Republic of Tunisia
Ministry of Higher Education and Scientific Research
University of Tunis
Tunis Business School

Security Report
CypherSafe

Security Course Project for the Bachelor's in Information Technology and Business Administration

BY:

Mohamed Achek
Mohamed Amine Saoudi

ACADEMIC ADVISOR
Prof. Manel Abdelkader

Academic Year
2024-2025

Contents

Introduction to the Concept **1**

1 Overview of Algorithms **2**
1.1 Symmetric Encryption Algorithms 2
1.1.1 AES (Advanced Encryption Standard) 2
1.1.2 DES (Data Encryption Standard) 2
1.1.3 Blowfish 3
1.2 Asymmetric Encryption Algorithms 3
1.2.1 RSA (Rivest–Shamir–Adleman) 3
1.2.2 Elliptic Curve Cryptography (ECC) 3
1.2.3 Diffie-Hellman Key Exchange 3
1.3 Hashing Algorithms 4
1.3.1 SHA-256 (Secure Hash Algorithm 256-bit) 4
1.3.2 MD5 (Message Digest Algorithm 5) 4
1.3.3 SHA-3 (Keccak) 4

2 Mathematical Foundations **5**
2.1 Symmetric Encryption 5
2.2 Asymmetric Encryption 6
2.3 Hashing 7

3 Comparative Analysis **9**

4 Recent Advances and Future Directions **10**

Conclusion **11**

Bibliography **12**

Abstract

This report provides a detailed examination of the essential concepts of encryption, decryption, and hashing in the realm of information security. We delve into the algorithms and mathematical principles that underpin these processes, comparing different techniques, and discussing their strengths, weaknesses, and applications. Additionally, we explore recent advancements and emerging trends that shape the future of cryptographic technologies.

Introduction to the Concept

Encryption, decryption, and hashing are fundamental techniques in the field of information security, each serving distinct but complementary purposes. **Encryption** is the process of converting readable data (plaintext) into an unreadable format (ciphertext) to prevent unauthorized access. It ensures the **confidentiality** of information during storage or transmission. **Decryption** is the reverse operation: it transforms ciphertext back into its original readable form using a decryption key. It allows authorized users to access secured data. **Hashing**, on the other hand, is a process that transforms input data of arbitrary size into a fixed-size hash value (digest). Hashing is primarily used to verify data **integrity** — ensuring that information has not been tampered with.

Historically, the need for protecting information dates back thousands of years. One of the earliest known encryption methods is the Caesar cipher, used by Julius Caesar to send military messages. Over time, more sophisticated encryption methods emerged, particularly during the world wars, with machines like the Enigma cipher used by the Germans in WWII. With the advent of computers in the 20th century, digital encryption algorithms like DES (Data Encryption Standard) were introduced, setting the foundation for modern cybersecurity practices. Hashing evolved similarly, with early checksum methods used in data transmission progressing into robust cryptographic hashes like SHA-1 and SHA-2. Today, encryption and hashing form the backbone of secure communication protocols such as HTTPS, VPNs, and digital signatures.

Chapter 1

Overview of Algorithms

1.1 Symmetric Encryption Algorithms

1.1.1 AES (Advanced Encryption Standard)

Description: AES is a block cipher that encrypts data in fixed blocks of 128 bits using keys of 128, 192, or 256 bits. It operates through multiple rounds of substitution, permutation, mixing, and key addition processes.
Typical Use Cases: Securing sensitive data, cloud storage, VPNs, and disk encryption.
Strengths: Extremely fast and efficient for both hardware and software. Highly secure and resistant to known attacks.
Weaknesses: Requires secure key management.
Symmetric keys must be exchanged securely.

1.1.2 DES (Data Encryption Standard)

Description: DES encrypts 64-bit blocks of data using a 56-bit key through 16 rounds of permutation and substitution.
Typical Use Cases: Legacy banking systems, secure email communications (historically).
Strengths: Simple design and implementation. Historical significance as a pioneer in digital security.
Weaknesses: Short key length makes it vulnerable to brute-force attacks.
Superseded by AES and Triple DES.

1.1.3 Blowfish

Description: Blowfish is a symmetric block cipher that encrypts data in 64-bit blocks with a variable-length key (32–448 bits). It is known for its simplicity and speed.
Typical Use Cases: Password hashing, secure file storage, and VPNs.
Strengths: Extremely fast encryption. Free and unpatented.
Weaknesses: 64-bit block size is considered small for modern standards.
Being gradually replaced by newer algorithms like AES.

1.2 Asymmetric Encryption Algorithms

1.2.1 RSA (Rivest–Shamir–Adleman)

Description: RSA relies on the difficulty of factoring large prime numbers and uses a pair of keys (public and private) for encryption and decryption.
Typical Use Cases: Secure data transmission, digital signatures, and SSL/TLS.
Strengths: Strong security with appropriate key lengths. No need to exchange secret keys.
Weaknesses: Computationally intensive and slow compared to symmetric algorithms.

1.2.2 Elliptic Curve Cryptography (ECC)

Description: ECC is based on the algebraic structure of elliptic curves over finite fields. It offers the same security as RSA but with smaller keys.
Typical Use Cases: Mobile device encryption, secure messaging, blockchain security.
Strengths: Smaller keys with equivalent security levels. Faster computations, ideal for constrained environments.
Weaknesses: More complex mathematical structure. Patents and licensing concerns (historically).

1.2.3 Diffie-Hellman Key Exchange

Description: Not an encryption algorithm itself, but a method for securely exchanging cryptographic keys over a public channel.
Typical Use Cases: Establishing a shared secret for symmetric encryption in VPNs and HTTPS.
Strengths: No prior shared secret needed. Foundation for secure communications.

Weaknesses: Vulnerable to man-in-the-middle attacks without authentication. Does not itself encrypt data.

1.3 Hashing Algorithms

1.3.1 SHA-256 (Secure Hash Algorithm 256-bit)

Description: SHA-256 is a member of the SHA-2 family that produces a 256-bit hash from an input of any length.
Typical Use Cases: Data integrity verification, blockchain, password storage.
Strengths: High resistance to collision and preimage attacks. Widely adopted and standardized.
Weaknesses: Computationally heavier compared to simpler hashes like MD5.

1.3.2 MD5 (Message Digest Algorithm 5)

Description: MD5 produces a 128-bit hash value and was once widely used for data integrity checks.
Typical Use Cases: Legacy checksum verification, old password storage.
Strengths: Fast and simple implementation.
Weaknesses: Cryptographically broken; vulnerable to collision attacks. Should not be used for secure applications.

1.3.3 SHA-3 (Keccak)

Description: SHA-3 is based on a sponge construction and is the latest member of the Secure Hash Algorithm family.
Typical Use Cases: Secure data hashing in future-proof systems.
Strengths: Resilient to many cryptographic attacks. Independent from SHA-2's internal structure.
Weaknesses: Slightly slower in software compared to SHA-2.

Chapter 2

Mathematical Foundations

The underlying mathematical principles of encryption and hashing algorithms rely heavily on number theory, algebra, and modular arithmetic.

2.1 Symmetric Encryption

AES (Advanced Encryption Standard)

AES uses finite field arithmetic, particularly operations over $GF(2^8)$, the Galois Field with 2^8 elements. The encryption process includes substitution using an S-Box (based on multiplicative inverses in $GF(2^8)$), permutation via shifting rows and mixing columns, and addition of round keys.

Key equation for AES round transformation:

$$\text{State} = \text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(\text{State}))) \oplus \text{RoundKey}$$

DES (Data Encryption Standard)

DES operates on 64-bit blocks using a 56-bit key, applying 16 rounds of a Feistel structure. Each round uses a round-specific subkey generated from the main key.

Core operations include:

- Expansion of half-blocks,
- Key mixing via XOR,
- Substitution using S-boxes (nonlinear transformations),
- Permutation for diffusion.

Mathematical structure of a round function:

$$F(R, K) = P(S(E(R) \oplus K))$$

where E is the expansion function, S represents substitution, and P denotes permutation.

Blowfish

Blowfish uses a Feistel network with 16 rounds and key-dependent S-boxes. It relies on modular addition, XOR operations, and substitution via four S-boxes.

The round function is defined as:

$$F(x) = ((S_1(a) + S_2(b)) \oplus S_3(c)) + S_4(d)$$

where x is divided into four 8-bit parts a, b, c, d.

2.2 Asymmetric Encryption

RSA (Rivest–Shamir–Adleman)

RSA's security is based on the difficulty of factoring the product of two large primes.

Key generation steps:

- Choose two primes, p and q,
- Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$,
- Choose public exponent e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$,
- Find d such that $ed \equiv 1 \pmod{\varphi(n)}$.

Encryption:

$$c = m^e \mod n$$

Decryption:

$$m = c^d \mod n$$

Elliptic Curve Cryptography (ECC)

ECC is based on the algebraic structure of elliptic curves over finite fields. The fundamental operation is point multiplication.

Elliptic Curve Equation (over a finite field F_p):

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

Key operations:

- Point addition and point doubling,
- Scalar multiplication: $Q = kP$ where P is a base point and k is a scalar.

Security is based on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

Diffie-Hellman Key Exchange

Diffie-Hellman is based on modular exponentiation and the hardness of the Discrete Logarithm Problem.

Public parameters:

- Prime number p,
- Generator g of the multiplicative group Z_p^*.

Key exchange process:

- Alice picks private a and computes $A = g^a \mod p$,
- Bob picks private b and computes $B = g^b \mod p$,
- Shared secret:

$$K = B^a \mod p = A^b \mod p$$

2.3 Hashing

SHA-256 (Secure Hash Algorithm 256-bit)

SHA-256 processes data in 512-bit blocks and produces a 256-bit hash value.

Core compression function:

$$H_i = H_{i-1} + \text{Compress}(H_{i-1}, M_i)$$

where H_i is the new hash value and M_i is the message block.

The `Compress` function involves modular additions, bitwise operations, and logical shifts to ensure avalanche properties.

MD5 (Message Digest Algorithm 5)

MD5 processes 512-bit blocks of input and produces a 128-bit output. It uses modular addition, nonlinear functions (F, G, H, I), and left bitwise rotations.

Core step for each operation:

$$a = b + \text{leftrotate}(a + F(b, c, d) + M_i + K_i, s)$$

where M_i is the current 32-bit chunk of the message, K_i is a constant, and s is the shift amount.

SHA-3 (Keccak)

SHA-3 uses a sponge construction rather than the traditional Merkle–Damgård construction.

Process:

- **Absorbing phase**: XOR input blocks into a fixed-size state and apply a permutation,
- **Squeezing phase**: Extract output bits from the state.

Mathematical structure:

- State matrix $S[x, y, z]$ where (x, y) are plane coordinates and z is a bit position,
- Permutation steps involve:
  - θ (theta) for mixing columns,
  - ρ (rho) for rotating bits,
  - π (pi) for rearranging bits,
  - χ (chi) for nonlinear step,
  - ι (iota) for adding round constants.

Chapter 3

Comparative Analysis

Algorithm	Speed	Security	Scalability	Typical Use	Strengths	Weaknesses
AES	Very fast	Very secure	Excellent	Disk encryption, VPN	Fast, hardware acceleration	Key exchange challenges
DES	Moderate	Obsolete	Poor	Legacy systems	Easy to implement	Vulnerable to attacks
Blowfish	Fast	Moderate	Good	Password storage	Free and efficient	64-bit block size
RSA	Slow	Very secure	Moderate	Secure key exchange	No need to share secrets	Computationally intensive
ECC	Faster than RSA	Very secure	Excellent	Mobile security	Smaller key sizes	Complex math
Diffie-Hellman	Moderate	Secure (with authentication)	Excellent	Key exchange	Simple idea	Susceptible without authentication
SHA-256	Fast	Very secure	Excellent	Blockchain, password storage	Highly collision-resistant	Heavy computation
MD5	Very fast	Insecure	Poor	Legacy checksum	Easy to use	Broken security
SHA-3	Moderate	Very secure	Excellent	Future applications	Resistant to new attacks	Slightly slower

Interpretation and Recommendations

- **AES** is recommended for fast and secure file encryption due to its high speed, strong security, and excellent scalability. It is particularly suitable for applications requiring both high performance and robust data protection, such as disk encryption and VPNs.

- **RSA** should be used for secure key exchanges and digital authentication. While it is slow compared to symmetric encryption methods, it is highly secure and eliminates the need for pre-shared secrets. However, its computational intensity should be considered when performance is critical.

- For secure data hashing, **SHA-256** or **SHA-3** are ideal choices. Both algorithms provide excellent security and scalability. SHA-256 is commonly used in blockchain and password storage, while SHA-3 is resistant to new cryptographic attacks and can be applied in future-proof systems.

- Avoid using **DES** and **MD5** in new systems. DES is obsolete and vulnerable to modern cryptographic attacks, while MD5 is broken and susceptible to collision attacks. These algorithms should only be used in legacy systems where no other option is available.

Chapter 4

Recent Advances and Future Directions

Recent advances in cryptography emphasize the development of encryption methods that can withstand the threats posed by quantum computing. **Post-quantum cryptography** initiatives are leading to the creation of quantum-resistant algorithms such as NTRU, CRYSTALS-Kyber, and Lattice-based cryptography.

Another important area is the growth of **homomorphic encryption**, which allows computations on encrypted data without decrypting it, promising revolutions in cloud computing privacy. Additionally, the application of **blockchain** technology has dramatically boosted the need for secure, scalable hashing mechanisms.

Looking into the future, we can expect encryption systems to become more lightweight and efficient, ensuring security in constrained environments like IoT devices. Moreover, **Zero-Knowledge Proofs (ZKPs)** and **multi-party computation (MPC)** are gaining momentum to enhance privacy without compromising usability. **AI-driven cryptanalysis** may also uncover new vulnerabilities, prompting the need for ever-stronger algorithms and verification techniques.

Cryptography remains a dynamic, evolving field essential to the security and trustworthiness of the digital world.

Conclusion

This comprehensive exploration of encryption, decryption, and hashing illustrates the profound mathematical beauty and practical necessity behind secure data handling. Through historical evolution, theoretical grounding, algorithmic comparison, and future prospects, it becomes evident that cryptography is a dynamic field, critical for safeguarding the digital future.

Bibliography

[1] Bruce Schneier, *Cryptography Engineering: Design Principles and Practical Applications*, Wiley, 2010.

[2] William Stallings, *Cryptography and Network Security: Principles and Practice*, 7th Edition, Pearson, 2016.

[3] Alfred J. Menezes, *Handbook of Applied Cryptography*, CRC Press, 1997.

[4] National Institute of Standards and Technology (NIST), *FIPS PUB 197: Advanced Encryption Standard (AES)*, 2001. `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf`

[5] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21, No. 2, February 1978. `https://dl.acm.org/doi/10.1145/359340.359342`