:

- **Lattice-based Cryptography:** Based on the difficulty of finding the shortest vector in a high-dimensional geometric lattice.
- **Code-based Cryptography:** Based on the difficulty of decoding a random linear error-correcting code.
- **Hash-based Cryptography:** Based on the security of cryptographic hash functions.
- **Multivariate Cryptography:** Based on the difficulty of solving systems of multivariate polynomial equations.

The following table provides a high-level comparison of RSA with its main classical peer, ECC, and the symmetric workhorse, AES, to contextualise their roles and vulnerabilities.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Feature	AES (Advanced Encryption Standard)	RSA (Rivest-Shamir-Adleman)	ECC (Elliptic Curve Cryptography)
Type	Symmetric (Single Shared Key)	Asymmetric (Public/Private Key Pair)	Asymmetric (Public/Private Key Pair)
Primary Use	Bulk data encryption (files, connections)	Key exchange, Digital signatures	Key exchange, Digital signatures
Key Size	128, 192, 256 bits	2048, 3072, 4096 bits	256, 384, 521 bits
Performance	Very Fast	Slow (computationally intensive)	Faster than RSA for equivalent security
Quantum Threat	Grover's Algorithm (mitigated by doubling key size)	Shor's Algorithm (completely broken)	Shor's Algorithm (completely broken)

The transition to PQC is already underway. A common strategy being deployed is the use of **hybrid schemes**, where a classical algorithm like RSA or ECC is used alongside a new PQC algorithm (like the NIST-selected CRYSTALS-Kyber). This ensures that the connection remains secure even if one of the algorithms is broken, providing backward compatibility and a bridge to a fully post-quantum future. [5]

CONCLUSION:

The Enduring Legacy of an Idea

The RSA algorithm stands as one of the most significant intellectual achievements of the 20th century. It was far more than a clever application of number theory; it was the first practical embodiment of public-key cryptography, a concept that fundamentally altered our ability to forge trust and privacy in a world built on public networks. By transforming the mathematical asymmetry of integer factorisation into a workable cryptographic primitive, RSA provided the essential tools for secure e-commerce, private digital communications, and verifiable online identity, thereby enabling the very architecture of the modern internet.

Its story is a perfect microcosm of the perpetual cycle that defines the field of cybersecurity. A deep mathematical principle—the Dharma of factoring's computational difficulty—gave rise to a revolutionary technology. This technology, in turn, forced society to make new and critical Decisions about how to build and manage digital trust. Now, a new technological horizon—quantum computing—threatens to shatter that original mathematical principle, compelling the global community to seek new foundational truths and make the next generation of security decisions.

While the RSA algorithm itself may eventually be relegated to the annals of cryptographic history, succeeded by quantum-resistant successors, the paradigm it unleashed is permanent. The revolutionary idea that secrets can be established in the open, that identity can be proven with mathematical certainty, and that trust can be engineered without a pre-existing relationship is an indispensable and enduring part of our connected civilisation. RSA's greatest legacy is not the code or the mathematics, but the demonstration that such a world is possible.

REFERENCES

1. Durai Raj Vincent, P. M. (2016, June 1). *RSA encryption algorithm - A survey on its various forms and its security level*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/306215546_RSA_encryption_algorithm_A_survey_on_its_various_forms_and_its_security_level
2. Galbraith, S. (2025, December 1). *Post-Quantum Cryptography*. ResearchGate. Retrieved December 21,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

2025, from https://www.researchgate.net/publication/398244591_Post-Quantum_Cryptography
3. Grover, L. K. (1996, June 1). *Fast quantum mechanical algorithm for database search*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/2201607_Fast_quantum_mechanical_algorithm_for_database_search
4. Ireland, K., & Rosen, M. (1990). *A Classical Introduction to Modern Number Theory* (2nd ed.). Springer-Verlag New York Inc.
5. Jain, B., & Mittal, H. K. (2025, May 1). *Post-Quantum Cryptography: A Comprehensive Review of Past Technologies and Current Advances*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/392329453_Post-Quantum_Cryptography_A_Comprehensive_Review_of_Past_Technologies_and_Current_Advances
6. Kulandei, B., & Dhenakaran, S. S. (2017, October 1). *An Overview of Cryptanalysis of RSA Public key System*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/321031748_An_Overview_of_Cryptanalysis_of_RSA_Public_key_System
7. Lenstra, A. K., & Lenstra, H. W. (1993). *The Development of the Number Field Sieve*. Springer. Retrieved December 21, 2025, from https://link.springer.com/book/10.1007/BFb0091534
8. Luo, Z. J., Liu, R., Mehta, A., & Ali, M. L. (n.d.). *Demystifying the RSA Algorithm: An Intuitive Introduction for Novices in Cybersecurity*. Arxiv. Retrieved December 21, 2025, from https://arxiv.org/html/2308.02785v2
9. Mahto, D., & Yadav, D. K. (2017, January 1). *RSA and ECC: A comparative analysis*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/322558426_RSA_and_ECC_A_comparative_analysis
10. Olutola, A., & Matthew, O. (2023, January 1). *Comparative Analysis of Encryption Algorithms*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/366889851_Comparative_Analysis_of_Encryption_Algorithms
11. Paul, J. (2025, July 1). *USE OF DES, AES, AND RSA IN SECURE COMMUNICATION PROTOCOLS (SSL/TLS, VPN, ETC.)*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/393976056_USE_OF_DES_AES_AND_RSA_IN_SECURE_COMMUNICATION_PROTOCOLS_SSLTLS_VPN_ETC
12. PITE, J., ZHONG, Y., & ZHU, H. (1977). *THE RSA CRYPTOSYSTEM*. Massachusetts Institute of Technology. Retrieved December 21, 2025, from https://math.mit.edu/research/highschool/primes/circle/documents/2024/Honglin.pdf
13. Rivest, R., Shamir, A., & Adelman, L. (2021, February 1). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (1978)*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/349073609_A_Method_for_Obtaining_Digital_Signatures_and_Public-Key_Cryptosystems_1978
14. Shor, P. W. (2006, June 1). *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/280821144_Polynomial-Time_Algorithms_for_Prime_Factorization_and_Discrete_Logarithms_on_a_Quantum_Computer
15. Trappe, W., & Washington, L. C. (2006). *Introduction to Cryptography with Coding Theory* (2nd ed.). Prentice Hall.
16. Van Meter, R., & Itoh, K. M. (2004, September 1). *Fast Quantum Modular Exponentiation*. ResearchGate. Retrieved December 21, 2025, from https://www.researchgate.net/publication/2193429_Fast_Quantum_Modular_Exponentiation