Cryptography: Symmetric and Asymmetric Encryption

Comprehensive Reference for Overleaf

Generated Documentation

January 23, 2026

Abstract

This document provides a detailed, formal reference on cryptography focusing on symmetric and asymmetric encryption. It covers theoretical foundations, practical algorithms, operational considerations (modes of operation, padding, IVs), key management, common attacks, and modern usage patterns including hybrid cryptosystems and TLS. The document is formatted for Overleaf and intended for system administrators, developers, security practitioners, and students.

Contents

1. Introduction to Cryptography

1.1. Definition and Goals

Cryptography is the science of securing information and communication through the use of mathematical techniques. Core security goals include:

- **Confidentiality**: Prevent unauthorized disclosure of information.
- **Integrity**: Detect and prevent unauthorized modification.
- **Authentication**: Verify identities of parties.
- **Non-repudiation**: Prevent entities from denying actions.

1.2. Basic Concepts

Plaintext The original readable message.
Ciphertext The encrypted message.
Key Secret (or private/public) value used by the algorithm.
Cipher The algorithm used for encryption or decryption.
Nonce Number used once, often an IV or counter to ensure uniqueness.

2. Symmetric Encryption

2.1. Overview

Symmetric encryption uses a single shared secret key for both encryption and decryption. It is efficient for bulk data encryption but requires secure key distribution between parties.

2.2. Categories

- **Block Ciphers** operate on fixed-size blocks (e.g., AES with 128-bit blocks).
- **Stream Ciphers** produce a keystream combined with plaintext (e.g., AES-CTR, historically RC4).

2.3. Block Cipher Modes of Operation

Block ciphers by themselves encrypt a single block; modes define how to apply the cipher to sequences of blocks. Key modes include:

ECB (Electronic Codebook) Encrypts blocks independently. **Weakness:** Identical plaintext blocks map to identical ciphertext blocks; not semantically secure.
CBC (Cipher Block Chaining) Uses an IV and XORs plaintext block with previous ciphertext block before encryption. Requires random IV.
CFB / OFB Convert block cipher into stream-like behavior; have specific error propagation properties.
CTR (Counter) Uses a counter+nonce for each block and encrypts the counter to produce a keystream; supports parallelism.
GCM (Galois/Counter Mode) CTR combined with Galois-field authentication to provide Authenticated Encryption with Associated Data (AEAD). Preferred for performance + authentication.

2.4. Padding Schemes

Block ciphers often require padding when plaintext length is not a multiple of block size. Common schemes:

- PKCS#7 (widely used): pad with bytes each equal to the number of padding bytes.
- ANSI X.923, ISO/IEC 7816-4, Zero padding (zero padding ambiguous for text ending with zero byte).

2.5. Initialization Vector (IV) and Nonces

IV / nonce ensures that identical plaintext encrypted multiple times yields different ciphertexts. Requirements:

- IV must be unpredictable/random for CBC.
- Nonce must be unique for CTR/GCM; reuse can be catastrophic (reveals keystream).

2.6. Common Symmetric Algorithms

AES (Advanced Encryption Standard) Block cipher with 128-bit block size; key sizes 128, 192, 256. Standard for modern encryption.

DES / 3DES Deprecated (DES: 56-bit key) or legacy (3DES) due to insufficient key sizes and performance.

ChaCha20 Stream cipher designed as an alternative to AES-CTR with high performance on software; often paired with Poly1305 for AEAD (ChaCha20-Poly1305).

RC4 Historical stream cipher; considered insecure and deprecated.

2.7. Authenticated Encryption

Authenticated Encryption (AE) provides both confidentiality and integrity. AEAD constructions (e.g., AES-GCM, ChaCha20-Poly1305) are recommended over encrypt-then-MAC ad-hoc designs.

2.8. Practical Examples (OpenSSL)

Encrypt a file with AES-256-CBC:

```
# Encrypt
openssl enc -aes-256-cbc -salt -in plaintext.bin -out ciphertext.bin -pass pass:
# Decrypt
openssl enc -d -aes-256-cbc -in ciphertext.bin -out decrypted.bin -pass pass:MyS
```

Encrypt with AES-256-GCM (using openssl CLI requires attention to IV and tag handling):

```
# Example requires careful handling; many prefer libs that expose AEAD APIs.
```

2.9. Key Management Considerations

Symmetric keys require secure generation (CSPRNG), secure storage (HSMs, KMS), secure distribution (out-of-band, key-exchange protocols), rotation, and revocation policies.

2.10. Use Cases

- Disk encryption (LUKS uses AES).
- TLS record encryption (bulk encryption in TLS session).
- VPN tunnels (IPsec ESP, WireGuard uses symmetric primitives).
- Database encryption at rest.

3. Asymmetric Encryption (Public-Key Cryptography)

3.1. Overview

Asymmetric cryptography uses a key pair: a public key (distributable) and a private key (kept secret). Asymmetric algorithms support encryption, key exchange, and digital signatures.

3.2. Core Algorithms

RSA Widely used for encryption and signatures. Security depends on integer factorization difficulty. Key sizes commonly 2048 or 3072 bits; 4096 bits for long-term security.

Diffie–Hellman (DH) Key exchange protocol; classical (modular exponentiation) and elliptic-curve variants (ECDH).

Elliptic Curve Cryptography (ECC) Uses elliptic curves for equivalent security with smaller key sizes (e.g., secp256r1, secp384r1). Faster and smaller keys than RSA for comparable security.

ElGamal Public-key encryption based on discrete logarithms; less common directly but used in certain protocols.

3.3. Digital Signatures

Digital signatures provide integrity and authentication.

- RSA signatures (PKCS#1 v1.5 or PSS recommended)
- DSA (Digital Signature Algorithm)
- ECDSA (Elliptic Curve Digital Signature Algorithm)

3.4. Key Sizes and Security

Asymmetric algorithms require much larger keys than symmetric for equivalent security. Representative equivalences:

Symmetric key size	Asymmetric RSA equivalent	ECC curve approx
128-bit	RSA 3072-bit (approx)	secp256r1
256-bit	RSA 15360-bit	secp521r1

Table 1: Approximate key strength equivalences

3.5. Operations and Use Cases

- **Encryption/Decryption:** Public key encrypts, private key decrypts (RSA).
- **Digital Signatures:** Private key signs, public key verifies.
- **Key Exchange / Establishing Shared Secret:** DH / ECDH used to agree on symmetric keys for bulk encryption.

3.6. Hybrid Encryption

Because asymmetric encryption is computationally expensive, practical systems use hybrid encryption: generate a random symmetric session key, encrypt data with a symmetric algorithm, and encrypt the session key with the recipient's public key.

3.7. Public Key Infrastructure (PKI)

PKI provides a framework for issuance, distribution, and revocation of certificates binding identities to public keys. Components:

- Certificate Authorities (CAs)
- Certificate Signing Requests (CSRs)
- X.509 certificates
- Certificate Revocation Lists (CRLs) and OCSP

3.8. Practical Examples (OpenSSL)

Generate RSA key and self-signed certificate:

```
# Generate private key
openssl genpkey -algorithm RSA -out rsa_private.pem -pkeyopt rsa_keygen_bits:204
# Generate public key / certificate signing request
openssl req -new -key rsa_private.pem -out req.csr -subj "/CN=example.com"
# Self-sign (for test)
openssl x509 -req -in req.csr -signkey rsa_private.pem -out cert.pem -days 365
```

Perform hybrid encryption (conceptual):

1. Generate symmetric key (AES-256).
2. Encrypt message with AES-256-GCM.
3. Encrypt AES key with recipient's RSA public key (RSA-OAEP recommended).
4. Package ciphertext, encrypted key, IV, and authentication tag.

4. Cryptanalysis and Common Attacks

4.1. Types of Attacks

- **Brute force**: Try all keys; mitigated by sufficiently large keys.
- **Cryptanalytic attacks**: Mathematical weaknesses (e.g., differential / linear cryptanalysis against block ciphers).
- **Side-channel attacks**: Timing, power analysis, electromagnetic leaks.
- **Replay attacks**: Reusing valid transmissions to deceive systems; mitigated by nonces/-timestamps.
- **Chosen plaintext / ciphertext attacks**: Adaptive adversary models that can reveal weaknesses in certain modes.
- **Man-in-the-Middle (MitM)**: Intercepts and alters key exchange unless authenticated.

4.2. Common Implementation Pitfalls

- Reusing IVs or nonces with CTR/GCM (catastrophic).
- Using ECB mode for structured data.
- Roll-your-own cryptography (insecure).
- Inadequate randomness for keys/IVs.
- Improper error handling leaking side-channel information.

5. Protocols and Real-World Systems

5.1. TLS / SSL

Transport Layer Security (TLS) uses hybrid cryptography: asymmetric primitives for authentication and key exchange; symmetric ciphers for data transport; MACs or AEAD for integrity.

Modern versions (TLS 1.2/1.3) favor AEAD ciphers (AES-GCM, ChaCha20-Poly1305) and secure key-exchange mechanisms (ECDHE).

5.2. SSH

Secure Shell uses a combination of asymmetric keys for host/user authentication and symmetric ciphers for session confidentiality.

5.3. PGP / OpenPGP

Provides end-to-end encryption for email and files; supports signatures, encryption, and web-of-trust models.

5.4. Disk Encryption

LUKS/dm-crypt use symmetric ciphers; keys often derived from passphrases using PBKDF2/scrypt/Argon2.

6. Best Practices

- Use well-vetted libraries and protocols (OpenSSL libs, libsodium, BoringSSL, NaCl/Libsodium).
- Prefer AEAD modes (AES-GCM, ChaCha20-Poly1305).
- Use authenticated key-exchange (ECDHE) and avoid static RSA key exchange.
- Ensure secure random number generation (OS CSPRNG: /dev/urandom, getrandom).
- Rotate keys and maintain key lifecycle management.
- Protect private keys (HSM, TPM, KMS) and enforce least privilege.
- Use modern curves (e.g., X25519 for ECDH) and safe padding (RSA-OAEP, RSA-PSS).

7. Appendices

7.1. Appendix A: Common OpenSSL Commands

```
# Generate AES key
openssl rand -out aes.key 32
# Encrypt with AES-256-CBC
openssl enc -aes-256-cbc -salt -in file.txt -out file.enc -pass file:./aes.key
# Decrypt
openssl enc -d -aes-256-cbc -in file.enc -out file.dec -pass file:./aes.key
```

7.2. Appendix B: Glossary

AEAD Authenticated Encryption with Associated Data.
Ciphertext-only attack Attacker has only ciphertext samples.
Nonce Number used once; ensures uniqueness.
Entropy Measure of randomness.

7.3. Appendix C: Further Reading

- Katz, Lindell — Introduction to Modern Cryptography.
- Ferguson, Schneier — Practical Cryptography.
- Menezes, van Oorschot, Vanstone — Handbook of Applied Cryptography.
- RFC 5280 (X.509), RFC 8446 (TLS 1.3).

References

[1] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* CRC Press.

[2] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography.* CRC Press.

[3] "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446.