

Block Cipher and DES (Attacks on DES)

Prepared by Dr. Sumaya Alkubaisi

FNT625
Cybersecurity Concepts

Lecture 5



Lecture

5

Learning Goals

By the end of this lecture, you will be able to:

- identify and explain common cryptanalytic attacks on DES.
- Understand why DES became insecure over time.
- Recognize how 3DES mitigated some vulnerabilities.
- Relate DES attacks to modern FinTech encryption practices.

Quick Recap: What is DES?

- DES (Data Encryption Standard) is a symmetric block cipher that encrypts data in 64-bit blocks using a 56-bit secret key.
- It follows the Feistel cipher structure, which divides data into two halves and processes them through 16 rounds of transformations.
- Each round applies substitution (confusion) and permutation (diffusion) using key-dependent operations like XOR and S-box lookups.
- The same key is used for both encryption and decryption — making it a symmetric algorithm.
- Developed by IBM in the 1970s and adopted by NIST in 1977 as the U.S. data encryption standard.
- DES was widely used in banking, financial transactions, and secure communications for over two decades.
- Eventually, advances in computing power and cryptanalysis made its 56-bit key too short to be secure.

Understanding DES Vulnerabilities

Why Study Attacks on DES?

- DES was once the standard for government, banks, and ATMs.
- Its analysis shaped the design of modern ciphers.
- Learning DES weaknesses helps us understand cryptographic security evolution.

Key Weakness of DES

- DES uses a 56-bit key, which means there are 2^{56} possible keys (about 72 quadrillion).
- In the 1970s, that was safe because computers were slow.
- But today, computers can try all keys (brute force) in a short time.
- So the problem isn't the DES algorithm itself, but its short key size — it's too easy to guess with modern technology.

Timeline of DES Vulnerability

- **1977:**
DES (Data Encryption Standard) was officially adopted by the U.S. government as a national encryption standard.
It was considered strong and reliable at the time, mainly used for protecting sensitive—but not classified—information.
- **1980s–1990s:**
As computers became faster, researchers started warning that DES's 56-bit key might be too short to stay secure for long.
However, it was still widely used in banking, communication, and government systems.
- **1998:**
The Electronic Frontier Foundation (EFF) built a special computer called “Deep Crack”, designed only to test all possible DES keys.
It successfully broke a DES-encrypted message in just 56 hours, proving that DES could be brute-forced with affordable hardware.

Timeline of DES Vulnerability

- **1999:**

A collaboration between Distributed.net (a network of volunteers sharing computer power) and the EFF repeated the test.

With combined computing power from many users, they broke DES in 22 hours, showing that even average computers could defeat it with teamwork.

- **Today:**

With modern cloud computing and GPU-based attacks, DES can be cracked in minutes or even seconds.

This makes DES completely unsafe for any real-world encryption use.

As a result, DES has been replaced by 3DES (Triple DES) and later by AES (Advanced Encryption Standard).

Real-World Impact

- **Banking and ATMs:**
In the past, many ATMs, credit card systems, and financial networks used DES to protect transactions between banks and customers.
Each transaction was encrypted with a 56-bit DES key to keep it secret.
- **Security Problem:**
As computing power grew, hackers could guess all possible keys using brute-force attacks. This meant that sensitive financial information, such as PIN codes, account balances, or money transfers, could be exposed if DES was used.
- **Transition to Stronger Systems:**
To fix this issue, the industry moved to Triple DES (3DES) — which applies DES encryption three times with different keys, making it much stronger.
Later, the U.S. adopted AES (Advanced Encryption Standard), which uses key sizes of 128, 192, or 256 bits — far beyond what brute-force computers can handle.

Lecture

5

Brute-Force Attack Definition

- A brute-force attack is when a hacker tries every possible key until the correct one is found.
- Think of it like trying every combination on a lock until it opens.
- It works because the attacker doesn't need to know anything about the message—just the key.

Example

- Suppose we have a ciphertext: DA02CE3A89ECAC3B
- The goal is to find the key that decrypts it to the message “HELLO”.
- With DES, the key is 56 bits long → that means there are about 72 quadrillion possible keys.
- On average, a hacker only needs to try half of them → 36 quadrillion attempts.
- This shows why DES is vulnerable today—computers are fast enough to do this in a short time.

Lecture

5

Practical Brute-Force Power Growth

- 1980s: Mainframe computers → would take centuries to break DES.
- 1998: EFF built Deep Crack → broke DES in 56 hours.
- 2025: Modern cloud GPUs → can break DES in less than 10 minutes.

Lesson:

As computers get faster, key length that was once secure is no longer safe.

- **Use longer keys:** DES uses 56 bits; alternatives like 3DES or AES use 112–256 bits.
- **Rotate keys often:** Change encryption keys regularly to reduce risk.

Example:

- A company encrypts its customer database with a key that changes every 3 months.
- Even if a hacker starts trying to brute-force the key, by the time they get close, the key has already changed.
- Think of it like changing the password to your Wi-Fi regularly so old attempts become useless.

Mitigation

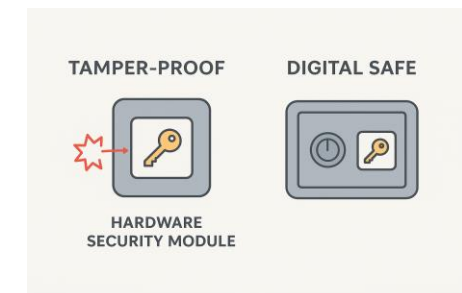
- **Use hardware encryption modules (HSMs):** Specialized devices protect keys from attacks.

Example:

- A bank stores encryption keys in a secure HSM device rather than on a regular computer.
- The HSM is tamper-proof and performs encryption without revealing the key to the main system.
- Like a digital safe: you can use the key, but no one can steal it directly.

- ❖ **Tamper-proof** means the device is built so that if someone tries to open, hack, or physically interfere with it, it will destroy the secret inside or stop working.

Example: An HSM (hardware security module) will erase the encryption key if someone tries to open it.



- **Combine security methods:** Don't rely only on key length.

Example:

- Use AES-256 encryption + multi-factor authentication (MFA) + frequent key rotation.
- Even if someone tries brute-force on AES, they also need to pass MFA to access data.
- Think of it as a lock, a guard, and a motion sensor protecting the same door—breaking one is not enough.

Lecture

5

Lesson from Brute Force

- Security depends on key length—but it's not forever.
- Moore's Law: Computational power roughly doubles every 2 years.
- Even strong keys eventually become breakable if technology keeps improving.
- Always update encryption standards and use longer keys to stay safe.

Lecture

5

What Is Differential Cryptanalysis?

- A chosen-plaintext attack that studies how differences in plaintexts propagate through the cipher.
- Focuses on finding statistical patterns in how plaintext differences affect ciphertext differences.
- Goal: Infer information about the secret key by analyzing these patterns.
- Significance: One of the first successful attacks showing that DES's internal structure could be exploited systematically.

How Differential Cryptanalysis Works

- Select pairs of plaintexts with specific differences (ΔP).
- Encrypt both under the same secret key.
- Compare the ciphertext differences (ΔC).
- Look for patterns in ΔC that reveal information about subkey bits used in S-boxes.
- Repeat with multiple plaintext pairs to increase confidence in key deduction.

How Differential Cryptanalysis Works (Example)

Step 1 – Choose plaintexts with a known difference

- Plaintext 1 (P1): 10101010
- Plaintext 2 (P2): 10101110
- Difference: only one bit is flipped (in this case, the 5th bit from the left).
- This difference is called ΔP (delta P).

Why one bit? Small differences make it easier to see how changes propagate through the cipher.

Step 2 – Encrypt both plaintexts

- Encrypt P1 → Ciphertext 1 (C1)
- Encrypt P2 → Ciphertext 2 (C2)

These encryptions use the same secret key.

How Differential Cryptanalysis Works (Example)

Step 3 – Compare ciphertexts

- Calculate $\Delta C = C1 \oplus C2$ (XOR of the ciphertexts).
- ΔC shows which ciphertext bits changed due to the one-bit difference in plaintext.
- Some bits may change predictably; others may appear random.
- The “predictable changes” are where S-boxes leak information about key bits.

Step 4 – Analyze S-box behavior

- DES divides plaintext into blocks and passes them through S-boxes in each round.
- Each S-box transforms 6 input bits into 4 output bits.
- By observing ΔC patterns, you can see which S-boxes caused certain output differences.
- This lets attackers narrow down possible subkey bits for that round.

How Differential Cryptanalysis Works (Example)

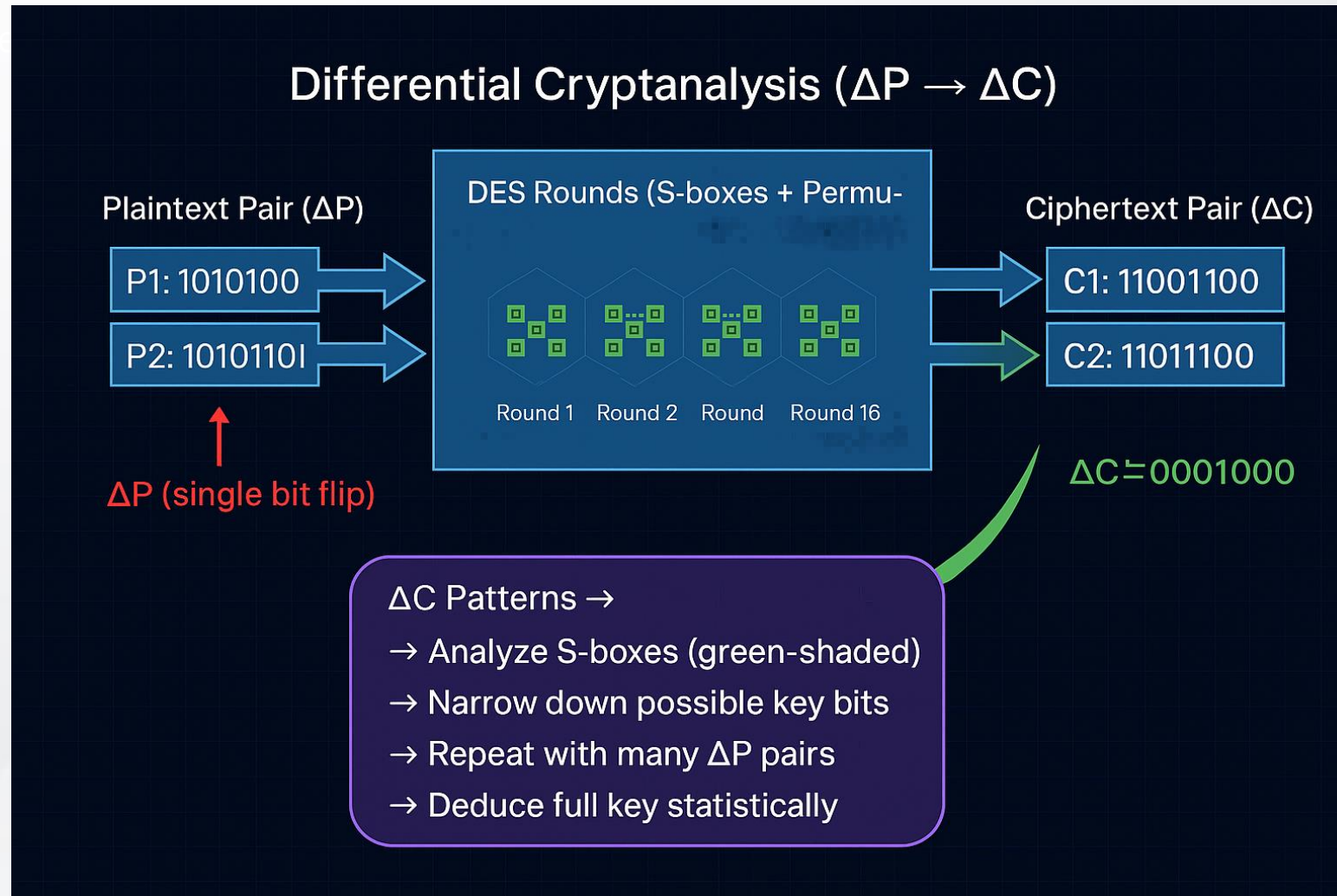
Step 5 – Repeat and accumulate statistics

- One pair of plaintexts is usually not enough.
- The attacker collects many pairs with specific differences.
- By analyzing statistical patterns, the attacker can gradually deduce the full key.

□ Think of DES as a maze with multiple doors (S-boxes).

Changing one bit in the input is like opening one door differently.

By watching which exits (ciphertext bits) change, you can figure out which internal doors (key bits) caused it.



The process of Differential Cryptanalysis, focusing on a single-bit plaintext difference.

Why Differential Cryptanalysis Dangerous

- Exploits structural weaknesses of DES S-boxes.
- Reduces brute-force effort from $2^{56} \rightarrow \sim 2^{47}$.
- Only requires around 50 carefully chosen plaintexts—feasible for a determined attacker.
- Reveals that even well-designed ciphers can leak key information if patterns exist.

Defense Mechanism

- DES S-boxes were designed to resist differential cryptanalysis, though this wasn't publicly known initially – small differences in plaintext don't lead to easily predictable differences in ciphertext.
- Modern ciphers like AES are explicitly tested against differential attacks.
- Additional measures: larger keys, more complex S-boxes, and multiple rounds to diffuse differences.

FinTech Example

Scenario: A bank encrypts every transaction before storing it.

Risk: If an attacker could pick specific transactions and see how the encrypted results look, they might notice patterns.

Why it matters: Those patterns could help the attacker figure out the secret key that encrypts all transactions.

Defense: banks restrict direct encryption access and monitor transaction anomalies.

Differential Cryptanalysis' Limitations

- Needs system access to perform chosen-plaintext attack.
- Requires careful selection of multiple plaintext pairs.
- Practically hard to perform without insider or API-level access (the part of the bank's system that allows programs or clients to interact with it—like submitting transactions or checking balances).
- Not a universal attack—works mainly against structured ciphers like DES.

Lecture

5

Summary of Differential Cryptanalysis

- First major analytical attack on DES.
- Reduced the effective key search space but did not fully break DES — instead of testing all 2^{56} , the attacker might only need to test a smaller number, like 2^{47} or 2^{43} , depending on the technique.
- Highlighted importance of S-box design and key length.
- Inspired development of stronger cipher designs and evaluation techniques.

In conclusion: the Differential Cryptanalysis attack made DES easier to crack, but not completely broken. It showed that theoretical weaknesses existed without immediately recovering the full key.

Linear Cryptanalysis

- Linear cryptanalysis is a known-plaintext attack, meaning the attacker already knows some plaintexts and their matching ciphertexts.
- The goal is to find simple linear (XOR-based) relationships between the plaintext bits, ciphertext bits, and key bits.
- By spotting statistical patterns in these relationships, the attacker can make educated guesses about key bits.

In conclusion: Linear cryptanalysis finds tiny mathematical patterns between input and output bits.

By counting how often those patterns appear, attackers can slowly guess parts of the secret key.

Historical Significance

- Developed by Mitsuru Matsui in 1993, linear cryptanalysis was the first practical statistical attack that could analyze DES more efficiently than trying all keys by brute force.
- Matsui showed that, with enough known plaintexts, it was possible to recover key bits and reduce the search space, proving that DES had small but real weaknesses.
- This was a major breakthrough because, until then, DES was thought to be nearly unbreakable except by brute force.
- Together with differential cryptanalysis, this attack changed how cryptographers evaluated ciphers, leading to more scientific and transparent testing methods.
- These studies directly influenced the design of modern ciphers like AES, which were built to resist both differential and linear attacks.

Lecture

5

How Linear Cryptanalysis Works

- The attacker looks for a simple rule that connects one plaintext bit (P), one ciphertext bit (C), and one key bit (K).
- Example of a rule:
 $P \text{ bit} \oplus C \text{ bit} = K \text{ bit}$
("⊕" means XOR — like checking if two bits are the same or different.)
- The attacker tests this rule on many plaintext–ciphertext pairs (for example, millions of them).
- If the rule is true a little more than half the time (like 52% instead of 50%), it means there's a small bias — a clue about what the key bit might be.
- By repeating this for different bits, the attacker can slowly guess parts of the secret key.

Complexity

- To make these small biases statistically meaningful, the attacker needs about 2^{43} known plaintext–ciphertext pairs.
- That's a huge number — not practical for ordinary attackers, but it proves that DES isn't perfectly random.
- The attack shows DES's structure has slight linear patterns that can be mathematically exploited.

Lecture

5

Linear & Differential Cryptanalysis: Theory vs Practice

Did the attacks happen in theory? — Yes.

Researchers demonstrated linear and differential cryptanalysis in papers and experiments. Those attacks proved DES had mathematical weaknesses.

Did attackers break DES in the real world because of those attacks? — Not exactly.
The linear/differential attacks were mostly academic tools that showed weaknesses but required huge amounts of data or computation, so they weren't widely used by everyday criminals.

Was DES broken in practice anyway? — Yes — by brute force.

Lecture
5

Comparison: Differential vs Linear

Attack	Data Needed	Type	Reduction in Effort	Practicality
Differential	Chosen plaintext– ciphertext pairs	Non-linear differences	$\sim 2^{47}$	Moderate
Linear	Known plaintext– ciphertext pairs	Linear equations	$\sim 2^{43}$	Moderate

Summary:

- Differential = studies differences in outputs.
- Linear = studies linear relationships between bits.
- Both reduce brute-force effort but are mostly theoretical proofs of weakness, not real-world tools for breaking banking systems.

Lecture

5

Beyond Math: Implementation & Side-Channel Attacks

- Even if DES' mathematics are strong, real systems (hardware or software) can leak information.
- These are side-channel attacks: they use how a device computes (time, power, errors, etc.), not the cipher itself.
- Important point: secure algorithm + insecure implementation = insecure system.

In conclusion:

- Protect both the algorithm and the device that runs it.
- Side-channel and implementation attacks can affect any cryptographic algorithm, even the most modern ones like AES, RSA, or ECC.
- Hardware Security Modules (HSMs) in banking use physical shielding and monitoring to prevent these attacks.

Lecture

5

Timing Attacks

- Timing Attacks Measure how long encryption or decryption takes for different inputs.
- Why it works: Some operations take slightly different time depending on secret data (e.g., branching on a key bit).

Example: If processing a “1” bit takes longer than a “0” bit, many timing samples can reveal key bits.

- If an attacker measures how long the program takes, they can guess the value of key_bit.
- Long time → key bit = 1
- Short time → key bit = 0
- That’s what “branching on a key bit” means: the program’s path changes depending on the key, and the time difference leaks information.
- Think of it like a door that opens slower for some keys than others — by timing how long it takes, you can figure out which key it is.

Timing Attacks

Countermeasures:

- Write constant-time code (no data-dependent branches).
- Use fixed-latency algorithms or add carefully controlled noise.
- Run sensitive operations in protected environments (HSMs).

Timing leaks are stealthy — they don't require physical access, only the ability to measure response times.

Power Analysis

Power Analysis: When a device encrypts data, it uses electricity.

The amount of power it uses can slightly change depending on the secret key and the data.

Example:

Suppose a device does a “1” operation with more electricity than a “0” operation.

By measuring the power while it encrypts many messages, an attacker can guess some key bits.

Types:

Simple Power Analysis (SPA): looks at one power trace and tries to see obvious patterns.

Differential Power Analysis (DPA): looks at many power traces and uses statistics to find tiny correlations with key bits.

Power Analysis

Countermeasures:

Masking: mix in random values so power patterns don't match the real key.

Noise injection: add random electrical noise to hide patterns.

Hardware design: make the circuit use the same power for all operations so no patterns appear.

Countermeasures are like making all numbers use the same electricity, or adding fake electricity patterns to confuse the observer.

Fault Injection Attacks

Fault Injection Attacks intentionally cause errors (faults) in the device during encryption and analyze incorrect outputs.

How the attack works :

- The attacker runs an encryption on the target device while causing a controlled glitch (e.g., short voltage spike, clock glitch, or laser pulse).
- The device produces a faulty ciphertext instead of the correct one.
- The attacker compares the faulty ciphertext with a correct ciphertext for the same input.
- The difference often reveals information about the device's internal state at the moment of the fault — and that can be used to recover subkeys or the full key.

Fault Injection Attacks

Why it works: Faults can expose internal state or cause predictable corruptions that reveal key bits.

Example: One faulty encryption result plus a correct result can let the attacker solve for a subkey.

Mitigations:

- Redundancy: perform operations twice and compare results.
- Error-detection: checksums, MACs, or consistency checks.
- Tamper-detection: sensors that wipe keys if a physical attack is detected.
- Slide note: Fault attacks require physical access or close proximity — higher effort but very powerful.

Beyond DES – 3DES and Legacy Use

Why DES Became Obsolete?

- 56-bit key too short for modern processing power.
- Vulnerable to multiple analytical and hardware attacks.

Beyond DES – 3DES and Legacy Use

Triple DES (3DES)

- It applies DES three times in an Encrypt–Decrypt–Encrypt (EDE) sequence using 3 keys (K1, K2, K3).
- The effective key length is 112 or 168 bits, making it much stronger than regular DES (56 bits).
- It was designed as a temporary fix to extend DES's life.
- Still based on DES, so it's slower and less efficient than modern algorithms.
- Today: It's largely phased out — most systems now use AES, which is faster, stronger, and supports larger block sizes (128-bit).
- 3DES is deprecated by major standards (like NIST) for new applications due to its performance limits and smaller block size.

3DES Example

- Encrypt with K1 $\rightarrow$ Cipher1
- Decrypt Cipher1 with K2 $\rightarrow$ Cipher2
- Encrypt Cipher2 with K3 $\rightarrow$ Final Cipher

This “EDE” sequence balances backward compatibility with DES.

Lecture

5

How attackers get ciphertext — brute-force enablers

- Stolen DBs or backups
- Network capture
- Exposed logs or debug endpoints
- Compromised endpoints / malware
- Insider theft or third-party breach
- ATM/POS compromise or physical theft
- Misconfigured cloud storage (public snapshots)

Lecture

5

Quick prevention steps

- Protect keys with HSMs and never store keys alongside ciphertext.
- Use separate keys for environments (production, development/testing) and rotate keys regularly.
- Avoid logging raw ciphertext where possible; redact or encrypt logs.
- Harden endpoints & servers; use EDR, malware detection, and patch promptly.
- Limit and monitor privileged access (least privilege + auditing).
- Secure cloud storage (S3 policies, remove public access, IAM least privilege).
- Use tokenization or format-preserving encryption for sensitive fields so raw ciphertext is less useful.
- Monitor for unusual bulk data access and exfiltration patterns (IDS/UEBA).
- Physical security for backups and anti-skimming controls for ATMs/POS.

How attackers get chosen-plaintext — Differential Cryptanalysis

- Attacker has access to an API or a user account
- Insider or compromised client/device
- Test or development environment using real production keys
- Client interfaces that accept flexible/custom requests
- A shared or public “encryption channel” inside the system
- Integration with an untrusted third party
- Physical attacks or compromised input points (e.g., hacked ATMs)

Quick prevention steps

- Don't expose a public encryption service — restrict and authenticate all access to any encryption endpoints.
- Use separate keys for test/dev and production — never use production keys in non-production environments.
- Log and monitor encryption interfaces; apply rate limits and IDS — detect unusual patterns and block abuse early.

Lecture

5

Recap

- DES: 64-bit block cipher using 56-bit key.
- Main vulnerabilities: short key, predictable structure.
- Brute-force feasible by late 1990s.
- Differential and Linear Cryptanalysis reduced complexity of attacks.
- Side-channel attacks target physical or timing leaks.
- 3DES extended DES's life but is now deprecated.
- Encryption must evolve with technology.
- Strong key management and algorithm choice are vital for secure financial systems.

References

- William Stallings, Cryptography and Network Security, 7th Ed., 2022.
- Cyber Security Tools, 2022.
- Computer Forensics & Cyber Crime, 2021.
- NIST SP 800-67 Rev. 2 (Recommendation for Triple-DES).
- EFF Deep Crack Report (1998).