Implementation and Analysis of RSA Digital Signatures and ISO/IEC 9796 Standard

Author: **Vikash Kumar Ojha (CS22B013)**

Department of Computer Science and Engineering

Indian Institute of Technology Madras, Chennai, Tamil Nadu, India

cs22b013@smail.iitm.ac.in

September 25, 2025

Abstract

Research Overview

- Digital signatures are fundamental components of modern information security
- Ensure authenticity, integrity, and non-repudiation of electronic communications
- RSA scheme remains a cornerstone for secure digital transactions
- ISO/IEC 9796 standard specifies digital signature schemes with message recovery
- Study evaluates functional aspects, security properties, and computational performance

Index Terms

Key Concepts Covered

- RSA Algorithm
- Digital Signatures
- ISO/IEC 9796 Standard
- Cryptography
- Information Security
- Message Recovery
- Public-Key Infrastructure
- Authentication
- Non-Repudiation
- Data Integrity

These fundamental concepts form the foundation of this research on secure digital signature implementations and their standardized protocols.

Introduction

Digital Signatures in Modern Communication

- Essential for ensuring authenticity, integrity, and non-repudiation
- RSA algorithm: widely used public-key scheme for digital signatures
- ISO/IEC 9796 standard defines signature schemes with message recovery
- Improves efficiency by embedding parts of the message into the signature
- Reduces transmission overhead but introduces performance vs. security trade-offs

RSA History: Origins

The Birth of Asymmetric Cryptography

- **1976** Whitfield Diffie and Martin Hellman introduced asymmetric public-private key concept
- Diffie-Hellman key exchange used exponentiation modulo prime numbers
- Also introduced the concept of digital signatures
- Need for computationally hard-to-invert functions identified
- Revolutionized cryptography by moving beyond symmetric key systems

RSA History: Development Journey

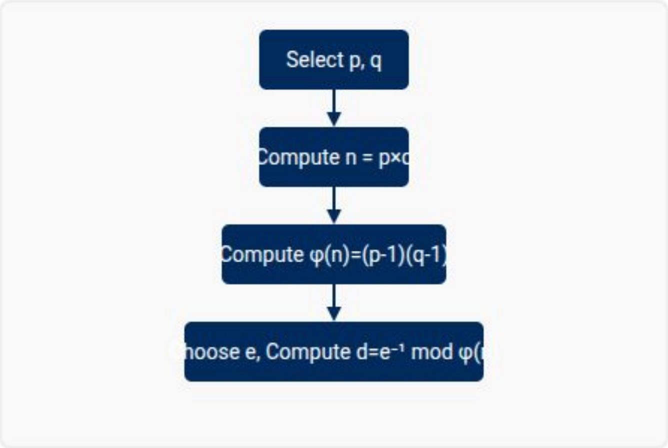
Path to RSA Algorithm

- **The MIT Team (1976-1977):**
  - **Ron Rivest** (Computer Scientist)
  - **Adi Shamir** (Computer Scientist)
  - **Leonard Adleman** (Mathematician)
- **Failed Approaches:**
  - Knapsack-based approach (NP-hardness of knapsack problem) - broken
  - Permutation polynomials (special polynomials over finite fields) - weakness found
- **Success in 1977:**
  - RSA algorithm based on mathematical difficulty of factoring large composite integers
  - Named after initials of discoverers' surnames

RSA Algorithm: Key Generation

Step-by-Step Key Generation Process

1. **Select** two large prime numbers p and q
2. **Compute** $n = p \times q$ (modulus for both keys)
3. **Compute** Euler's totient function $\varphi(n) = (p - 1)(q - 1)$
4. **Choose** integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$
5. **Compute** private exponent $d \equiv e^{-1} \pmod{\varphi(n)}$
6. **Publish** public key (e, n) and keep private key (d, n) secret



RSA key generation flowchart illustrating the mathematical process

Security Parameter Bounds: Key Length

Minimum Security Standards

Key Size (n)	Minimum (bits)	Recommended (bits)	Future Proof (bits)
RSA Modulus	2048	3072	4096

Prime Selection Criteria:

- $|p|, |q| \geq 1024$ bits (for 2048-bit modulus)
- $|p - q| > 2^{|n|/2-100}$ (sufficient prime gap)
- $\gcd(p - 1, q - 1) \leq 2^{64}$ (small common factors)

Note: These parameters represent minimum requirements for cryptographically secure RSA implementations.

Security Parameter Bounds: Exponent Constraints

Public and Private Exponent Security

Public Exponent Constraints:

- $e \geq 65537 = 2^{16} + 1$ (minimum recommended)
- $e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$
- e should be odd and have small Hamming weight

Private Exponent Security:

- $d > 2^{|n|/4}$ (Wiener's attack protection)
- $d < \lambda(n)$ where $\lambda(n) = \mathrm{lcm}(p - 1, q - 1)$
- $|d| \approx |n|$ (full-length private exponent preferred)

Security Implications:

Weak exponent selection can lead to complete system compromise through mathematical attacks without needing to factor the RSA modulus

Security Level Equivalencies

RSA vs. Symmetric Encryption Comparison

RSA Key Size	Security Level	AES Equivalent	Hash Function
3072 bits	128 bits	AES-128	SHA-256
7680 bits	192 bits	AES-192	SHA-384
15360 bits	256 bits	AES-256	SHA-512

Note: These equivalencies represent comparable security levels across different cryptographic primitives based on NIST recommendations and current computational capabilities.

Padding Scheme Requirements

Secure RSA Implementation

- For OAEP Padding (Optimal Asymmetric Encryption Padding):
  - Message length constraint: `|M| ≤ |n| - 2k - 2`
  - Where k = hash output length in bytes
  - Random padding length: ≥ 8 bytes minimum
- Security Considerations:
  - Prevents chosen-ciphertext attacks
  - Achieves semantic security under random oracle model
  - Adds randomization to ensure identical messages encrypt differently

RSA Encryption and Decryption

Mathematical Operations

Encryption:

For plaintext message M ($0 \le M < n$)

$$C \equiv M^e \pmod{n}$$

Decryption:

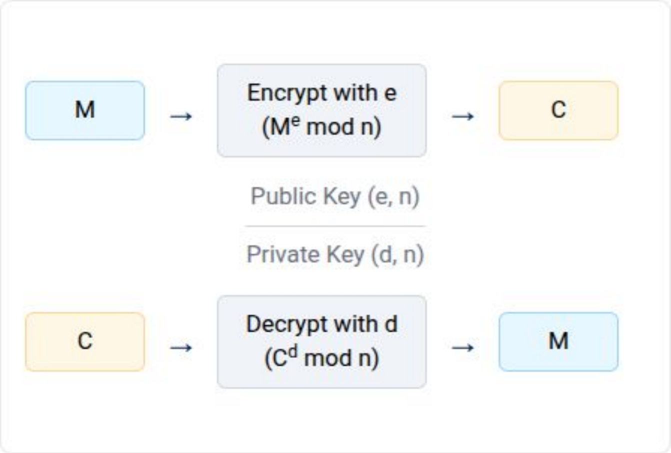
Original message recovery

$$M \equiv C^d \pmod{n}$$

Correctness Property:

$$M^{ed} \equiv M \pmod{n}$$

Holds due to Euler's theorem when $ed \equiv 1 \pmod{\varphi(n)}$



RSA encryption and decryption workflow using modular exponentiation

RSA Digital Signature: Generation

Creating Digital Signatures

- **Step 1:** Compute message digest H(M) using secure hash function
- **Step 2:** Using sender's private key (d, n), compute signature: `S ≡ (H(M))^d (mod n)`
- **Step 3:** Transmit pair (M, S) to receiver

Security Properties:

- Ensures authenticity, integrity, and non-repudiation
- Only private key holder can generate valid signature

RSA Digital Signature: Verification

Verifying Digital Signatures

Signature Verification Process:

- **1.** Receiver computes message digest `H(M)`
- **2.** Using sender's public key `(e, n)`, recover hash:

  `H'(M) ≡ S^e (mod n)`
- **3.** Compare `H(M)` and `H'(M)`:
  - Equal: signature valid
  - Not equal: message altered or signature invalid

Detection Capability:

- Any modification to message or signature will be detected

ISO/IEC 9796 Standard

Introduction

Standardized Digital Signature Protocol

ISO/IEC 9796 History: Evolution

Standard Development Timeline

- **Earlier Versions (Withdrawn):** ISO/IEC 9796:1991 (ISO/IEC 9796-1), ISO/IEC 9796-2:1997
- **Current Secure Versions:** ISO/IEC 9796-2:2002, ISO/IEC 9796-3:2006
- **Reason for Updates:** Earlier versions were cryptographically attacked and broken
- Replaced with more secure implementations addressing vulnerabilities
- Standards evolution represents ongoing improvements in cryptographic security

Vulnerabilities in Earlier Versions

Why Earlier Standards Failed

ISO/IEC 9796:1991 (9796-1) Vulnerabilities:

- No cryptographic hash function used
- Relied on error-correcting codes for redundancy
- Signature not uniquely bound to message content
- Predictable redundancy structure enabled chosen-message attacks
- Withdrawn in 1999

ISO/IEC 9796-2:1997 Vulnerabilities:

- Introduced hash functions but still vulnerable
- Encoding with deterministic redundancy created algebraic patterns
- Exploitable through adaptive chosen-message attacks
- Insecure under certain attack scenarios

ISO/IEC 9796 Features

Key Capabilities

Message Recovery:

Unlike typical signatures requiring both message and signature
Allows verifier to extract original message from signature during verification
More efficient than traditional schemes

Redundancy:

Includes specific redundancy patterns
Prevents existential forgery attacks
Ensures recovered message authenticity

Padding:

Defines specific padding schemes
Ensures security against various cryptographic attacks
Critical for implementation security

ISO/IEC 9796-2 Protocol: Signature Generation

Algorithm Implementation

Input Requirements:

- Message M
- Private key (d, n)
- Hash function H

Generation Steps:

1. salt = RandomBytes(16)
2. trailer = ConstructTrailer(|M|, H_id, salt)
3. M' = M ‖ trailer
4. digest = H(M')
5. μ = FormatRepresentative(digest, M')
6. S = RSA_sign(μ, d, n)

Representative Format:

```
μ = 0x6A ‖ digest ‖ padding ‖ M' ‖ 0xBC
```

Input Message M

↓

Generate Random Salt
16 bytes of entropy

↓

Construct Trailer
Format metadata

↓

Append Trailer to Message
M' = M ‖ trailer

↓

Compute Hash
digest = H(M')

↓

Format Representative
Add headers, padding

↓

RSA Sign
$S = \mu^d \bmod n$

ISO/IEC 9796-2 Protocol: Message Recovery

Verification and Recovery Process

- **Input Requirements:** Signature S, Public key (e, n), Hash function H
- **Recovery Steps:**
  1. μ = RSA_verify(S, e, n)
  2. (digest_rec, M'_rec) = ParseRepresentative(μ)
  3. Validate format or return INVALID
  4. digest_comp = H(M'_rec)
  5. If digest_rec ≠ digest_comp return INVALID
  6. Extract and return message M
- **Dual Functionality:** Simultaneously extracts original message and verifies signature authenticity

Message Recovery Process Flow

Signature → RSA Decrypt → Parse Format → Validate Structure → Extract Message + Hash → Compare Hash → Verify + Recover

ISO/IEC 9796-2 combines verification with message recovery in a single operation

Format Validation Framework

Security Validation Criteria

Check	Requirement	Security Purpose
Header	0x6A at position 0	Format identification
Footer	0xBC at final position	Structural integrity
Padding	Alternating 0xBB/0xAA	Forgery prevention
Trailer	Valid length/hash ID	Metadata verification
Hash	Digest consistency	Message authenticity

Security Implementation:

- Multiple security checks ensure compliance with ISO/IEC 9796-2
- Invalid formats immediately terminate verification
- Prevents potential security vulnerabilities

Key Advantages and Trade-offs

Performance vs. Security Analysis

Advantages:

- Reduced transmission overhead through message recovery
- Combines signature verification with message extraction
- Enhanced efficiency compared to traditional signature schemes
- Standardized implementation ensures interoperability

Trade-offs:

- Increased computational complexity during verification
- More complex implementation compared to basic RSA signatures
- Careful balance required between efficiency and security
- Historical vulnerabilities required multiple standard revisions

Real-world Applications

Practical Implementation Scenarios

- **Bandwidth-constrained environments** - Optimizes transmission in limited network capacity scenarios
- **Systems requiring message authentication with recovery** - Critical for scenarios where original message must be extractable
- **Digital document signing** - With space optimization for efficient storage and transmission
- **Secure communication systems** - With stringent efficiency requirements

Implementation Considerations:

- Must follow current secure standards (2002/2006 versions)
- Proper validation framework implementation critical
- Regular security assessment recommended

Compliance Requirement

Must adhere to latest cryptographic best practices

References

Key Sources

1. **NIST Digital Signature Standard (DSS)** - Federal Information Processing Standards Publication 186-5, Feb. 2023
2. **NIST Key Management Recommendation** - Special Publication 800-57 Part 1 Revision 5, May 2020
3. **ISO/IEC 9796-2:2002** - Digital Signature Schemes Giving Message Recovery, Part 2: Integer Factorization-based Mechanisms

Thank you for your attention!