Cryptography Basics - Comprehensive Exam Notes

Table of Contents

Overview and Classifications

Types of Cryptography

- **Classical Cryptography**: Historical methods (e.g., Little Dancing Men from Sherlock Holmes)
- **Modern Cryptography**:
  - **Symmetric-key**: ke = kd (encryption key = decryption key)
  - **Public-key**: ke ≠ kd (encryption key ≠ decryption key)

Security Assurance Goals

- **Confidentiality**: Keeping information secret
- **Integrity**: Ensuring data hasn't been tampered with
- **Authenticity**: Verifying the source of information

Cryptographic Tools by Purpose

- **Confidentiality**: Symmetric & Public-key encryption
- **Integrity & Authenticity**: Message Authentication Codes (MACs) & Digital Signatures

---

Tools for Confidentiality

Symmetric-Key Encryption

Basic Model

- **Encryption**: M → E(secret key) → C
- **Decryption**: C → D(secret key) → M
- Same secret key used for both encryption and decryption

Types of Symmetric Ciphers

Stream Ciphers

- **Operation**: Process plaintext one bit (or byte) at a time
- **Formula**: $C = c_1c_2... = Ek_1(p_1)Ek_2(p_2)...$
- **Simple Stream Cipher**:
  - Keystream generator: {ki}, i=1,2,...n
  - Plaintext bits: {pi}, i=1,2,...n
  - Ciphertext bits: {ci}, i=1,2,...n
  - Encryption: ci = pi ⊕ ki
  - Decryption: pi = ci ⊕ ki
- **Security Issues**:
  - Security depends entirely on keystream generator
  - If keystream is all zeros → no security
  - If keystream is truly random → one-time pad (perfect security)

Block Ciphers

- **Operation**: Process plaintext in fixed-size blocks (typically 64, 128, or 256 bits)
- **Formula**: $C = c_1c_2... = Ek(p_1)Ek(p_2)...$
- **Structure**: Multiple rounds of:
  - Key mixing (XOR with subkey)
  - Substitutions
  - Permutations
- **Avalanche Effect**: Small changes in plaintext or key cause significant changes in ciphertext
- **Examples**: DES, AES, RC4

Asymmetric-Key (Public-Key) Encryption

Basic Model

- **Encryption**: M → E(public key) → C
- **Decryption**: C → D(private key) → M
- Key pair: public key (published) and private key (kept secret)

Key Concepts

- User generates public/private key pair
- Public key is published, private key kept secret
- To send message to Bob: encrypt with Bob's public key
- Only Bob can decrypt (only he has the private key)

RSA Cryptosystem

Mathematical Foundation

- **Based on**: Difficulty of factoring large composite numbers
- **Easy**: Find primes and multiply them
- **Hard**: Factor a composite number back into primes

Key Generation

1. Choose two large primes p and q
2. Compute $n = pq$ and $m = \varphi(n) = (p-1)(q-1)$
   - $\varphi(n)$ = Euler's totient function
3. Choose e where $1 < e < m-1$ and $\gcd(e,m) = 1$
4. Find d such that $ed \equiv 1 \pmod{m}$
   - d is multiplicative inverse of e modulo m
   - Found using extended Euclidean algorithm
5. **Public key**: (e, n)
6. **Private key**: (d, n)

Encryption/Decryption

- **Encryption**: $Y = X^e \bmod n$
- **Decryption**: $X = Y^d \bmod n$
- X and Y are integers in $\{0, 1, ..., n-1\}$

Example

- $p = 11, q = 13$
- $n = 143, m = 120$
- $e = 37$ ($\gcd(37,120) = 1$)
- $d = 13$ ($37 \times 13 = 481 \equiv 1 \bmod 120$)

- **Encrypt X = 3**: $Y = 3^{37} \bmod 143 = 42$
- **Decrypt Y = 42**: $X = 42^{13} \bmod 143 = 3$

ElGamal Cryptosystem

Mathematical Foundation

- **Based on**: Discrete Logarithm Problem
- **Hard**: Given g, h, p, find a such that $h = g^a \bmod p$
- **Operates on**: Zp* = {1, 2, ..., p-1} where p is prime

Generator Concept

- Element α is a generator of Zp* if $\alpha^i \bmod p$ for $0 < i \leq p-1$ generates all numbers 1, ..., p-1

Key Generation

1. Alice chooses prime p and random numbers g, u < p
2. g must be a generator of Zp*
3. Calculate $y = g^u \bmod p$
4. **Public key**: (p, g, y)
5. **Private key**: u

Encryption/Decryption

- **Encryption** (Bob to Alice):
  - Choose random k < p-1
  - Calculate $a = g^k \bmod p$
  - Calculate $b = y^k \times X \bmod p$
  - Ciphertext: (a, b)
- **Decryption** (Alice):
  - $X = b/a^u \bmod p$ (division = multiplicative inverse)
- **Note**: Ciphertext is twice the length of plaintext

RSA vs ElGamal

- **RSA**: Deterministic (same plaintext → same ciphertext)
- **ElGamal**: Probabilistic (randomness k makes different ciphertexts)
- **Small domains**: RSA vulnerable to exhaustive search, ElGamal better

RSA in Practice: OAEP

- **Problem**: Textbook RSA is vulnerable
- **Solution**: Optimal Asymmetric Encryption Padding (OAEP)
- **Process**: Message → padding with random number → hash functions → RSA encryption

---

Tools for Integrity and Authenticity

Message Authentication Codes (MACs)

Purpose

- **Message Integrity**: Preventing unauthorized modification
- **Authenticity**: Verifying message source
- **Different from error detection**: Uses secret key (error detection doesn't)

MAC Process

1. Transmitter and receiver share secret key K
2. To send message M: calculate MAC and send (M, MACK(M))
3. Receiver: calculate MACK(M) and compare with received MAC
4. If match → message authentic; if not → message tampered

Common Constructions

- **Hash function based**: HMAC
- **Block cipher based**: Various modes

HMAC (Hash-based MAC)

Properties of Cryptographic Hash Functions

1. **Variable input size**: Can hash any size input
2. **Fixed output size**: Always produces same size output
3. **Easy to compute**: Efficient calculation
4. **Pre-image resistant**: Given Y, hard to find X where H(X) = Y
5. **Collision resistant**: Hard to find X ≠ Y where H(X) = H(Y)

HMAC Formula

$HMAC(K,M) = H(K \oplus opad \,||\, H((K \oplus ipad) \,||\, M))$

Where:

- K^+ = K padded with zeros on left
- ipad = [0x36 × blocksize]
- opad = [0x5c × blocksize]
- || = concatenation

Digital Signatures

Purpose

- **Public-key analogy** of MACs
- Provides **non-repudiation** (sender can't deny sending)
- **Verification**: Anyone with public key can verify
- **Signing**: Only holder of private key can sign

Basic Model

- **Signing**: M → S(private key) → signature t
- **Verification**: (M, t) → V(public key) → 0/1 (valid/invalid)

RSA Signature Scheme

Key Generation

- Same as RSA encryption
- Generate primes P, Q; compute N = PQ
- Find d, e such that $de \equiv 1 \mod (P-1)(Q-1)$
- **Public key**: (N, e)
- **Private key**: d

Signing and Verification

- **SIGN**: Given message m, compute $s = m^d \mod N$
- **VERIFY**: Given (m, s), check if $m = s^e \mod N$

Example

- P = 13, Q = 17, N = 221, e = 5, d = 77

- **Sign message 124**: $s = 124^{77} \bmod 221 = 37$
- **Verify (124, 37)**: $37^5 \bmod 221 = 124$ ✓

Hash-then-Sign

- **Problem**: How to sign long messages?
- **Solution**: Hash the message first, then sign the hash
- More efficient and secure

Hybrid Systems

Problem with Pure Approaches

Symmetric Key Systems

- **Advantages**: Fast encryption/decryption
- **Disadvantages**: Key establishment, distribution, and management problems

Public Key Systems

- **Advantages**: Solves key distribution problem
- **Disadvantages**: Slow, key authenticity issues

Hybrid Solution

- **Best of both worlds**: PKC (with PKI) + one-time symmetric key
- **Process**:
  1. Generate random symmetric key for session
  2. Encrypt data with fast symmetric algorithm
  3. Encrypt symmetric key with public-key algorithm
  4. Send both encrypted data and encrypted key
- **Benefits**: Fast encryption + secure key distribution

Certificates & Public Key Infrastructure (PKI)

- **Problem**: How to verify public key authenticity?
- **Solution**: Digital certificates issued by trusted Certificate Authorities (CAs)
- **PKI**: Complete system for managing public keys and certificates

---

Key Exam Tips

Important Formulas to Remember

- **RSA**: $Y = X^e \bmod n$, $X = Y^d \bmod n$
- **ElGamal**: $a = g^k \bmod p$, $b = y^k \times X \bmod p$
- **Stream cipher**: $c_i = p_i \oplus k_i$
- **HMAC**: $HMAC(K,M) = H(K \oplus opad \,||\, H((K \oplus ipad) \,||\, M))$

Common Exam Questions

1. **Compare symmetric vs asymmetric encryption**
2. **Explain RSA key generation and encryption/decryption**
3. **Describe the discrete logarithm problem**
4. **Explain MAC vs digital signature differences**
5. **Why use hybrid systems?**
6. **Security properties of hash functions**
7. **Avalanche effect in block ciphers**

Key Concepts to Understand

- **Modular arithmetic**: Essential for RSA and ElGamal
- **Prime numbers**: Critical for RSA security
- **Generators**: Important for ElGamal
- **Hash functions**: Foundation of HMAC and digital signatures
- **Key management**: Why hybrid systems are necessary