

# Table of contents

<b>Synthetic version</b>	<b>3</b>
<b>Cryptography and Information Security</b>	<b>3</b>
Fundamentals of Cryptography . . . . .	3
Kerckhoffs' Principle . . . . .	3
Classification of Encryption Systems . . . . .	3
Entropy . . . . .	3
Oracles and Security Models . . . . .	4
History of Cryptography . . . . .	4
Symmetric Cryptography . . . . .	4
Stream Ciphers . . . . .	4
Block Ciphers . . . . .	6
Cryptanalysis Techniques . . . . .	11
Asymmetric Cryptography . . . . .	12
Mathematical Foundations . . . . .	12
Base Problems and Complexity . . . . .	13
RSA . . . . .	14
ElGamal . . . . .	15
Rabin . . . . .	16
Elliptic Curves . . . . .	16
Hash Functions and Integrity . . . . .	18
One-Way Functions . . . . .	18
Hash Functions . . . . .	18
Message Authentication Codes . . . . .	18
Attacks on MDCs . . . . .	19
Computational Resistance . . . . .	19
MDCs Based on Encryption . . . . .	19
Customized MDCs . . . . .	19
MACs Based on Encryption . . . . .	20
Nested MACs and HMACs . . . . .	20
Applications . . . . .	20
UNIX Salt . . . . .	21
Digital Signatures . . . . .	21
Definitions and Classifications . . . . .	21
Signatures with Appendix . . . . .	21
Signatures with Recovery . . . . .	21
RSA Signature . . . . .	22
Blind Signatures . . . . .	22
Rabin Signature . . . . .	22
ElGamal Signature . . . . .	22

Signatures and Cryptocurrencies . . . . .	22
Summary Table . . . . .	23
Attack Types . . . . .	23
Authentication . . . . .	23
Message Authentication Methods . . . . .	23
Entity Authentication . . . . .	24
Dictionary Attacks . . . . .	24
Plaintext Equivalence . . . . .	24
Weak Authentication . . . . .	24
Strong Symmetric Authentication . . . . .	25
Strong Asymmetric Authentication . . . . .	25
Zero-Knowledge Proofs . . . . .	26
ZKIP Graph Isomorphism . . . . .	26
ZKIP Fiat-Shamir . . . . .	26
Practical Implementations . . . . .	27
Mafia Attack . . . . .	27
Attacks and Protections Table . . . . .	27
Key Establishment . . . . .	27
KEP (Key Establishment Protocol) . . . . .	27
Symmetric Protocols . . . . .	28
Asymmetric Protocols . . . . .	29
Attacks and Vulnerabilities . . . . .	30
Asymmetric KTP . . . . .	31
Protocols with KDC . . . . .	32
SSL/TLS . . . . .	32
KEP Best Practices . . . . .	33
Trusted Third Parties (TTP) . . . . .	33
TTP Modes . . . . .	33
KDC (Key Distribution Center) . . . . .	33
CA (Certification Authority) . . . . .	34
Certification Functional Entities . . . . .	34
Other TTPs . . . . .	35
Certificates . . . . .	35
Authentication Trees . . . . .	35
Certification Topologies . . . . .	36
PKI - Infrastructure . . . . .	36

## Synthetic version

# Cryptography and Information Security

## Fundamentals of Cryptography

### Kerckhoffs' Principle

**Security based on the key:** The security of the system relies solely on the secrecy of the key, never on the secrecy of the algorithm.

**Public algorithm:** The system must remain secure even if the algorithm is known to the adversary.

**No security through obscurity:** Kerckhoffs explicitly rejected this principle as early as the 19th century, stating that security must be mathematically demonstrable.

### Classification of Encryption Systems

**Unconditional:** Perfect and theoretical security, independent of computational power. Example: *one-time pad*.

**Provable security:** Cryptanalysis is equivalent to solving a reputedly difficult mathematical problem (like factorization for RSA).

**Computational:** Practical security based on the unrealistic cost of attacks with current resources. This is the most commonly used category.

### Entropy

**Entropy:** Measures the effective amount of information contained in a message, approximating the number of bits needed to encode it.

**Conditional entropy:** Quantifies the remaining uncertainty about the plaintext after observing the ciphertext.

**Redundancy:** Difference between the actual encoding of a message and its minimal entropy.

## Oracles and Security Models

**Random Oracle:** “Ideal” hash function used in theoretical proofs (ROM model), which returns uniform and random values.

**CPA/CCA Oracles:** Simulate access to encryption/decryption operations with the secret key to test the system’s resistance against an adversary.

**IND-CPA:** Indistinguishability property ensuring that an adversary cannot distinguish the encryptions of two different messages (equivalent to semantic security).

**Probabilistic Encryption:** Adds randomness to the message to prevent dictionary attacks, essential in asymmetric cryptography.

**OAEP:** Padding method that adds the necessary randomness to RSA encryption by combining the message with a random number via hash functions.

## History of Cryptography

**Historical:** Classical systems rely on substitution (Caesar, Vigenère) and transposition (permutation of characters).

**One-Time Pad:** Only system with absolute security if the key is random, unique, and as long as the message. Shannon’s condition:  $H(K) = H(X)$ .

**Steganography:** Technique that hides the very existence of the message rather than making it unreadable (example: LSB technique in digital images).

## Symmetric Cryptography

### Stream Ciphers

#### General Characteristics

**Stream Ciphers:** Bit-by-bit encryption in 2 phases (keystream generation + substitution). Block size = 1 bit.

#### Advantages:

- Fast (register-level encryption)
- Lightweight (limited CPU resources)
- No buffering
- No error propagation (retransmission sufficient for WiFi)

#### Disadvantages:

- Keystream quality critical for robustness
- Keystream reuse = major vulnerability

### Synchronous Stream Ciphers

**Synchronous:** Keystream depends only on the key, independent of plaintext/ciphertext. Equations:  $\sigma_{i+1} = f(\sigma_i, k)$ ,  $z_i = g(\sigma_i, k)$ ,  $c_i = h(z_i, m_i)$ .

**Requires synchronization:** Sender and receiver must share the same key AND the same state  $\sigma_i$ .

**No error propagation:** Modification of  $c_j$  affects only  $m_j$ , but deletion = desynchronization.

**Vulnerable to bit modifications:** Adversary can modify bits and analyze the impact. Requires additional authentication mechanisms.

**Common case:** Additive stream cipher with function  $h = \text{XOR}$  (like one-time pad).

### Asynchronous Stream Ciphers

**Asynchronous** (self-synchronized): Keystream depends on the key AND the last ciphertexts. State  $\sigma_i = \text{buffer of } t \text{ previous ciphertexts}$ :  $\sigma_i = (c_{i-t}, \dots, c_{i-1})$ .

**Self-synchronization:** Automatic re-synchronization after insertion/deletion of ciphertexts thanks to the buffer.

**Limited error propagation:** Error propagates only over  $\frac{n}{r}$  blocks ( $n = \text{nominal size}$ ,  $r = \text{plaintext size}$ ). After buffer exhaustion, correct decryption resumes.

**Better diffusion** of statistics: Each bit of the plaintext influences all subsequent ciphertexts. Ideal for redundant plaintexts or low entropy.

### LSFR Generators

**LSFR:** Long keystream generator ( $m$  bits) from a short key ( $l$  bits) with  $l \ll m$ . Base = linear combinations of bits.

**Advantages:**

- Hardware efficient
- Long periods
- Good random quality

**Problem:** Insufficient security compared to modern block ciphers. Vulnerable to the Berlekamp-Massey algorithm which calculates linear complexity and generates arbitrarily new sequences.

**Solution:** NLFSR (Non Linear Feedback Shift Registers) using a non-linear function  $f$  instead of a linear combination.

## RC4

**RC4** (Rivest Cipher 4, 1987): Software stream cipher with variable key, synchronous mode,  $10\times$  faster than DES.

**Architecture:**  $8\times 8$  S-box (0-255 permutation depending on the key) + XOR between keystream and plaintext. Linear and non-linear combinations.

**2 steps:**

- KSA (Key Scheduling Algorithm, S-box permutation)
- PRGA (Pseudo Random Generator Algorithm, keystream generation)

**Applications:** SSL/TLS, Windows, Oracle, Lotus Notes. Very widespread commercial use.

**Vulnerability:** WEP protocol (WiFi) completely broken due to a flaw in the usage mode, not the RC4 algorithm itself.

## Block Ciphers

### General Characteristics

**Block cipher:** Bijective function transforming  $n$ -bit blocks with a  $k$ -bit key. Each key defines a different bijection.

**Quality criteria:**

- Key entropy 128 bits (brute force protection)
- Block size 128 bits (avoid plaintext/ciphertext dictionaries)
- Cryptanalysis resistance = brute force effort

**Usage:** Confidentiality, authentication, hashing, random generation (cornerstone of cryptography).

### Operation Modes

#### ECB (Electronic Codebook)

**ECB:** Each block encrypted independently with the same key:  $c_i = E_K(m_i)$ .

**Major vulnerability:** Identical plaintexts  $\rightarrow$  identical ciphertexts. Patterns visible, plaintext structure transparent.

**Advantages:**

- Parallelizable
- No error propagation

**Never use** for redundant data.

### **CBC (Cipher Block Chaining)**

**CBC:** Each plaintext XORed with previous ciphertext before encryption:  $c_i = E_K(m_i \oplus c_{i-1})$  with  $c_0 = IV$ .

**Advantages:**

- Identical plaintexts  $\rightarrow$  different ciphertexts (if IV changes)
- Patterns erased by chaining
- Limited error propagation ( $c_j$  affects  $m_j$  and  $m_{j+1}$  only)

**IV (Initialization Vector):** Must be random or pseudo-random, transmissible in clear, different for each message with the same key.

**Parallelization:** Not parallelizable in encryption (sequential), parallelizable in decryption.

### **CFB (Cipher Feedback)**

**CFB (asynchronous):** Works like a stream cipher where keystream depends on previous ciphertexts:  $c_i = m_i \oplus E_K(c_{i-1})$  with  $c_0 = IV$ .

**Error propagation:** Error on  $c_j$  affects  $\frac{n}{r}$  following blocks ( $n$  = nominal size,  $r$  = plaintext size).

**Not parallelizable.** IV not confidential but must be transmitted.

**Usage:** Suitable for transmissions with frequent packet loss.

### **OFB (Output Feedback)**

**OFB (synchronous):** Stream cipher where keystream depends only on key + IV:  $z_i = E_K(z_{i-1})$ ,  $c_i = m_i \oplus z_i$  with  $z_0 = IV$ .

**Advantages:**

- No error propagation (error on  $c_j$  affects only  $m_j$ )
- Keystream pre-calculable (parallelizable if pre-calculated)

**CRITICAL:** Never reuse the same IV with the same key (otherwise identical keystream = major vulnerability). Change the IV for each new message.

## CTR (Counter Mode)

**CTR:** Keystream generated by encrypting an incremented counter:  $c_i = m_i \oplus E_K(\text{counter} + i)$ .

**Advantages:**

- Parallelizable (encryption + decryption)
- Random access to each block
- No error propagation
- SIMD-friendly (no dependencies between blocks)

**Counter:** Size  $2^b$  ( $b$  = block size), increment modulo  $2^b$  after each iteration.

**CRITICAL:** Never reuse the same counter with the same key. First block of stream  $i + 1 >$  last block of stream  $i$ .

**Usage:** ATM, IPsec, high bandwidth, video, selective block transmission.

## Product Ciphers and Feistel

**Product cipher:** Scheme combining a series of successive transformations (transpositions, substitutions, XOR, modular multiplications) to strengthen cryptanalysis resistance.

**Feistel cipher:** Iterative product cipher transforming plaintext  $2t$  bits =  $(L_0, R_0)$  into ciphertext  $(R_r, L_r)$  after  $r$  rounds (generally even and  $\geq 3$ ).

**Round equations  $i$ :**  $L_i = R_{i-1}$  and  $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$  where  $K_i$  = subkeys generated from main key  $K$ .

**Decryption:** Identical to encryption with subkeys applied in reverse order ( $K_r$  to  $K_1$ ).

**Example:** DES (16 rounds).

## DES (Data Encryption Standard)

### Structure

**DES:** Feistel cipher, 64-bit blocks, 56-bit effective key (64 total with 8 parity), 16 rounds, 16 48-bit subkeys.

**Structure:** Initial permutation IP  $\rightarrow$  16 Feistel rounds  $\rightarrow$  Final permutation IP<sup>-1</sup>.

## Round Function

**Function  $f$**  per round:

- Expansion E (32→48 bits)
- XOR with subkey  $K_i$  (48 bits)
- 8 S-boxes (48→32 bits, each S-box: 6 bits input → 4 bits output)
- Permutation P (32 bits)

**S-box:** 6 bits input  $a_1a_2a_3a_4a_5a_6 \rightarrow \text{row} = a_1 + 2a_6$  (outer bits),  $\text{column} = a_2 + 2a_3 + 4a_4 + 8a_5$  (inner bits) → 4 bits output.

## Subkey Generation

**Subkey generation:**

- PC-1 (56-bit selection)
- Division  $C_0, D_0$  (28 bits)
- Left circular rotations (1 or 2 positions)
- PC-2 (48-bit selection for  $K_i$ )

## Triple-DES and Security

**DES vulnerability:** Key space  $2^{56}$  breakable in 24h (1999, parallel brute force, 100,000 PCs).

**Triple-DES:**  $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$  with two 56-bit keys → space  $2^{112}$ .

**Advantages:**

- Satisfactory security
- Reuses existing DES hardware/software
- Compatibility
- Gradual migration to AES

**Disadvantage:** 3× slower (3 successive DES executions).

**DES group:** Composite encryption significantly strengthens security. If DES were a group, exhaustive search  $2^{56}$  would break the algorithm regardless of the number of executions.

**4 weak keys:** Generate identical subkey pairs ( $k_1 = k_{16}, k_2 = k_{15}, \dots, k_8 = k_9$ ) → facilitates cryptanalysis. 6 pairs of semi-weak keys.

## AES (Advanced Encryption Standard)

## General Structure

**AES** (Rijndael, 2001): Iterative block cipher (NOT Feistel), 128-bit blocks, 128/192/256-bit keys  $\rightarrow$  10/12/14 rounds respectively.

**State:**  $4 \times 4$  byte matrix (16 bytes = 128 bits). Operations in field  $GF(2^8)$  (degree 7 polynomials with  $GF(2)$  coefficients).

## Round Operations

**4 operations per round:**

- **SubBytes:** Non-linear substitution via S-box (resistance to linear/differential cryptanalysis)
- **ShiftRows:** Line shifts (line 0: 0, line 1: 1, line 2: 2, line 3: 3 positions left)
- **MixColumns:** Linear combinations of columns (matrix multiplication in  $GF(2^8)$ , maximal diffusion)
- **AddRoundKey:** XOR State with round subkey

**Final round:** Identical EXCEPT no MixColumns.

## Key Schedule and Performance

**Key Schedule:** Key expansion  $\rightarrow 4 \times 4 \times (N_e + 1)$  byte matrix ( $N_e$  = number of rounds)  $\rightarrow$  subkey selection (rotations, S-box substitutions, XOR with Rcon constants).

**Decryption:** Inverse operations (InvSubBytes, InvShiftRows, InvMixColumns) with subkeys in reverse order.

**Advantages:**

- 2 $\times$  faster than DES
- $10^{22}$  times more secure (theoretically)
- Open process
- Scalable
- Works on 8-bit cards

## AES Security

**Strengths:** Simplicity, performance (even on limited platforms like 8-bit smart cards), hardware/software optimizations.

**Published attacks:**

- **XSL (2002):** System of 8000 quadratic equations, 1600 binary unknowns, effort  $2^{100}$  (conjecture). Contested as based on AES's "strongly algebraic" nature

- **Related Key Attacks (2009-2011):** Interesting results on reduced versions, do not compromise full AES
- **Side Channel Attacks:** On implementation (cache timing, power analysis, electromagnetic). Example: 128-bit key extraction with 6-7 plaintext/ciphertext pairs via cache access analysis (2005)
- **Bicyclic Meet-in-Middle (2011-2015):** Reduces AES-128 effort to  $2^{126}$  (factor 4 vs brute force), remains well above current capabilities

**Practical security:** Key entropy maximization fundamental assumption. Recent attacks (WPA2) = weak passwords/passphrases, not AES flaw. Problem = key generation from weak passwords.

## Cryptanalysis Techniques

### Differential Cryptanalysis

**Differential:** Chosen plaintext attack analyzing propagation of differences ( $\Delta x = x_a \oplus x_b$ ) between plaintexts through rounds.

**Method:** Assign probabilities to keys based on observed changes in ciphertexts. Most probable key = correct key after many trials.

**DES effort:**  $2^{47}$  chosen plaintext pairs.

**Very sensitive to number of rounds:** Success chances increase exponentially when rounds decrease.

### Linear Cryptanalysis

**Linear:** Known plaintext attack creating a linear simulator of the block cipher via linear approximations. Simulator key bits tend to coincide with real key (probabilistic calculation).

**DES effort:**

- $2^{38}$  known plaintexts  $\rightarrow$  10% success probability
- $2^{43} \rightarrow$  85% success probability

**Most powerful analytical attack** to date on block ciphers. Very sensitive to number of rounds.

## Meet-in-the-Middle

**Meet-in-the-Middle:** Attack on composite constructions  $y = E_{K_2}(E_{K_1}(x))$ .

**Method:** Build 2 lists  $L_1 = \{E_{K_1}(x)\}$  and  $L_2 = \{D_{K_2}(y)\}$  for all  $K_1, K_2$ . Identify repeated elements. Verify with second known plaintext.

**Composite DES effort:**  $2^{57}$  operations +  $2^{56}$  block storage «  $2^{112}$  intuitively expected.

**Applications:** Composite constructions, internal cryptanalysis of block ciphers.

## Common Observations

**Common difficulties** (differential/linear):

- Less efficient parallelization than parallel brute force
- Rounds sensitivity

**DES and S-boxes:** Widespread conjecture = DES designers knew these attacks (unpublished in the 1970s). S-box design offers very high resistance to both techniques.

## Asymmetric Cryptography

### Mathematical Foundations

#### Number Theory

**Unique factorization:** Every integer = product of prime numbers (order irrelevant).

$\phi(n)$ : Euler's totient function counting integers  $< n$  coprime with  $n$ .

**Key for RSA:** If  $n = pq$  with  $p$  and  $q$  primes, then  $\phi(n) = (p-1)(q-1)$ .

**Euler's theorem:** If  $\gcd(a, n) = 1$ , then  $a^{\phi(n)} \equiv 1 \pmod{n}$ .

**Fermat's little theorem:** Special case if  $p$  prime:  $a^{p-1} \equiv 1 \pmod{p}$ .

**Modular inverse:**  $a^{-1} \equiv a^{\phi(n)-1} \pmod{n}$ . If  $p$  prime,  $a^{-1} \equiv a^{p-2} \pmod{p}$ .

**RSA basis:** These theorems enable encryption/decryption with exponents.

## Multiplicative Groups

$\mathbb{Z}_n^*$ : Set of elements coprime with  $n$ , cardinality  $= \phi(n)$ .

**Generator:** Element of order  $\phi(n)$  that generates the entire group by successive powers.

**Crucial for DH and ElGamal:** Security based on discrete logarithm in cyclic group.

**Safe prime:** Prime number  $n = 2p + 1$  with  $p$  also prime. Generator test simplified:  $\alpha$  is a generator iff  $\alpha^2 \not\equiv 1 \pmod{n}$  and  $\alpha^p \not\equiv 1 \pmod{n}$ .

## Fast Exponentiation

**Idea:** Use the binary representation of the exponent to efficiently compute  $a^k \bmod n$ .

**Complexity:**  $O(\log^3 n)$  - polynomial and very efficient.

**Essential:** Makes RSA, ElGamal, Diffie-Hellman practical in reasonable time.

**Alternative:** Extended Euclidean algorithm to compute modular inverses, same complexity  $O(\log^3 n)$ .

## Chinese Remainder Theorem

**Solves:** Simultaneous congruence systems with coprime moduli.

**Unique solution:** Modulo the product of moduli. Gauss's algorithm gives the explicit solution.

**Complexity:**  $O(\log^3 n)$  - polynomial.

**Cryptographic usage:**

- RSA calculation optimization (use  $p$  and  $q$  separately)
- Secret sharing
- Certain attacks if small exponent with multiple messages

## Base Problems and Complexity

### Fundamental Problems

**FACTP** (Factorization): Factor  $n$  into prime numbers  $\rightarrow$  basis of RSA/Rabin.

**DLP** (Discrete Logarithm): Find  $x$  such that  $\alpha^x \equiv \beta \pmod{p}$   $\rightarrow$  basis of ElGamal/Diffie-Hellman.

**SQROOTP** (Square Root mod Composite): Find  $\sqrt{a} \bmod n$  with  $n$  composite  $\rightarrow$  basis of Rabin.

**Proven equivalences:** Breaking the algorithm = solving the corresponding base problem.

### Factorization Techniques

**Sub-exponential:** NFS (Number Field Sieve) currently the fastest, complexity  $O(\exp(c \cdot (\ln(n))^{1/3}))$ .

**2020 record:** RSA-829 (829 bits, 250 digits) factored in 2700 core-years with GNFS.

**Recommendation:** RSA keys 2048 bits minimum (3072-4096 bits for long-term security).

**Future threat:** Quantum computers with Shor's algorithm (polynomial complexity  $O(\log^c n)$ ).

## RSA

### Principle

**Public key:**  $(n, e)$  with  $n = pq$  (product of two large prime numbers).

**Private key:**  $d$  such that  $ed \equiv 1 \pmod{\phi(n)}$ .

**Encryption:**  $c = m^e \bmod n$ .

**Decryption:**  $m = c^d \bmod n$ .

**Security:** Based on the difficulty of factoring  $n$ . Finding  $d$  factoring  $n$  (proven equivalence).

**Recommended size:**  $n \geq 2048$  bits minimum.

### Parameter Selection

**Small  $e$ :** Often 3, 17, or 65537 for fast encryption, but requires mandatory padding.

**Large  $d$ :** Must be at least  $\text{size}(n)/2$  for security. Decryption exponent always large.

**Separate keys:** Use distinct key pairs for encryption and signature.

## Attacks on RSA

**Same message, small  $e$ :** If identical message sent to multiple recipients with  $e = 3$ , the Chinese Remainder Theorem allows extracting  $m$  directly by cube root.

**Message too small:** If  $m < n^{1/e}$ , then  $c = m^e$  in  $\mathbb{Z}$  (not modulo)  $\rightarrow$  direct  $e$ -th root possible.

**Multiplicative property:**  $E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2) \bmod n$ . Enables chosen-ciphertext attacks and blind signatures.

**Essential protection:** Always use padding/randomization (standard OAEP) before encryption to avoid these attacks.

**General attack:** Most effective method remains factoring  $n$  if parameters are well chosen and implementation correct.

## ElGamal

### Principle

**Basis:** Discrete logarithm problem (DLP) in  $\mathbb{Z}_p^*$ .

**Public key:**  $(p, \alpha, y)$  with  $p$  prime,  $\alpha$  generator,  $y = \alpha^a \bmod p$ .

**Private key:**  $a$  (secret exponent).

**Encryption:** For message  $m$ , choose unique random  $k$ , compute  $\gamma = \alpha^k \bmod p$  and  $\delta = m \cdot y^k \bmod p$ . Send  $(\gamma, \delta)$ .

**Decryption:**  $m = \delta \cdot \gamma^{-a} \bmod p$ .

### Properties and Limitations

**Security:**  $k$  must be unique and large for each message.

**Major disadvantage:** Doubles the message size (1:2 expansion).

**Equivalence:** Based on DLP but strict equivalence not proven (unlike Rabin).

**$k$  unique CRITICAL:** If  $k$  reused for two messages,  $\delta_1/\delta_2 = m_1/m_2$  reveals the messages.

**Exponent size:**  $k$  and  $a$  must be large, otherwise vulnerable to baby-step giant-step algorithms.

**Extensions:** Can be generalized to  $GF(2^n)$  fields or elliptic curves.

## Rabin

### Principle

**Basis:** SQROOTP problem (square root modulo composite).

**Public key:**  $n = pq$  (product of two large prime numbers).

**Private key:**  $(p, q)$  the prime factors.

**Encryption:**  $c = m^2 \bmod n$ .

**Decryption:** Compute the 4 possible square roots of  $c \bmod n$  via efficient algorithms modulo  $p$  and  $q$ .

### Security

**Unique advantage:** Only algorithm with PROVEN equivalence to factorization (SQROOTP FACTP). “Provably secure” category.

**Problem:** 4 possible decryption solutions, requires redundancy mechanism or indication to identify the correct message.

**Chosen-ciphertext attack:** If adversary  $M$  sends  $c = m^2 \bmod n$  and receives root  $m_x \neq m$ , then  $\gcd(m - m_x, n)$  gives a factor of  $n$  (0.5 success probability).

**Countermeasure:** Require sufficient redundancy in messages to unambiguously identify the correct solution and reject others.

## Elliptic Curves

### Mathematical Structure

**Equation:**  $y^2 = x^3 + ax + b$  with discriminant  $4a^3 + 27b^2 \neq 0$ .

**Structure:** Forms a commutative group with point at infinity  $\mathcal{O}$  as identity element.

**Fundamental operation:** Geometric point addition (3rd intersection point + symmetry).

**Inverse:**  $-P = (x, -y)$  (symmetry about the x-axis).

**Addition:** Draw line between  $P$  and  $Q$ , find 3rd intersection point, take its symmetric.

**Doubling:** Use tangent at point  $P$  instead of line between two points.

## Security and Advantages

**Hard problem:** ECDLP (Elliptic Curve Discrete Logarithm Problem) - finding  $k$  in  $Q = kP$  requires exponential effort.

**Major gain:** Keys about 6-10× shorter than RSA/DH for equivalent security.

**Limitation:** Representing messages as curve points remains a complex operation.

**Adoption:** NSA purchased Certicom patent in 2003 for cryptographic use.

## Key Size Comparison

For security equivalent to AES 128 bits:

- RSA requires 3072 bits
- Elliptic curves only 256 bits (ratio 1:12)

For security equivalent to AES 256 bits:

- RSA requires 15360 bits
- Elliptic curves only 512 bits (ratio 1:30)

## ElGamal on Elliptic Curves

**Principle:** Direct adaptation of ElGamal replacing operations in  $\mathbb{Z}_p^*$  with operations on curve  $E_p$ .

**Public key:**  $(E_p, P_0, P_a)$  with  $P_0$  point of large order,  $P_a = xP_0$ .

**Private key:**  $x$  (secret scalar).

**Encryption:** For message  $m_i \in E_p$ , choose random  $k$ , compute  $\gamma = kP_0$  and  $\delta = kP_a + m_i$ . Send  $(\gamma, \delta)$ .

**Decryption:**  $m_i = \delta - x\gamma$ .

**Operations:** Point addition and scalar multiplication on elliptic curve.

**Security:** Relies on ECDLP (difficulty of computing discrete logarithm on curve).

**Authentication:** Necessary to prevent man-in-the-middle attacks, as for classical ElGamal.

**Advantage:** Much shorter keys for equivalent security (factor 6-30).

## Hash Functions and Integrity

### One-Way Functions

**OWF**: Easy in one direction ( $f(x) \rightarrow y$ ), impossible in the other ( $y \rightarrow x$ ).

Examples:

- Modular squares
- $E_k(x) \oplus x$

OWF   OWHF (hash functions = more constraints).

### Hash Functions

#### Types and Properties

**Hash function**: Compression + easy computation

**MDC** (without key) for integrity

**MAC** (with key) for authentication

**Properties**:

1. preimage resistance
2. 2nd-preimage resistance
3. collision resistance

**OWHF** = (1)+(2)

**CRHF** = (2)+(3)

### Message Authentication Codes

**MAC** = hash with key  $k$

Without  $k$ : impossible to forge  $(x, h_k(x))$  or recover  $k$

Guarantees origin authentication + integrity.

## Attacks on MDCs

To break 2nd-preimage resistance with  $m$ -bit digest:  $2^{m-1}$  trials (prob 0.5).

**Birthday paradox:** To break collision resistance with  $m$ -bit digest:  $2^{m/2}$  trials (prob  $> 0.5$ ).

Example: 23 people sufficient for matching birthdays.

## Computational Resistance

**Efforts:**

- preimage  $2^n$
- 2nd-preimage  $2^{n-1}$
- collision  $2^{n/2}$

**Sizes:**

- OWHF 128 bits
- CRHF 256 bits
- MAC key 256 bits

## MDCs Based on Encryption

MDCs from symmetric crypto: break reversibility + XOR chaining.

Models:

- Matyas-Meyer-Oseas
- Davies-Meyer
- Miyaguchi-Preneel

MDC-2/4 with DES  $\rightarrow$  128 bits.

## Customized MDCs

**Customized MDCs:**

- MD5 (broken)
- SHA-0 (broken)
- SHA-1 (weak)
- SHA-2 (secure)
- SHA-3/Keccak (current standard)

Construction: padding + constants + rounds + chaining.

## MACs Based on Encryption

**CBC-MAC:** CBC mode + IV=0, only last block kept. DES: 56/112-bit key, 64-bit MAC.

## Nested MACs and HMACs

**HMAC:** Double hash with derived keys (ipad/opad).

$$\text{HMAC}_k(x) = H((k \oplus \text{opad}) \parallel H((k \oplus \text{ipad}) \parallel x))$$

Standard, secure, efficient.

## Applications

### Integrity

**Integrity:** MAC alone, MDC+crypto, MDC+signature.

Vulnerable to replay without timestamps/nonces.

## Blockchain

**Blockchain:** Block chaining via hash.

**Proof of Work:** Find nonce for hash < target.

Security = effort > all miners.

Bitcoin: ~10 min/block,  $10^{21}$  hash/sec.

## Other Applications

**Applications:**

- authentication
- virus checking
- public key distribution
- timestamp
- one-time passwords (hash chain)

## UNIX Salt

**UNIX salt:** 12 random bits added to password before hash.

4096 possible variations.

Prevents pre-computed codebooks and duplicate detection.

## Digital Signatures

### Definitions and Classifications

**Digital signature** = string associating message + entity

**Two types:**

- with appendix (requires original message)
- with recovery (reconstructs the message)

Based on asymmetric cryptography

### Signatures with Appendix

Signature:  $S_A(m_h) = s$  (private key).

Verification:  $V_A(m_h, s)$  (public key).

Impossible to forge without private key.

### Signatures with Recovery

**With recovery:** Redundancy function  $R(m) = m_R$ .

Signature  $s = S_A(m_R)$ .

Verification:  $m_R = V_A(s)$ , reconstruct  $m = R^{-1}(m_R)$ .

Redundancy crucial for security.

## RSA Signature

**RSA signature:**  $s = m_R^d \bmod n$  (private).

Verification:  $m'_R = s^e \bmod n$  (public).

With appendix:  $s = H(m)^d \bmod n$ .

Signature slow, verification fast.

## Blind Signatures

**Blind signature:** Exploits RSA multiplicativity.

Camouflage  $f(m) = m \cdot k^e$

Decamouflage  $g(m) = k^{-1} \cdot m$ .

B signs  $f(m)$ , A obtains  $S_B(m)$  without B seeing  $m$ .

## Rabin Signature

**Rabin:**  $s = \sqrt{m_R} \bmod n$ .

Verification:  $m'_R = s^2 \bmod n$ .

**Provably secure** (equivalent to factorization).

Vulnerable to active chosen-ciphertext attacks.

## ElGamal Signature

**ElGamal:**  $(r, s)$  with  $r = \alpha^k \bmod p$ ,  $s = k^{-1}(m_h - ar) \bmod (p-1)$ .

Verification:  $y^r r^s \stackrel{?}{=} \alpha^{H(m)} \bmod p$ .

Basis of DSA.

$k$  unique crucial.

## Signatures and Cryptocurrencies

**Cryptocurrencies:** Bitcoin/Ethereum use **ECDSA** (ElGamal on elliptic curves).

Each transaction signed with holder's private key.

Security based on ECDLP.

## Summary Table

### Classical signatures:

- RSA/Rabin (recovery)
- ElGamal/DSS (appendix)

### Specialized:

- One-time
- Undeniable
- Fail-Stop
- Blind

Base problems: RSAP, SQROOTP, DLP, depends on the OWF.

## Attack Types

### Breaking signature:

- Total break (private key)
- selective forgery (fixed message)
- existential forgery (any message)

### Attacks:

- key-only
- known/chosen/adaptive-chosen-messages

## Authentication

### Message Authentication Methods

#### 4 methods:

- MAC alone
- MDC+encryption
- MDC+confidential encryption
- MDC+signature

**Warning:** Vulnerable to replay attacks without temporal mechanisms

## Entity Authentication

### Authentication Levels

3 levels:

- Weak (reveals secret)
- Strong (proof of possession)
- Zero-knowledge (no info revealed)

4 objectives:

- Acceptance if honest
- non-reusability
- impersonation resistance
- observation resistance

### Dictionary Attacks

**Offline** (via DB or capture) > **Online** (limited by system)

**Protection:**

- salting
- attempt limiting
- strong authentication

### Plaintext Equivalence

**Plaintext-equivalent:** Data usable like the original password

**Danger:** If the system transmits  $H(p)$  and stores  $H(p) \rightarrow H(p)$  is plaintext-equivalent

**Good design:** System transmits  $p$ , stores  $H(p) \rightarrow$  not plaintext-equivalent

### Weak Authentication

#### Types and Storage

2 types:

- Fixed password (static)
- Variable password (changes per instance)

**Storage:**

- Cleartext (highly vulnerable)
- Encrypted/Hashed (offline attacks)

**Protections:**

- Strict rules
- attempt limiting
- salting
- non-dissemination

**Specific Methods**

**Lamport:**  $w_{n+1} = H(w_n)$ , authentication via hash chain verification

**Hardware:** Synchronized generator (30-60s), limited to pre-play

**Warning:** Requires B's authentication to prevent pre-play and small-n attacks

**Strong Symmetric Authentication**

**Challenge-Response:** B sends challenge R, A responds with  $E_k(R)$

**Alternative:** MAC instead of encryption (faster)

**With timestamp:** One fewer message but requires synchronized clocks

**Reflection attack:** Use one session's response to authenticate another

**Protection:** Include identities + asymmetry in challenges  $(R1, R2)$  vs  $(R2, R1)$

**Strong Asymmetric Authentication**

**Vulnerability:** Chosen-ciphertext attacks if no structure

**Protection:** Include  $H(R)$ , B's identity in the encrypted message, A verifies before revealing R

**Needham-Schroeder:** 3 messages with identity inclusion to prevent chosen-ciphertext

## Zero-Knowledge Proofs

### Fundamental Properties

3 properties:

- Completeness (accepts if honest)
- Soundness (requires secret)
- Zero-knowledge (no info revealed)

**Structure:** Witness  $\rightarrow$  Challenge  $\rightarrow$  Response (repeat  $n$  times)

**Perfect ZK:** Indistinguishable from simulation even with infinite resources

### Cave Example

**Cave:** A enters randomly (y or z), B asks for exit (left/right)

**Cheating probability:**  $2^{-n}$  after  $n$  repetitions

**ZK:** B verifies knowledge but learns no secret, cannot convince third party

### ZKIP Graph Isomorphism

**Problem:** Finding permutation between 2 isomorphic graphs = difficult

**Protocol:** A creates random H, B asks for permutation to G1 or G2, A responds

**Perfect ZK:** Transcripts indistinguishable from simulator

### ZKIP Fiat-Shamir

**Secret:**  $s$  such that  $v = s^2 \bmod n$  (public key)

**Protocol:**

- Witness  $r^2$
- challenge  $e \in \{0, 1\}$
- response  $y = r \cdot s^e$

**Verification:**  $y^2 \equiv x \cdot v^e \pmod{n}$

**Perfect ZK:** Pairs (x,y) simulatable by B

## Practical Implementations

**FSS:** Multiple witnesses/challenges  $\rightarrow$  probability  $2^{-nk}$

**GQ:** Expanded challenge domain  $\rightarrow$  fewer exchanges

**Schnorr:** DLP + large challenges  $\rightarrow$  **3 exchanges only**

**All:** More efficient than RSA, suitable for smart cards

## Mafia Attack

**Mafia attack:** Relay messages via accomplice  $\rightarrow$  transparent fraudulent authentication

### Protections:

- Faraday cage
- strong synchronization
- distance bounding

## Attacks and Protections Table

Attack	Protection
replay	zero-knowledge, challenge-response, one-time password
known/chosen-plaintext	zero-knowledge
chosen-ciphertext	zero-knowledge, witness + structure
reflection	include identities, message asymmetry
interleaving	include identities, cryptographic chaining
collusion	Faraday cage, strong synchronization

## Key Establishment

### KEP (Key Establishment Protocol)

#### Definitions

Mechanism to share a secret for cryptographic exchanges.

#### Protocol types:

- **KTP:** One entity creates and transmits the key
- **KAP:** Entities jointly derive the key
- **Pre-distribution:** Keys determined a priori

- **DKE (Dynamic Key Establishment):** Keys changing per execution

## Authentication Properties

Authentication properties:

- **Implicit key authentication:** Assurance only the correspondent can access the key
- **Key confirmation:** Assurance the correspondent possesses the key
- **Explicit key authentication:** Implicit + confirmation

## Security Properties

Security properties:

- **PFS (Perfect Forward Secrecy):** Past keys protected even if long-term keys compromised
- **Future Secrecy:** Future keys protected even if compromise by passive attacker
- **Deniability:** Participation not provable to third party

## Symmetric Protocols

### Trivial Symmetric KAP (pre-distribution)

$n(n-1)/2$  keys for  $n$  users

Information-theoretically secure

Problem:  $O(n^2)$  storage,  $O(n)$  keys per user

### Simple Symmetric KAP DKE

$$K := E_S(r_a \oplus r_b)$$

Properties:

- Implicit key authentication
- Entity authentication, key confirmation, PFS

## AKEP2

Uses MACs for authentication, derived key  $K := h'_{S'}(r_b)$

Properties:

- Mutual entity authentication
- Implicit key authentication
- Key confirmation, PFS

## Asymmetric Protocols

### Diffie-Hellman

$K := \alpha^{xy} \bmod p$  computed independently by A and B

Secure against passive attacks (DHP  $\neq$  DLP)

Vulnerable to Man-in-the-Middle without authentication

Generate symmetric keys:  $K_{sym} := \text{SHA}(K)$

Properties:

- Entity authentication, implicit key authentication, key confirmation

### Station to Station (STS)

Authenticated DH with digital signatures

Properties:

- Mutual entity authentication
- Implicit + explicit key authentication
- PFS: past session keys protected even if signature key compromised

Used in IPv6

## OTR/Signal

Protocols for instant messaging

SIGMA: signature + MAC

KDF generates  $K_e$  (encryption) and  $K_m$  (MAC)

Reveal old MAC keys  $\rightarrow$  repudiability

Properties:

- PFS, future secrecy, repudiability

Used: WhatsApp, Facebook Messenger

## SRP (Secure Remote Password)

Asymmetric KAP based on password

B stores verifier  $v := \alpha^x$  (not the password)

Resistant to dictionary attacks

Properties:

- All KEP properties

## Attacks and Vulnerabilities

### Logjam (2015)

Attack on DH:

- Downgrade to 512-bit groups via MIM
- Pre-computation (1 week) + individual computation (1 minute)
- Reuse if same prime  $p$

State actors can break PFS on 1024 bits

## **Asymmetric KTP**

### **Trivial Symmetric KTP**

$K := r_a$  with  $E_S(r_a)$

Properties:

- Implicit key authentication
- Entity authentication, key confirmation, PFS

### **Shamir's No-key Protocol**

Transport via successive exponentiations  $K^a, (K^a)^b, K^b$

Vulnerable to Man-in-the-Middle

### **Needham-Schroeder Asymmetric**

$K := H(k_1, k_2)$  with exchanges encrypted by public keys

Properties:

- Entity authentication, implicit key authentication, key confirmation
- PFS

### **EKE (Encrypted Key Exchange)**

Hybrid symmetric + asymmetric protocol

Password + asymmetric crypto

Robust even with weak password

Properties:

- PFS if keys regenerated each time

## Protocols with KDC

### Symmetric Needham-Schroeder (with KDC)

KDC generates and distributes  $k_{AB}$

Vulnerable to replay and known-key attacks

Solution: add timestamp

Basis for Kerberos

### Kerberos

Authentication and key distribution with KDC

AS issues TGT (Ticket Granting Ticket)

TGS issues service tickets

Tickets contain encrypted session keys

Authentication via authenticators

Properties:

- Entity authentication, implicit key authentication
- Partial key confirmation
- PFS

Vulnerabilities: password guessing, replay attacks

Solutions: pre-authentication, timestamps

### SSL/TLS

Meta-protocol between TCP and application layer

Services:

- confidentiality
- integrity
- authentication
- server identification

Handshake: parameter negotiation + certificate authentication

Keys derived by cascade:  $pre\_master\_secret \rightarrow master\_secret \rightarrow key\_block$

Properties:

- Entity authentication, implicit key authentication, key confirmation
- PFS depends on exchange protocol (DH yes, RSA no)

HTTPS standard

Notable flaws: random generation, heartbleed, renegotiation

## **KEP Best Practices**

**KEP best practices:**

- Define objectives (confidentiality, authentication, non-repudiation)
- Define desired security level (key confirmation, PFS)
- Establish environmental constraints
- Choose proven and robust solution
- Verify properties: practical + formal analysis
- Avoid pitfalls: reflection attacks, random number control, encrypted quantity redundancy

## **Trusted Third Parties (TTP)**

### **TTP Modes**

**TTP modes:**

- **In-line:** TTP intermediary relays all exchanges
- **On-line:** TTP participates in real-time, A and B communicate directly
- **Off-line:** TTP does not participate in real-time, info available a priori (ex: CA)

Off-line: no permanent availability required but revocation more complex

## **KDC (Key Distribution Center)**

Solves distribution problem:  $n^2$  keys  $\rightarrow$  only  $n$  keys

Scalable: +1 entity = +1 key

Session keys generated dynamically

**Risks:**

- single point of failure (security + operational)
- performance bottleneck

Solutions: mirroring, load balancing

## CA (Certification Authority)

### Role and Operation

Authenticates entity   public key association

Creates and signs certificates

Off-line mode

CRLs (Certificate Revocation Lists) for invalid certificates

CA compromise = serious consequences

### PoP (Proof of Possession)

Verify private key possession (not just identity)

Without PoP: attacker can impersonate identity via fraudulent certificate

Protocol: CA sends challenge  $A, r$ , verifies  $S_{priv_A}(A, r)$

Introduces trust levels for CAs

### Separation of Duties

Certificates   CRLs signed with different keys

CA (certification)   Revocation Authority (revocation)

Separate machines and security policies

Avoids post-compromise attacks on CA key

### Certification Functional Entities

Certification functional entities:

- **Name Server:** unique name management + DNSSec
- **RA (Registration Authority):** direct contact, identity/PoP verification
- **Key Generator:** key pair generation ( loses non-repudiation as private key known)
- **Certificate Directory:** read-only certificate access

## Other TTPs

### Other TTPs:

- **TA (Timestamp Agent)**: certifies document existence at specific time
- **Notary Agent**: TA + validity/origin (non-repudiation support)
- **KEA (Key Escrow Agent)**: session key access under legal conditions

Historical example: Clipper/Capstone/Fortezza (controversial)

## Certificates

### Structure

#### Certificate - Structure:

- **Issuer**: CA signer identity
- **Subject**: certified entity name
- **Subject Public Key**: public key (ex: RSA  $(n, e)$ , DH  $(p, \alpha, \alpha^x)$ )
- **Subject Public Key Algorithm**: RSA, DH, etc.
- **Validity**: validity period (UTC)
- **Signature**: CA signs everything, guarantees authenticity

## CRLs (Certificate Revocation Lists)

Lists of certificates that have become invalid (key compromised, algorithm change, etc.)

Structure: issuer, issue dates, revoked serial numbers, signature

Frequent publication required (wide audience)

**Achilles' heel** of PKI systems

Alternative: very short-lived certificates (a few minutes) → return to on-line mode

## Authentication Trees

Alternative certification via hash functions + tree structure

Only root  $R$  requires signature

Verification: provide path from leaf ( $\sim \log_2 n$  values)

Construction: leaves  $Y_i$ , edges  $h(Y_i)$ , nodes  $h(h_1 || h_2)$

**Main application**: timestamping

TA publishes  $R$  daily (newspaper) to prevent cheating

## Certification Topologies

Certification topologies:

- **Cross-certification:**  $CA_A$  certifies  $pub_{CA_B}$
- **Chain:**  $CA_A\{CA_B\} \rightarrow CA_B\{B\}$
- **Hierarchical model** (PEM/X.509): universal root, public key assumed globally known
- **Graph model** (PGP): users act as CAs, decentralized
- **Hybrid:** hierarchy + bidirectional cross-certification

**Golden rule:** short chains (weakest link)

## PKI - Infrastructure

### Main Entities

PKI - Main entities:

- **CA:** certificate creation and maintenance
- **Certificate Repository:** accessible storage (X.500, LDAP, WWW, DNS)
- **Revocation:** CRL management
- **Key Backup/Recovery:** lost key backup (decryption keys only, not signature)
- **Automatic Key Update:** key renewal
- **Key/Certificate History:** obsolete key retrieval for old documents
- **Cross-Certification:** validation of other PKIs' certificates
- **Non-Repudiation Support:** data origin authentication, time-stamped signatures, signed receipts
- **Secure Time Stamping:** time reference accepted by all
- **Client Software:** user operations (certificate management, signatures, peripherals)

## Advantages

PKI - Advantages:

- **Security:** integrated environment without weak links
- **All-in-one:** multi-services (strong authentication, signatures, single sign-on, VPNs, B2C/B2B)
- **Interoperability:** widespread standards (X.509, PKCS, OCSP), inter-enterprise compatibility

## **Disadvantages**

### **PKI - Disadvantages:**

- **Cost:** expensive products, rare skills
- **Complexity:** implementation and management

**Solution:** PKI service outsourcing