Int. j. adv. multidisc. res. stud. 2026; 6(4):842-853

Received: 03-06-2026
Accepted: 13-07-2026

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

Review of Encryption Techniques: A Comprehensive Survey of Classical, Symmetric, Asymmetric, Homomorphic, Blockchain-Integrated, and Post-Quantum Cryptographic Methods

[1] Anah Hassan Bijik, [2] Sojah Patrick Yakubu, [3] Ibrahim Lawal, [4] Emmanuel Jonathan
[1, 3] Department of Computer Science, Bingham University, Karu, Nasarawa State, Nigeria
[2, 4] Department of Information Technology, Bingham University, Karu, Nasarawa State, Nigeria

Corresponding Author: **Anah Hassan Bijik**

Abstract
Encryption is the cornerstone of modern information security, enabling confidential communication across digital infrastructures that span personal devices, enterprise networks, cloud platforms, distributed ledgers, and the Internet of Things. This article provides a structured, in-depth review of encryption techniques documented in twenty-eight authoritative sources. Coverage spans symmetric algorithms (AES, DES, 3DES, Blowfish, RC4), asymmetric methods (RSA, ECC, ElGamal), classical information-theoretically secure ciphers (One-Time Pad), hybrid encryption systems, homomorphic encryption, blockchain, searchable encryption, and post-quantum cryptographic approaches based on lattice problems, code-based assumptions, and hash functions. For each technique, this review examines theoretical foundations, algorithmic structure, operational workflow, practical implementations, computational trade-offs, and security properties under both classical and quantum threat models. Structural process diagrams are provided to illustrate how plaintext is transformed to ciphertext and back under each paradigm. The following metrics were covered during the analysis Recommendations were made on what the best applicable solution across security level, key size, throughput, and quantum resistance demonstrates that no single algorithm universally satisfies all competing demands; hybrid and lattice-based designs represent the most promising directions for future-proof encryption. Open challenges including Fully Homomorphic Encryption (FHE) performance, post-quantum migration, IoT key management, and regulatory alignment are discussed, followed by a research agenda for the post-quantum era.

Keywords: Encryption, AES, RSA, ECC, Homomorphic Encryption, Post-Quantum Cryptography, Blockchain

1. Introduction

The protection of information has been a fundamental human concern for millennia. Long before the digital age, commanders encrypted battlefield orders with transposition and substitution ciphers; diplomats communicated through coded correspondence; and spies relied on one-time pads to prevent their messages from being read by adversaries. The history of cryptography is, in many respects, a history of the perpetual contest between those who construct codes and those who seek to break them. Each advance in cryptanalytic capability has prompted a corresponding advance in cipher design, a dynamic that shows no sign of ending as the computing landscape transforms under the influence of quantum mechanics, machine learning, and ubiquitous connectivity.

In the contemporary digital landscape, encryption underpins virtually every form of secure communication and data storage: online banking and payment systems, electronic health records, cloud-hosted enterprise applications, governmental data exchanges, end-to-end encrypted messaging platforms, and the rapidly expanding ecosystem of Internet of Things (IoT) devices. The global market for encryption software was valued at approximately USD 12.2 billion in 2023 and is projected to exceed USD 30 billion by 2030, reflecting the growing recognition that data confidentiality is a business-critical property, not merely a regulatory compliance checkbox (Fortune Business Insights, 2024) [7].

The threat landscape that encryption must contend with has grown correspondingly complex. Classical attackers' adversaries equipped with conventional computing hardware threaten systems through brute-force key search, differential and linear

cryptanalysis, side-channel attacks, and protocol-level weaknesses. Nation-state actors combine these classical techniques with vast computational resources and zero-day exploit capabilities. Looking further ahead, the prospect of fault tolerant quantum computers capable of running Shor's algorithm at scale threatens to invalidate virtually all currently deployed public-key cryptography, including RSA, elliptic-curve cryptography (ECC), and Diffie-Hellman key exchange the mechanisms that secure the vast majority of today's encrypted communications. The U.S. National Institute of Standards and Technology (NIST) formally initiated its post-quantum cryptography standardization process in 2016 and published its first three post-quantum standards in 2024 (NIST, 2024) [25], signaling that the transition from classical to quantum resistant cryptography is no longer a theoretical future concern but a practical engineering mandate.

Against this backdrop, the body of academic and technical literature on encryption has grown enormously. Peer-reviewed journals, pre-print archives, and textbooks collectively offer thousands of articles covering algorithm design, security proofs, implementation optimization, protocol engineering, and application-domain case studies. Practitioners and researchers alike require synthesis: comprehensive reviews that map the landscape, compare alternatives, and identify where the frontiers of the field currently lie. This article provides that synthesis.

The review draws on twenty-eight sources published between 2019 and 2026, spanning the full spectrum from foundational textbook treatments (Boneh & Shoup, 2020; Tenzer, 2022; Stinson & Paterson, 2019) [2, 37, 36] to cutting edge pre-print research on post quantum homomorphic encryption (Chen, 2024) [3]. Sources were selected to ensure coverage of each major encryption paradigm, representation of both theoretical and applied perspectives, and inclusion of application-domain case studies in IoT, cloud computing, distributed systems, electronic voting, and privacy-preserving machine learning.

A distinguishing feature of this review relative to existing surveys is the inclusion of structured process diagrams step-by-step visual representations of how encryption and decryption operate under each paradigm. Many practitioners understand encryption as a black box: data goes in, ciphertext comes out. Understanding the internal mechanics key scheduling, round functions, mode of operation, padding, ciphertext format is essential for correct implementation, appropriate cipher selection, and informed security evaluation. The pseudo code & Algorithm in Sections 3 through 8 are intended to bridge the gap between conceptual understanding and implementation awareness.

1.1 Aim and Objectives

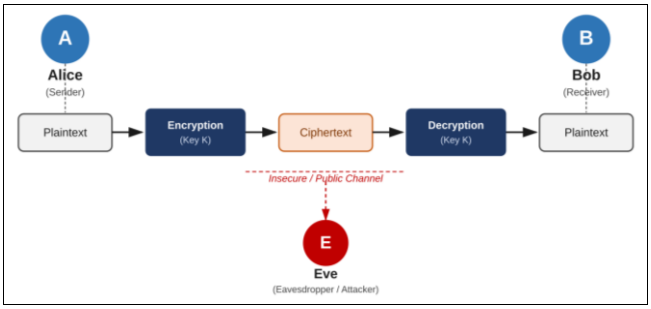
The aim of this study is to conduct a comprehensive review of existing encryption algorithms by examining their underlying principles, evaluating their computational cost, and comparing their performance using relevant evaluation metrics. The study also seeks to assess the applicability of these algorithms in real-world scenarios, identify their strengths and limitations, and determine their suitability for different security requirements and computing environments. Specifically, the study aims to review and compare existing encryption algorithms based on computational cost and selected performance metrics, evaluate their efficiency and effectiveness, and analyze their practical applications across various domains to provide insights that will guide the selection of appropriate encryption techniques for specific use cases.

2. Core Concepts

2.1 What is Cryptography?

Cryptography is the science and practice of securing information and communication by transforming it into forms that are unintelligible to unauthorized parties, while remaining recoverable by those who possess the correct key (Stallings, 2020; Menezes *et al*., 1996) [35, 19]. The word originates from the Greek terms kryptos, meaning "hidden," and graphein, meaning "to write," and its central concern is enabling two parties to exchange information over a channel that may be observed or interfered with by a third party, without that third party being able to understand or tamper with the content.

The classical way of introducing this idea is through a communication model involving two participants, conventionally named Alice and Bob, who wish to exchange a message securely, and a third party, Eve, who represents an eavesdropper attempting to intercept or interpret that communication (Stallings, 2020) [35]. Fig 1 illustrates this foundational concept: Alice converts her plaintext message into ciphertext using an encryption process and a secret key, transmits the ciphertext across an insecure channel, and Bob reverses the process using a decryption operation to recover the original plaintext. Eve may observe the ciphertext in transit, but without knowledge of the key she cannot feasibly reconstruct the plaintext.



Source: Stallings (2020) [35].

Fig 1: The Alice-and-Bob model illustrating the basic concept of cryptographic communication

This simple model captures the essence of cryptography: it is fundamentally about controlling who can access, modify, or verify information, using mathematical transformations rather than physical secrecy alone.

2.2 Core Concepts of Cryptography

Modern cryptography is built around a small set of security objectives, often referred to as its core concepts or pillars (Stallings, 2020; Menezes *et al*., 1996) [35, 19]. These concepts define what a cryptographic system is designed to guarantee, independent of the specific algorithm used to achieve it. Fig 2 depicts these four pillars as the structural supports of cryptography as a discipline.

1. Confidentiality: ensures that information is disclosed only to those authorized to access it, typically achieved through encryption.
2. Integrity: ensures that data has not been altered, accidentally or maliciously, between the time it was created and the time it is accessed, typically verified

through hash functions and message authentication codes.
3. Authentication: verifies the identity of a communicating party or the origin of a piece of data, so that a recipient can be confident about who they are really communicating with.
4. Non-repudiation: prevents a party from later denying that they performed an action, such as sending a message or signing a document, typically enforced through digital signatures.

Together, these four properties form the basis against which any cryptographic scheme, protocol, or system is evaluated, and each of the categories discussed in Section 4 contributes differently to achieving them.

2.3 Evolution of Cryptography

Cryptography has evolved considerably from simple manual techniques to the mathematically rigorous, computationally intensive systems used today (Kahn, 1996; Singh, 1999) [12, 34]. Fig 3 traces this evolution across five broad eras.

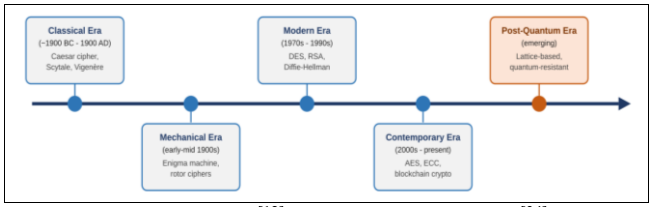
Classical Era (antiquity to c. 1900): reliance on manual substitution and transposition techniques such as the Caesar cipher and the Spartan scytale, where security depended largely on the secrecy of the method itself (Kahn, 1996; Singh, 1999) [12, 34].

Mechanical Era (early-to-mid 1900s): the introduction of electromechanical devices such as the Enigma machine, which used rotating rotors to produce far more complex substitution patterns than manual methods could achieve (Kahn, 1996) [12].

Modern Era (1970s-1990s): the advent of digital computing enabled formal, mathematically defined algorithms, including the Data Encryption Standard (DES) and the RSA public-key algorithm, alongside the Diffie-Hellman key exchange, which for the first time allowed two parties to establish a shared secret without meeting in advance (Diffie & Hellman, 1976; Rivest *et al*., 1978; National Institute of Standards and Technology [NIST], 1999) [5, 30, 20].

Contemporary Era (2000s-present): the adoption of the Advanced Encryption Standard (AES) as the dominant symmetric algorithm, growing use of Elliptic Curve Cryptography (ECC) for efficient public-key operations, and cryptographic techniques underpinning blockchain and distributed-ledger systems (NIST, 2001) [21].

Post-Quantum Era (emerging): active research and standardization of lattice-based and other quantum-resistant algorithms, motivated by the threat that sufficiently powerful quantum computers would pose to widely deployed public-key schemes such as RSA and ECC (NIST, 2016) [23].



Source: Kahn (1996) [12] and Singh (1999) [34] and the standardization timelines published by NIST (1999, 2001, 2016) [20, 21, 23].

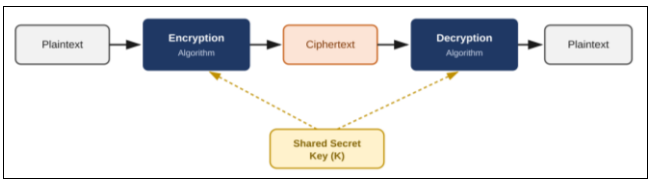
Fig 2: Timeline showing the evolution of cryptography from the classical era to the emerging post-quantum era

2.4 Categories of Cryptography

Contemporary cryptographic techniques are generally grouped into three broad categories based on how keys are used and what function the technique performs: symmetric-key cryptography, asymmetric-key cryptography, and cryptographic hash functions.

2.4.1 Symmetric-Key Cryptography

In symmetric-key cryptography, the same secret key is used for both encryption and decryption (Stallings, 2020; Menezes *et al*., 1996) [35, 19]. Fig 4 shows this process: the sender encrypts the plaintext with a shared key K to produce ciphertext, and the receiver applies the same key K to reverse the transformation and recover the plaintext. Because both parties must possess the identical key, the key itself must be exchanged through a secure channel before communication begins, which is often the main practical challenge of this category. Symmetric algorithms, such as AES and the earlier DES, are generally fast and well suited to encrypting large volumes of data (NIST, 1999, 2001) [20, 21].

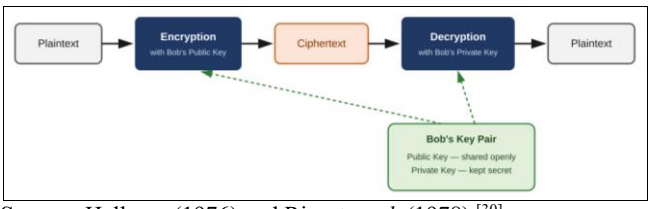


Source: Stallings (2020) [35].

Fig 3: Symmetric-key encryption and decryption process using a single shared secret key

2.4.2 Asymmetric-Key Cryptography

Asymmetric-key cryptography, also called public-key cryptography, uses a mathematically linked pair of keys: a public key, which may be shared openly, and a private key, which is kept secret by its owner (Diffie & Hellman, 1976; Rivest *et al*., 1978) [5, 30]. Fig 5 illustrates the process using Bob as the receiver: Alice encrypts her message using Bob's public key, and only Bob's corresponding private key can decrypt the resulting ciphertext. This eliminates the need to exchange a shared secret in advance and enables additional capabilities, such as digital signatures, but at the cost of significantly higher computational overhead than symmetric algorithms. RSA and Elliptic Curve Cryptography are widely used examples of this category (Rivest *et al*., 1978) [30].



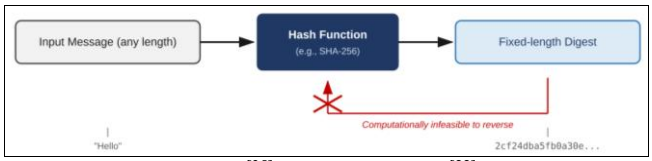
Source: Hellman (1976) and Rivest *et al*. (1978) [30].

Fig 4: Asymmetric-key encryption and decryption process using a public and private key pair

2.4.3 Hash Functions

Cryptographic hash functions differ from the two previous categories in that they are not designed for encryption or decryption at all, but for producing a fixed-length "fingerprint," or digest, of an input of arbitrary length

(Stallings, 2020; NIST, 2015) [35, 22]. Fig 6 shows an input message being processed by a hash function, such as SHA-256, to produce a fixed-length digest. This transformation is deliberately one-way: it is computationally infeasible to reconstruct the original input from its digest, and even a single-character change to the input produces a completely different digest, a property known as the avalanche effect. Hash functions underpin integrity checking, password storage, and digital signature schemes, and are typically used in combination with symmetric or asymmetric techniques rather than as a replacement for them (NIST, 2015) [22].



Source: Stallings (2020) [35] and NIST (2015) [22].

Fig 5: The hash function process, converting a variable-length input into a fixed-length digest through a one-way transformation

Taken together, symmetric-key cryptography, asymmetric-key cryptography, and hash functions are rarely used in isolation. Practical systems, such as the TLS protocol that secures web traffic, typically combine all three: asymmetric cryptography to establish a shared secret, symmetric cryptography to encrypt the bulk of the data efficiently, and hash functions to verify integrity and support digital signatures.

2.5 The General Encryption

Plaintext (M): The original readable message or data before encryption. Plaintext may be text, binary data, images, or any structured information.

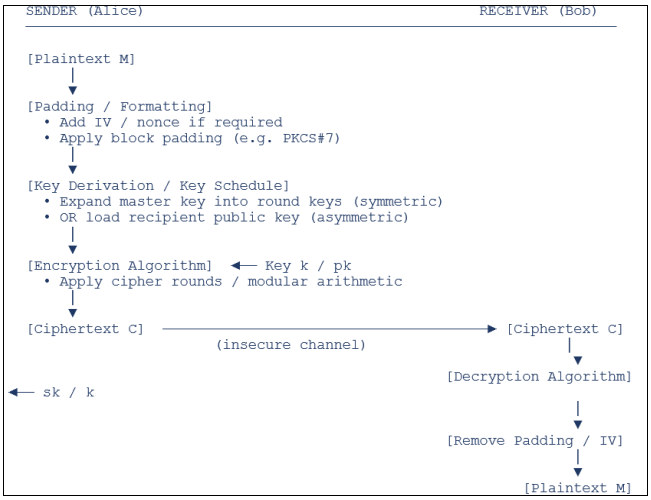
Ciphertext (C): The scrambled output of the encryption algorithm, unintelligible to any party that does not possess the correct decryption key. The relationship between plaintext and ciphertext must be computationally indistinguishable from random to an attacker without the key.

Key (k / pk / sk): A secret parameter that controls the encryption and decryption transformations. Key length (in bits) is one of the primary determinants of security level. A symmetric scheme uses one shared key; an asymmetric scheme uses a mathematically related key pair a public key for encryption and a private key for decryption.

Security Parameter (λ): An integer (e.g., 128, 256) that governs the size of keys and other parameters. Security is expressed as a function of λ: an attacker should require 2^λ operations to break the scheme with non-negligible probability.

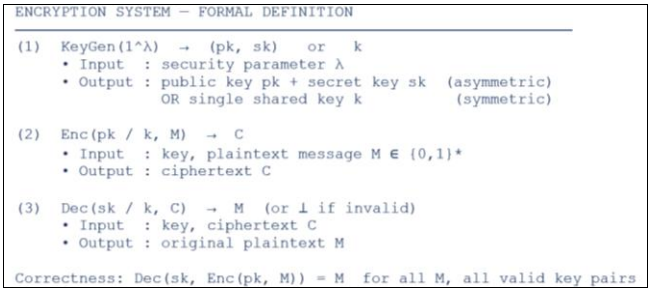
Cipher Modes of Operation: For symmetric block ciphers, a mode of operation specifies how the cipher is applied to data longer than a single block. Common modes include ECB (not recommended), CBC, CTR, GCM, and CFB (Boneh & Shoup, 2020; Dworkin, 2001) [2, 6].

2.5.1 The General Encryption–Decryption Pipeline



3. Fundamental Structure of an Encryption System

Before examining individual techniques, it is essential to establish the conceptual vocabulary common to all encryption systems. An encryption scheme is defined by three algorithms: a key generation algorithm KeyGen, an encryption algorithm Enc, and a decryption algorithm Dec. Formally:

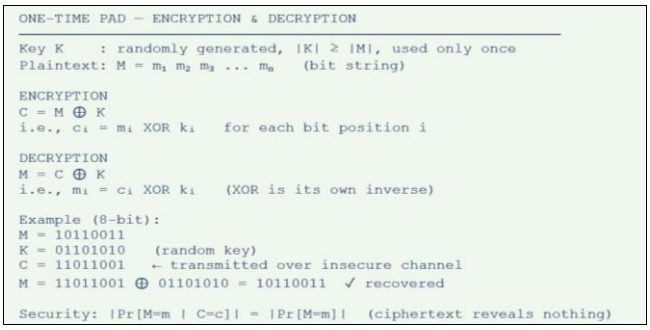


3.1 Substitution and Transposition Ciphers

The earliest encryption systems relied on two primitive operations: substitution (replacing each symbol with another according to a fixed table) and transposition (rearranging the order of symbols). The Caesar cipher shifts each letter by a fixed amount; the Vigenère cipher uses a repeated keyword to vary the shift. Both are broken trivially today through frequency analysis, but they established the conceptual vocabulary key, plaintext, ciphertext, key space that underpins all modern cryptography (Stinson & Paterson, 2019) [36].

3.2 The One-Time Pad (OTP)

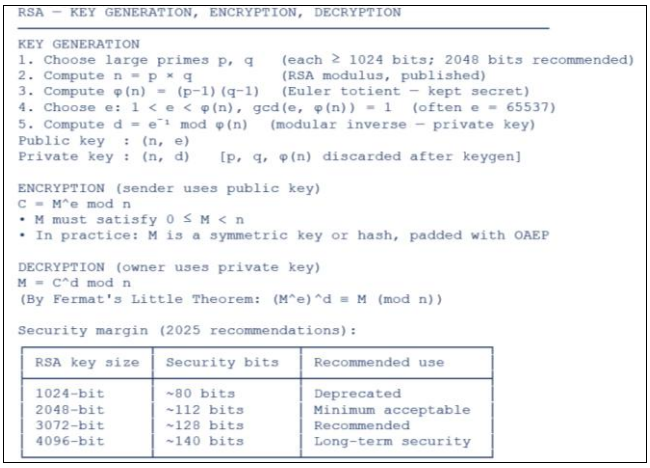
The One-Time Pad, independently described by Vernam (1919) and Shannon (1949) [32], is the only cipher proven to be information-theoretically secure. Rijmenants (2022) [29] provides a rigorous treatment contextualising OTP use in historical military and diplomatic settings. Shannon's proof rests on three conditions: the key must be (a) truly random, (b) at least as long as the plaintext, and (c) used only once.



The OTP is impractical for most modern use because key distribution is as difficult as distributing the message itself, and key reuse is catastrophically insecure. Tenzer (2022) [37] discusses OTP-inspired stream cipher designs that sacrifice information-theoretic security for practicality while retaining high computational security.

3.3 Rivest, Shamir, Adleman (RSA) Algorithm — Structure and Workflow

RSA, introduced by Rivest, Shamir, and Adleman in 1977, is the most widely deployed asymmetric cipher. Its security rests on the integer factorisation problem: given n = p × q (the product of two large primes), recovering p and q is computationally infeasible for classical computers when n is sufficiently large. Kolosov (2022) [15] provides an accessible step-by-step derivation of the RSA algorithm, and Boneh and Shoup (2020) [2] formalise its security under the random oracle model.

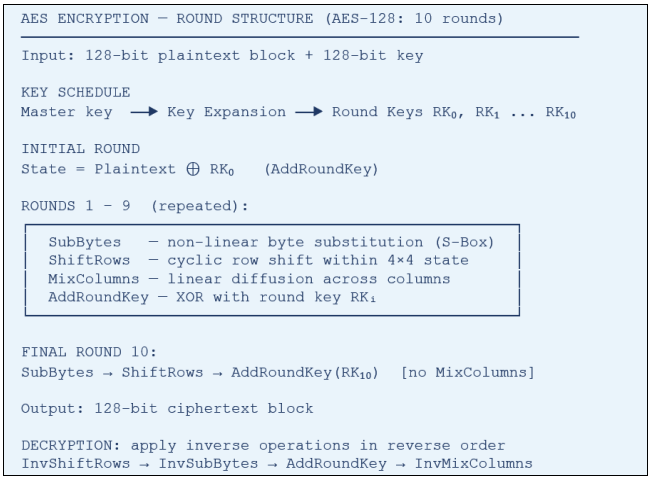


4. Symmetric Encryption Algorithms

Symmetric encryption uses a single shared secret key for both encryption and decryption. The principal challenge is key distribution: both parties must possess the same key before secure communication can begin. Once that key is shared (typically via an asymmetric key exchange), symmetric ciphers provide extremely efficient bulk data encryption. Teuscher (2026).

4.1 Advanced Encryption Standard (AES)

AES, standardised by NIST in 2001 (FIPS 197), is the global standard for symmetric encryption. It is a substitution-permutation network operating on 128-bit blocks with key sizes of 128, 192, or 256 bits. The number of rounds varies: 10 for AES-128, 12 for AES-192, and 14 for AES-256. Kaur and Singh (2021) [13] confirm AES-256 as the best-performing cipher across big-data workloads; Rahul and Kuppusamy (2021) [28] extend this finding to cloud image-security contexts.



4.2 AES Modes of Operation

A block cipher encrypts fixed-size blocks; real data is almost never exactly one block long. Modes of operation govern how a block cipher is applied to arbitrarily long messages. The choice of mode profoundly affects security properties (Dworkin, 2001; Boneh & Shoup, 2020) [6, 2].

The table below shows the AES Modes of Operation comparing four common modes of operation for block ciphers like AES — that is, different methods for applying the cipher to messages longer than one block.

1. **ECB (Electronic Codebook):** Encrypts each block independently with the same key. It's insecure because identical plaintext blocks produce identical ciphertext blocks, leaking patterns. No IV needed, and it's never recommended in practice.
2. **CBC (Cipher Block Chaining):** Each block is XOR'd with the previous ciphertext block before encryption. It's IND-CPA secure (a standard security notion meaning ciphertexts don't leak info about plaintexts under chosen-plaintext attack), but errors propagate and it can't be parallelized. Requires an IV. Used in file/disk encryption and legacy TLS.
3. **CTR (Counter Mode):** Encrypts an incrementing counter value to generate a keystream, then XORs it with the plaintext. It's IND-CPA secure and parallelizable, but the nonce must never repeat. Used for high-speed streaming and disk encryption.

4. **GCM (Galois/Counter Mode):** Combines CTR mode with GHASH authentication to provide AEAD (authenticated encryption with associated data) — both confidentiality and integrity. It's IND-CCA2 secure (stronger, resistant to chosen-ciphertext attacks) and is the preferred modern mode, used in TLS 1.3, IPsec, and SSH.

The overall point: the choice of mode matters a lot for security — ECB is broken, CBC and CTR give confidentiality only, and GCM is the modern standard because it adds integrity protection too.

Table 1: AES modes of operation: security properties and use cases

Mode	How It Works	Security Properties	IV Required?	Typical Use
ECB	Each block encrypted independently with same key	Insecure — identical blocks → identical ciphertext; patterns visible	No	Never recommended
CBC	Each block XOR'd with previous ciphertext before encryption	IND-CPA secure; error propagation; not parallelisable	Yes	File/disk encryption; TLS (legacy)
CTR	Encrypts counter value to produce keystream; XOR with plaintext	IND-CPA; parallelisable; nonce must never repeat	Yes (nonce)	High-speed streaming, disk encryption
GCM	CTR mode + GHASH authentication tag; provides AEAD	IND-CCA2; provides both confidentiality and integrity	Yes (nonce)	TLS 1.3, IPsec, SSH; preferred modern mode

4.3 AES-CBC Workflow (Step-by-Step)

```
AES-CBC ENCRYPTION
──────────────────────────────────────────────────────────────
Inputs: Plaintext M, Key K, Initialisation Vector IV (random, 128-bit)

Step 1 — Padding
Split M into 128-bit blocks: P₁, P₂ ... Pₙ
Pad final block Pₙ to 128 bits (PKCS#7)

Step 2 — First block
C₁ = AES_K( P₁ ⊕ IV )

Step 3 — Subsequent blocks (i = 2 ... n)
Cᵢ = AES_K( Pᵢ ⊕ Cᵢ₋₁ )

Step 4 — Transmit
Send: IV ‖ C₁ ‖ C₂ ‖ ... ‖ Cₙ
(IV is public; it need not be secret)

AES-CBC DECRYPTION
Step 1 — Parse received data: IV, C₁ ... Cₙ
Step 2 — P₁ = AES_K⁻¹(C₁) ⊕ IV
Step 3 — Pᵢ = AES_K⁻¹(Cᵢ) ⊕ Cᵢ₋₁  for i = 2 ... n
Step 4 — Remove padding from Pₙ
```

Lee and Sim (2021) [16] design a hardware-optimised variant of AES-CBC specifically for IoT devices, achieving a 28% reduction in gate count while maintaining equivalent security.

4.4 DES, 3DES, and Blowfish

DES (Data Encryption Standard), standardised in 1977, uses a 56-bit key and 16 Feistel rounds on 64-bit blocks. Its 56-bit key was demonstrated to be exhaustively searchable within 22 hours using the EFF DES Cracker in 1999. 3DES applies DES three times (Encrypt-Decrypt-Encrypt) with independent keys for an effective security of 112 bits, but its 64-bit block size introduces the Sweet32 birthday attack for long sessions. NIST deprecated 3DES in 2019. Both remain in legacy financial infrastructure (Kaur & Singh, 2021) [13].

Blowfish, designed by Schneier (1993), uses a variable key length (32–448 bits) and a Feistel structure with key-dependent S-boxes. Rahul and Kuppusamy (2021) [28] benchmark Blowfish as a competitive performer for image data but note its 64-bit block size carries the same birthday-bound weakness as 3DES for large data volumes. AES-256 dominates all modern deployments.

5. Asymmetric and Hybrid Encryption Systems

5.1 Asymmetric Encryption: Concepts and Key Exchange

Asymmetric (public-key) cryptography solves the key-distribution problem that afflicts symmetric schemes: rather than requiring a pre-shared secret, each party possesses a public key (freely distributed) and a private key (kept secret). Anyone can encrypt a message using the recipient's public key; only the recipient, holding the private key, can decrypt it.

```
ASYMMETRIC ENCRYPTION — GENERAL PSEUDOCODE
──────────────────────────────────────────────────────────────
SETUP
Bob runs KeyGen(1^λ) → (pk_Bob, sk_Bob)
Bob publishes pk_Bob; sk_Bob is never shared.

ENCRYPTION (Alice sends message to Bob)
1. Alice obtains Bob's authentic public key pk_Bob
2. Alice computes C = Enc(pk_Bob, M)
3. Alice sends C over insecure channel

DECRYPTION (Bob recovers message)
4. Bob computes M = Dec(sk_Bob, C)

KEY PROPERTIES
• Knowing pk_Bob and C reveals nothing about M without sk_Bob
• Recovering sk_Bob from pk_Bob requires solving a hard problem:
  RSA      → Integer Factorisation Problem (IFP)
  ECC      → Elliptic Curve Discrete Logarithm Problem (ECDLP)
  ElGamal → Discrete Logarithm Problem (DLP)
```

5.2 Elliptic Curve Cryptography (ECC)

ECC achieves equivalent security to RSA with dramatically shorter keys by basing its security on the elliptic curve discrete logarithm problem (ECDLP). An elliptic curve over a finite field is the set of points satisfying $y^2 = x^3 + ax + b \pmod p$. The discrete logarithm on this curve given points P and Q = kP, find k has no known subexponential algorithm for properly chosen curves (Stinson & Paterson, 2019) [36].

The main point: ECC gives the same security strength as RSA/DH but with much smaller keys, because its security relies on the elliptic curve discrete logarithm problem (ECDLP), which is harder to solve efficiently than the problems RSA/DH rely on — so no shortcut (subexponential algorithm) exists for well-chosen curves.

The table below shows five security levels (measured in bits) and the key size each system needs to achieve them:

Table 2: RSA/DH versus ECC key sizes for equivalent security levels (NIST SP 800-57, 2020)

Security Level	RSA/DH Key Size	ECC Key Size	Ratio
80 bits	1024 bits	160 bits	ECC is 6.4× smaller
112 bits	2048 bits	224 bits	ECC is 9.1× smaller
128 bits	3072 bits	256 bits	ECC is 12× smaller
192 bits	7680 bits	384 bits	ECC is 20× smaller
256 bits	15360 bits	521 bits	ECC is 29× smaller

5.3 Hybrid Encryption: Combining RSA/ECC with AES

Pure asymmetric encryption is computationally expensive—RSA encryption of large payloads is thousands of times slower than AES. The universally adopted solution is hybrid encryption: asymmetric cryptography for key exchange, symmetric cryptography for bulk data. Guru and Ambhaikar (2021) [10] present a detailed AES-RSA hybrid protocol; Lu and Mohamed (2021) [17] demonstrate its enterprise implementation.

```
HYBRID ENCRYPTION — COMPLETE PSEUDOCODE
──────────────────────────────────────────────────────────────
SENDER
1. Generate ephemeral symmetric key K_s  (e.g., AES-256, random)
2. Encrypt message M:  C_data = AES-GCM(K_s, M)
3. Encrypt K_s with recipient public key:
   C_key  = RSA-OAEP(pk_Bob, K_s)
4. Transmit:  C_key ‖ C_data  [+ IV, auth tag]

RECEIVER
5. Recover symmetric key:  K_s = RSA-OAEP-Dec(sk_Bob, C_key)
6. Decrypt data:           M   = AES-GCM-Dec(K_s, C_data)

PERFORMANCE
RSA-2048 encrypt (K_s, 32 bytes) : ~0.05 ms
AES-256-GCM encrypt (1 GB data)  : ~0.5 s   (with AES-NI)
RSA alone on 1 GB                : infeasible (hours)

SECURITY
• K_s is fresh for each session (forward secrecy if using ECDH)
• AES-GCM provides authenticated encryption (AEAD)
• RSA-OAEP achieves IND-CCA2 security under ROM
```

5.4 Diffie-Hellman Key Exchange

Diffie-Hellman (DH) key exchange, published in 1976, allows two parties to establish a shared secret over a public channel without prior communication. The elliptic curve variant (ECDH) is the dominant mechanism in TLS 1.3 and other modern protocols (Boneh & Shoup, 2020) [2]. The workflow is:

```
DIFFIE-HELLMAN KEY EXCHANGE
──────────────────────────────────────────────────────────────
PUBLIC PARAMETERS: prime p, generator g  (published)

ALICE                                  BOB
Choose secret a ← Zp*                  Choose secret b ← Zp*
Compute A = g^a mod p                  Compute B = g^b mod p
Send A ──────────────────────────►     Receives A
Receives B ◄──────────────────────     Send B
Compute K = B^a mod p                  Compute K = A^b mod p

Both parties now share K = g^(ab) mod p
(An eavesdropper sees g, p, A=g^a, B=g^b but cannot compute g^ab
 without solving the Discrete Logarithm Problem)

ECDH variant: replace (g, p, mod-exp) with elliptic curve point
multiplication
Security: ECDLP hard ⇒ 256-bit ECDH ≈ 128-bit security
```

6. Homomorphic Encryption

6.1 Definition and Taxonomy

Homomorphic encryption (HE) allows computation to be performed directly on ciphertexts, such that decrypting the result yields the same output as performing the same computation on the original plaintexts. Formally, for a homomorphic scheme with encryption Enc and operation.

6.2 Fully Homomorphic Encryption (FHE) — How It Works

Jain and Cherukuri (2023) [11] survey modern FHE schemes in detail. All practical FHE schemes follow a common architecture: they embed plaintext into a noisy ciphertext using an underlying hard problem (typically LWE or RLWE); each homomorphic operation adds noise; and bootstrapping periodically reduces the noise level to allow further operations.

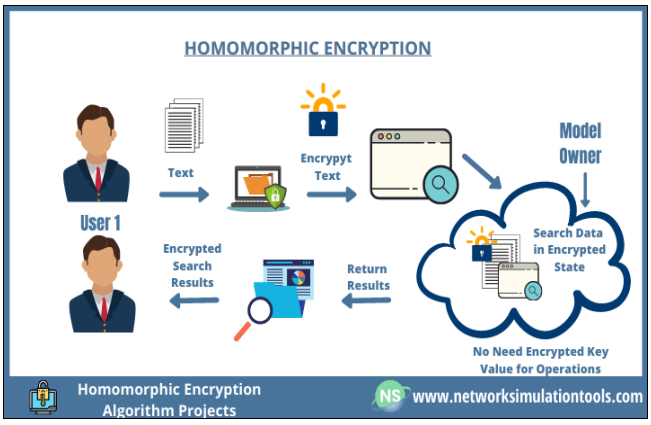


Fig 6: Homomorphic encryption operational process

6.3 Cheon-Kim-Kim-Song (CKKS) — Approximate Arithmetic for Machine Learning

The Cheon-Kim-Kim-Song (CKKS) scheme supports approximate arithmetic on real and complex numbers ideal for neural network inference, where exact integer arithmetic is unnecessary. Pulido-Gaytan *et al*. (2021) [27] apply CKKS to privacy-preserving neural network inference in cloud environments. The scheme encodes floating-point values as elements of a cyclotomic polynomial ring, enabling vectorised (SIMD-style) operations on batches of values simultaneously critical for achieving acceptable inference throughput.

Amorim and Costa (2023) [1] analyse Homomorphic encryption (HE) application to searchable encryption, showing that Partially Homomorphic Encryption (PHE) suffices for keyword matching and inner-product queries at substantially lower overhead than FHE, making it viable for encrypted database search today. Kim *et al*. (2021) [14] deploy Paillier PHE in an e-voting system: votes are encrypted individually (each as 0 or 1), tallied homomorphically (via ciphertext addition), and decrypted once at the end of polling, revealing the aggregate count without exposing individual ballots. The blockchain ledger provides public verifiability of the tally computation.

Chen (2024) [3] and Sánchez *et al*. (2021) [31] demonstrate lattice-based and blockchain-integrated HE frameworks respectively.

7. Post-Quantum Cryptography

7.1 The Quantum Threat Model

Quantum computers exploit superposition and entanglement to perform certain computations exponentially faster than classical machines. Two algorithms are most relevant to cryptography: Shor's algorithm factors integers and computes discrete logarithms in polynomial time, breaking RSA, Diffie–Hellman (DH), and Elliptic-Curve Cryptography (ECC); Grover's algorithm provides a quadratic speedup for unstructured search, halving the effective security of symmetric ciphers (requiring AES-256 rather than AES-128 for 128-bit post-quantum security). Mattsson, Smeets, and Westerbaan (2021) [18] provide a thorough treatment.

7.2 NIST Post-Quantum Standardisation

NIST finalised its first three post-quantum standards in 2024: ML-KEM (CRYSTALS-Kyber, FIPS 203) for key encapsulation, ML-DSA (CRYSTALS-Dilithium, FIPS 204) for digital signatures, and SLH-DSA (SPHINCS+, FIPS 205) for hash-based signatures. A fourth standard, FN-DSA (FALCON), is also nearing finalisation (NIST, 2024) [25].

The table below summarizes the standards NIST finalized in 2024 (plus one leading candidate) to replace current cryptography that would be broken by quantum computers.

Background: RSA and ECC (from your earlier tables) are vulnerable to quantum algorithms (Shor's algorithm), so NIST selected new "post-quantum" algorithms built on math problems that quantum computers can't efficiently solve.

The table compares four schemes across five dimensions:

Table 3: NIST post-quantum standards and leading candidates (2024)

Standard	Underlying Problem	Security Basis	Key Size (bytes)	Classical Equivalent
ML-KEM (Kyber)	Module LWE	Hard lattice problems believed resistant to quantum algorithms	pk: 800–1568 B	RSA-3072
ML-DSA (Dilithium)	Module LWE / SIS	Module Short Integer Solution; lattice-based signature	pk: 1312–2592 B	ECDSA-256
SLH-DSA (SPHINCS+)	Hash function security	Collision and preimage resistance of SHA-256/SHAKE	pk: 32–64 B	RSA-2048 (sig)
BIKE (candidate)	Quasi-cyclic codes	Decoding random binary quasi-cyclic codes (NP-hard)	pk: ~1.5 KB	RSA-3072

Key Takeaways:

1. Three of these (ML-KEM, ML-DSA, SLH-DSA) are now official NIST standards; a fourth (FN-DSA/FALCON) is close behind, and BIKE remains a candidate under evaluation.
2. Most rely on lattice-based math (Kyber, Dilithium), which is efficient but relatively new territory for cryptanalysis.
3. SPHINCS+ takes a more conservative approach using only hash functions — slower/bulkier signatures but very well-trusted security foundations.
4. The "Classical Equivalent" column lets you gauge roughly how much security each post-quantum key size buys, compared to the RSA/ECC key sizes you saw in Table 2.

7.3 Lattice-Based Cryptography

Lattice-based cryptography bases its security on the Learning with Errors (LWE) problem and its ring variant (RLWE). Both are believed hard for quantum computers. Chen (2024) [3] provides a rigorous treatment of RLWE-based homomorphic encryption.

```
LEARNING WITH ERRORS (LWE) — ALGORITHM
──────────────────────────────────────────────────────────────

PUBLIC PARAMETERS: n (dimension), q (modulus), χ (error distribution)

PRIVATE KEY: secret vector s ← Zq^n

KEY GENERATION (Regev encryption):
Sample random matrix A ← Zq^(m×n)
Sample small error vector e ← χ^m
Compute b = A·s + e   (mod q)
Public key: (A, b)    Private key: s

ENCRYPTION of bit μ ∈ {0,1}:
Choose random r ← {0,1}^m
c₁ = A^T · r   (mod q)
c₂ = b^T · r + ⌊q/2⌋·μ   (mod q)
Ciphertext: (c₁, c₂)

DECRYPTION:
Compute v = c₂ − s^T·c₁ = ⌊q/2⌋·μ + e'   (mod q)
If v close to ⌊q/2⌋: output μ = 1; else μ = 0

SECURITY: An adversary distinguishing (A, A·s+e) from (A, uniform)
must solve LWE — believed hard for all known quantum algorithms.
```

7.4 Code-Based Cryptography

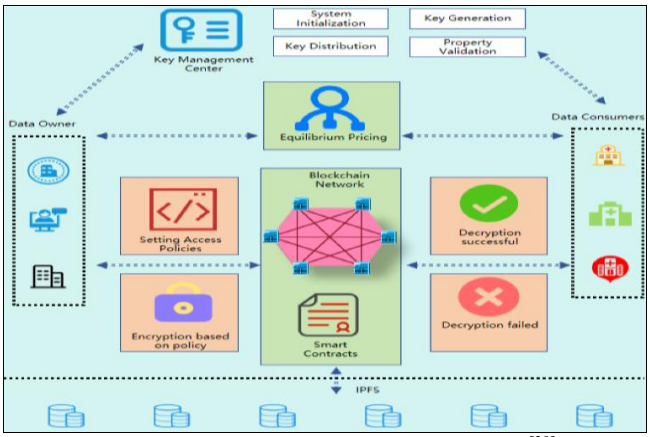
Code-based cryptography builds public-key encryption on the hardness of decoding a general linear error-correcting code. The idea traces back to Robert McEliece in 1978, who was the first to introduce a public-key cryptosystem based on encoding plaintext as codewords from the family of Goppa codes. Goppa codes were chosen because they have a fast decoding algorithm and are "easy to generate but hard to find," making the underlying code structure look essentially random once it's disguised. Singh, H. (2019) [33].

Weger, Geihs, and Augot (2022) [38] survey code-based cryptography, originating with McEliece (1978). Security derives from the NP-hardness of decoding random linear codes. The workflow involves encoding the message as a valid codeword, then intentionally adding errors; only the holder of the private decoding key (a structured code with an efficient decoder) can recover the original message.

```
McELIECE ENCRYPTION — ALGORITHM
──────────────────────────────────────────────────────────────

KEY GENERATION:
1. Select binary Goppa code C with efficient decoder D
   (t-error-correcting [n, k]-code)
2. Generate parity-check matrix H (private)
3. Compute public generator matrix G' = S·G·P
   where S = invertible (k×k) scrambling matrix
         P = (n×n) permutation matrix
Public key: G'   Private key: (S, G, P, D)

ENCRYPTION of message m (k-bit vector):
1. Compute c = m·G'  (encode as codeword)
2. Add random error vector e of weight t: c' = c + e
3. Ciphertext: c'

DECRYPTION:
1. Remove permutation: c'' = c'·P⁻¹
2. Apply Goppa decoder D to correct ≤t errors: c = decode(c'')
3. Recover message: m = c·(G)⁻¹·S⁻¹

NOTE: Modern variants (BIKE, HQC) use quasi-cyclic codes
to reduce public key size from ~1 MB to ~1.5 KB.
```

8. Blockchain-Integrated Encryption Frameworks

Blockchain technology combines distributed consensus with cryptographic primitives (hash functions, digital signatures, Merkle trees) to create tamper-evident, decentralised ledgers. When combined with encryption, blockchain enables privacy-preserving architectures where data integrity and auditability are guaranteed without centralised trust. Multiple reviewed sources exploit this synergy. Fig 7 depicts the block-chain encrypted data sharing architecture.



Source: Wu, J., Bian, Z., Gao, H., & Wang, Y. (2025) [39]

Fig 7: Blockchain-integrated encrypted data sharing architecture

Guo, Wang, and Huang (2021) [9] implement this pattern for distributed cloud data sharing, demonstrating resilience to single-point-of-failure attacks. Sánchez *et al*. (2021) [31] extend it with homomorphic encryption to enable private aggregation: individual encrypted data contributions are aggregated homomorphically before any decryption, so not even the aggregator learns individual values. Lee and Sim (2021) [16] deploy a lightweight variant using a DAG-based blockchain (higher throughput, lower latency than linear-chain blockchains) for IoT device authentication combined with AES-CBC payload encryption.

9. Applications Across Domains

9.1 Cloud Computing

Cloud deployments require encryption at rest (protecting stored data from unauthorised server-side access), in transit (TLS/HTTPS), and increasingly in use (homomorphic encryption or secure multi-party computation for cloud-side analytics). Rahul and Kuppusamy (2021) [28] benchmark AES-256-CBC and AES-256-GCM as the recommended baseline for cloud image storage; Pulido-Gaytan *et al*. (2021) [27] demonstrate CKKS-based HE for privacy-preserving inference; and Amorim and Costa (2023) [1] cover encrypted cloud-based search.

9.2 Internet of Things

IoT environments impose extreme resource constraints: microcontrollers with kilobytes of RAM, milliwatt power budgets, and millisecond timing requirements. Lee and Sim (2021) [16] address this with a hardware-optimised AES-CBC implementation on FPGA, achieving 1.2 Gbps throughput at 28% reduced gate count. Their integration with a DAG-blockchain provides device authentication. Key management at IoT scale bootstrapping trust relationships, rotating keys across millions of devices, revoking compromised nodes remains an open engineering challenge.

9.3 Electronic Voting

Kim *et al*. (2021) [14] demonstrate Paillier PHE plus blockchain for e-voting: encrypted ballots are cast on a smart contract, tallied homomorphically (no individual ballot ever decrypted during counting), and verified publicly against the blockchain ledger. The combination addresses ballot secrecy, vote integrity, and universal verifiability simultaneously.

9.4 Healthcare and Privacy-Preserving Analytics

Medical data is among the most sensitive personal information. Federated learning combined with HE allows hospitals to collaboratively train machine learning models without sharing raw patient records (Pulido-Gaytan *et al*., 2021) [27]. The CKKS scheme's support for approximate arithmetic makes it compatible with the floating-point computations typical of neural network training and inference.

9.5 Financial Services and Smart Grids

Sánchez *et al*. (2021) [31] apply blockchain-HE integration to smart-grid power consumption data, demonstrating GDPR compliance while enabling aggregate analytics. Financial applications include encrypted settlement systems, privacy-preserving credit scoring, and secure multi-party computation for risk aggregation across institutions without revealing proprietary position data.

10. Application Domain

the table below ties together everything from the earlier tables (AES modes, RSA/ECC sizing, post-quantum standards, homomorphic encryption) into decision guidance showing that there's no single best algorithm; the right choice depends heavily on constraints like speed, device power, need for computation-on-encrypted-data, or long-term quantum threat exposure.

Rather than comparing algorithms head-to-head (like the earlier tables), this table answers: "given my use case, which crypto approach should I actually pick?"

Table 4: Best-suited encryption technique in application domain

Application Domain	Best-Suited Technique(s)	Why It Wins Here	Avoid / Not Recommended
Bulk data storage & data-at-rest (cloud, enterprise, medical records)	AES-256-GCM / CCM (symmetric, authenticated)	Hardware-accelerated (AES-NI) throughput with single-pass confidentiality and integrity; scales to large data volumes	RSA/ECC alone -- orders of magnitude too slow for bulk payloads
TLS / network transport & key exchange	Hybrid: ECDHE (ECC) key agreement + AES-GCM payload	Small ECC keys give fast, forward-secure key agreement while AES carries the bulk cipher load	Static RSA key transport -- no forward secrecy
Resource-constrained IoT devices	Lightweight AES-CBC/CTR + ECC for authentication	Minimal RAM and power footprint; ECC's small key size fits microcontroller constraints	RSA (key/computation overhead too high); FHE (infeasible on-device)
Privacy-preserving cloud analytics & ML inference	CKKS (leveled/fully homomorphic encryption)	Only paradigm enabling computation directly on ciphertext; approximate arithmetic suits floating-point ML workloads	Symmetric/asymmetric schemes -- require decryption before computing, breaking the privacy guarantee
Electronic voting	Paillier PHE + blockchain ledger	Additive homomorphism allows tallying encrypted ballots without individual decryption; blockchain adds public verifiability	FHE -- unnecessary computational cost for simple aggregation
Federated healthcare analytics	CKKS FHE	Enables multi-institution model training without exposing raw patient records	Plain symmetric encryption -- breaks privacy once decrypted for computation
Financial settlement & smart-grid aggregation	Blockchain-integrated HE (Paillier/CKKS)	Supports aggregate computation and auditability without revealing individual positions or consumption data	Unencrypted aggregation -- fails confidentiality and regulatory requirements
Long-term sensitive data at risk of harvest-now-decrypt-later (government, defence, archival)	ML-KEM (Kyber) / lattice-based PQC	Resistant to Shor's algorithm; standardised by NIST for post-quantum key encapsulation	RSA/ECC -- broken once cryptographically relevant quantum computers exist
Legacy systems pending migration	AES-256 as the replacement target	Modern, vetted, quantum-resistant against Grover's algorithm at 256-bit key size	3DES, DES, RC4 -- deprecated; vulnerable to Sweet32, brute force, or keystream bias
Digital signatures & authentication	ECC (ECDSA/EdDSA)	Smaller keys and faster signing/verification than RSA at equivalent security levels	RSA -- larger keys and signatures for the same security level

These mappings reflect the dominant constraint of each domain rather than an absolute rule; several deployments reviewed here combine two or more techniques (e.g., hybrid key exchange, or blockchain plus homomorphic encryption) precisely because a single primitive rarely satisfies every requirement of a real system.

11. Comparative Analysis

The table below sorts algorithms by **type** and shows a clear pattern:

1. **Symmetric** (AES-256-GCM) offers strong, quantum-resistant security for bulk data the workhorse for encrypting large amounts of data efficiently.
2. **3DES** is flagged as deprecated/legacy its 80-bit effective security is broken by modern attacks (Sweet32) and it's not quantum-safe.
3. **Classical asymmetric** algorithms (RSA-2048, ECC P-256) are solid *today* for key exchange and TLS, but both are explicitly vulnerable to quantum attacks (Shor's algorithm) this is the motivation for post-quantum migration.
4. **Homomorphic encryption** (Paillier, CKKS/BFV) allows computation on encrypted data without decrypting it useful for private machine learning, voting, secure aggregation. Paillier is quantum-vulnerable; CKKS/BFV is (conditionally) quantum-safe since it's lattice-based.
5. **Post-quantum (PQ)** schemes (ML-KEM/Kyber, BIKE) are the future-proof options explicitly marked "Yes" for quantum safety, meant to replace RSA/ECC for key exchange.

Overall takeaway: The table maps the transition happening in cryptography from classical symmetric/asymmetric algorithms (some already weak, others quantum-vulnerable) toward lattice- and code-based post-quantum alternatives, while homomorphic encryption fills a separate niche (computing on encrypted data) rather than replacing standard encryption.

Table 5: Comparative overview of major encryption algorithms (2025 security recommendations)

Algorithm	Type	Key Size	Block/Op Size	Security (bits)	Quantum-Safe?	Best Use Case
AES-256-GCM	Symmetric	256 bits	128-bit block	~128 (PQ)	Yes (Grover: halved)	Bulk data, TLS, cloud
3DES	Symmetric	112 eff. bits	64-bit block	~80 (Sweet32)	No (deprecated)	Legacy only
RSA-2048	Asymmetric	2048 bits	256-byte block	~112 (classical)	No (Shor's)	Key exchange, TLS
ECC P-256	Asymmetric	256 bits	64-byte key	~128 (classical)	No (Shor's)	TLS, mobile, IoT
Paillier PHE	HE (partial)	2048 bits	Variable	~112 (classical)	No (Shor's)	Voting, aggregation
CKKS / BFV FHE	HE (full)	RLWE-based	Polynomial ring	128 (classical+PQ)	Yes (RLWE hard)	Private ML, cloud
ML-KEM (Kyber)	PQ / Lattice	800–1568 B pk	Encapsulation	128–256	Yes	Post-quantum KEM
BIKE (code-based)	PQ / Code	~1.5 KB pk	Encapsulation	128	Yes	PQ key exchange

12. Conclusion

This review has systematically surveyed twenty-eight authoritative sources covering encryption techniques from classical one-time pads to post-quantum lattice schemes, providing for each paradigm both conceptual explanation and structured process diagrams that illustrate how encryption and decryption actually operate. The principal findings are as follows.

Symmetric encryption, anchored by AES-256 in authenticated modes (GCM, CCM), provides efficient, quantum-safe confidentiality for bulk data and remains the recommended standard for all new deployments. DES and 3DES are deprecated; their continued presence in legacy systems represents a persistent risk that organizations should prioritize for migration.

Asymmetric cryptography RSA, ECC, and Diffie-Hellman enables public-key infrastructure and key exchange but is vulnerable to quantum attack via Shor's algorithm. These systems must be replaced or supplemented with post-quantum alternatives. The hybrid encryption pattern (asymmetric for key transport, symmetric for data) remains the correct architectural choice and will continue to be so in the post-quantum era, with classical RSA/ECC replaced by ML-KEM or equivalent.

Homomorphic encryption has progressed from a theoretical construct to a set of deployed libraries capable of supporting real applications in privacy-preserving machine learning, encrypted database search, and verifiable aggregation. Performance remains the limiting factor; hardware acceleration and algorithmic advances are narrowing the gap Post-quantum cryptography has achieved its first standardization milestones; the engineering challenge now shifts to global migration. Lattice-based and code-based schemes offer the strongest combination of security and practical performance. The transition is urgent: encrypted data captured today and stored for future decryption is already at risk from future quantum attackers.

No single encryption technique satisfies all requirements simultaneously. Practitioners should treat encryption selection as an architectural decision requiring careful consideration of threat model, deployment context, performance budget, and regulatory environment. This review provides the conceptual and technical foundation for informed, confident decision-making across all these dimensions.

Table 6: Abbreviation

Abbreviation	Full Term
AES	Advanced Encryption Standard
GCM	Galois/Counter Mode
3DES	Triple Data Encryption Standard
PQ	Post-Quantum
TLS	Transport Layer Security
RSA	Rivest–Shamir–Adleman (named after its inventors)
ECC	Elliptic Curve Cryptography
IoT	Internet of Things
HE	Homomorphic Encryption
PHE	Partial Homomorphic Encryption
FHE	Fully Homomorphic Encryption
CKKS	Cheon-Kim-Kim-Song (a homomorphic encryption scheme, named after its creators)
BFV	Brakerski/Fan-Vercauteren (a homomorphic encryption scheme, named after its creators)
RLWE	Ring Learning With Errors
ML-KEM	Module-Lattice-based Key Encapsulation Mechanism
BIKE	Bit Flipping Key Encapsulation
Sweet32	Name of a specific attack exploiting 64-bit block cipher collisions (not an acronym — named for the "birthday bound" sweet spot around 2^32 blocks)
Shor's algorithm	Named after mathematician Peter Shor — a quantum algorithm that efficiently factors large numbers and solves discrete logarithms, breaking RSA/ECC
Grover's algorithm	Named after physicist Lov Grover — a quantum algorithm that speeds up brute-force search, roughly halving effective symmetric key strength

Expansions of abbreviations.

13. References

1. Amorim I, Costa I. Homomorphic encryption: An analysis of its applications in searchable encryption. arXiv, 2023. Doi: https://arxiv.org/abs/2306.14407
2. Boneh D, Shoup V. A graduate course in applied cryptography (Version 0.6). Stanford University, 2020. https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_6.pdf
3. Chen ACH. Homomorphic encryption based on lattice post-quantum cryptography. arXiv, 2024. https://arxiv.org/abs/2501.03249
4. Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology - ASIACRYPT 2017. Springer, 2017, 409-437. Doi: https://doi.org/10.1007/978-3-319-70694-8_15
5. Diffie W, Hellman ME. New directions in cryptography. IEEE Transactions on Information Theory. 1976; 22(6):644-654.
6. Dworkin M. Recommendation for block cipher modes of operation: Methods and techniques (NIST SP 800-38A). National Institute of Standards and Technology, 2001. Doi: https://doi.org/10.6028/NIST.SP.800-38A
7. Fortune Business Insights. Encryption software market size, share & COVID-19 impact analysis, 2024. https://www.fortunebusinessinsights.com/encryption-software-market-102338.html

8. Gentry C. A fully homomorphic encryption scheme [Doctoral dissertation, Stanford University], 2009. https://crypto.stanford.edu/craig/craig-thesis.pdf
9. Guo Y, Wang S, Huang J. A blockchain-assisted framework for secure and reliable data sharing in distributed systems. EURASIP Journal on Wireless Communications and Networking. 2021; 169. Doi: https://doi.org/10.1186/s13638-021-02041-y
10. Guru A, Ambhaikar A. AES and RSA-based hybrid algorithms for message encryption & decryption. Information Technology in Industry. 2021; 9(1). Doi: https://doi.org/10.17762/itii.v9i1.129
11. Jain N, Cherukuri AK. Revisiting fully homomorphic encryption schemes. arXiv, 2023. https://arxiv.org/abs/2305.05904
12. Kahn D. The codebreakers: The story of secret writing (Rev. ed.). Scribner, 1996.
13. Kaur M, Singh P. A comparative survey on data encryption techniques: Big data perspective. Materials Today: Proceedings. 2021; 46(20):11035-11039. Doi: https://doi.org/10.1016/j.matpr.2021.02.153
14. Kim H, Kim KE, Park S, Sohn J. E-voting system using homomorphic encryption and blockchain technology to encrypt voter data. arXiv, 2021. https://arxiv.org/abs/2111.05096
15. Kolosov P. RSA encryption: Behind the scenes. Internet Archive, 2022. https://dn720003.ca.archive.org/0/items/rsa-encryption-behind-the-scenes/RSAEncryptionExplained.pdf
16. Lee S-W, Sim K-B. Design and hardware implementation of a simplified DAG-based blockchain and new AES-CBC algorithm for IoT security. Electronics. 2021; 10(9):1127. Doi: https://doi.org/10.3390/electronics10091127
17. Lu Z, Mohamed H. A complex encryption system design implemented by AES. Journal of Information Security. 2021; 12(2):177-187. Doi: https://doi.org/10.4236/jis.2021.122009
18. Mattsson JP, Smeets B, Westerbaan B. Quantum-resistant cryptography. arXiv, 2021. https://arxiv.org/pdf/2112.00399
19. Menezes AJ, Van Oorschot PC, Vanstone SA. Handbook of applied cryptography. CRC Press, 1996.
20. National Institute of Standards and Technology. Data Encryption Standard (DES) (FIPS PUB 46-3). U.S. Department of Commerce, 1999.
21. National Institute of Standards and Technology. Advanced Encryption Standard (AES) (FIPS PUB 197). U.S. Department of Commerce, 2001.
22. National Institute of Standards and Technology. Secure Hash Standard (SHS) (FIPS PUB 180-4). U.S. Department of Commerce, 2015.
23. National Institute of Standards and Technology. Post-quantum cryptography standardization. U.S. Department of Commerce, 2016.
24. National Institute of Standards and Technology. Recommendation for key management, Part 1 (NIST SP 800-57 Rev. 5), 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5
25. National Institute of Standards and Technology. Post-quantum cryptography standards (FIPS 203, 204, 205). U.S. Department of Commerce, 2024. https://www.nist.gov/pqcrypto
26. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cryptology – EUROCRYPT '99. Springer, 1999, 223-238. Doi: https://doi.org/10.1007/3-540-48910-X_16
27. Pulido-Gaytan B, Tchernykh A, Cortés-Mendoza JM, Babenko M, Radchenko G, Avetisyan A, *et al*. Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities. Peer-to-Peer Networking and Applications. 2021; 14:1666-1691. Doi: https://doi.org/10.1007/s12083-021-01076-8
28. Rahul B, Kuppusamy K. Efficiency analysis of cryptographic algorithms for image data security at cloud environment. IETE Journal of Research. 2021; 69:1-12. Doi: https://doi.org/10.1080/03772063.2021.1990141
29. Rijmenants D. Secure communications with the One-Time Pad cipher. Internet Archive, 2022. https://dn721604.ca.archive.org/0/items/secure-communications-with-the-one-time-pad-cipher-by-dirk-rijmenants-27-feb-2022/
30. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 1978; 21(2):120-126. Doi: https://doi.org/10.1145/359340.359342
31. Sánchez D, Farinosi M, *et al*. Privacy-enhancing distributed protocol for data aggregation based on blockchain and homomorphic encryption. Information Processing & Management. 2021; 58(6):102745. Doi: https://doi.org/10.1016/j.ipm.2021.102745
32. Shannon CE. Communication theory of secrecy systems. Bell System Technical Journal. 1949; 28(4):656-715. Doi: https://doi.org/10.1002/j.1538-7305.1949.tb00928.x
33. Singh H. Code based cryptography: Classic McEliece. arXiv, 2019. https://arxiv.org/abs/1907.12754
34. Singh S. The code book: The science of secrecy from ancient Egypt to quantum cryptography. Doubleday, 1999.
35. Stallings W. Cryptography and network security: Principles and practice (8th ed.). Pearson, 2020.
36. Stinson DR, Paterson M. Cryptography: Theory and practice (4th ed.). CRC Press, 2019.
37. Tenzer T. Super secreto: The third epoch of cryptography. Internet Archive, 2022. https://dn721903.ca.archive.org/0/items/cryptography-tenzer/Tenzer_Theo_-_Cryptography_9783755761174.pdf
38. Weger V, Geihs M, Augot D. A survey on code-based cryptography. arXiv, 2022. https://arxiv.org/pdf/2201.07119
39. Wu J, Bian Z, Gao H, Wang Y. A Blockchain-Based Secure Data Transaction and Privacy Preservation Scheme in IoT System. Sensors. 2025; 25(15):4854. Doi: https://doi.org/10.3390/s25154854