Lecture Notes: Generic Attacks & Security Models

Course: INFO-F-537 Cryptanalysis (Topic 1)
Document compiled by Luca Volonterio

Based on Course Materials, *Slides-AlFardan*, *Notes-20250923* & *Hashing.pdf*

January 10, 2026

Abstract

This document serves as a comprehensive study guide for **Topic 1** of the oral exam. It covers the taxonomy of attacks, theoretical foundations (PRP/SPRP), formal security models (IND-CPA, EU-CMA), specific generic attacks on Block Cipher Modes, Hash Functions, and Sponge Functions, and case studies on primitive failures (RC4) and algorithmic tools (Missing Difference Problem).

Contents

1 Introduction: Scope and Taxonomy

1.1 What is a Generic Attack?

A **Generic Attack** is an attack that works independently of the underlying cryptographic primitive (e.g., AES, Keccak-f). It treats the primitive as an ideal "Black Box" and exploits only the parameters of the **Mode of Operation** or **Construction**.

- **Target:** Structure, Key size (k), Block size (n), Capacity (c), Tag size (τ).
- **Contrast:** *Shortcut Attacks* (e.g., Differential Cryptanalysis) exploit internal flaws of the specific primitive.

1.2 Taxonomy: How to Describe an Attack

Based on the course methodology, an attack is defined by three components:

1. **Goal:** e.g., Key Recovery, Distinguishing, Forgery.
2. **Data Model:** e.g., Known Plaintext, Chosen Ciphertext (CCA).
3. **Complexity:**
    - **Time (t):** Computational effort.
    - **Data (d):** Amount of blocks processed.
    - **Success Probability (ε):** Likelihood of success.

—

2 Theoretical Foundations & Algorithmic Tools

2.1 Ideal Primitives: PRP vs SPRP

To analyze generic security, we assume the components are mathematically ideal.

- **PRP (Pseudo-Random Permutation):** A Block Cipher E_K is a PRP if it is indistinguishable from a random permutation π under encryption queries only. *Implication:* Necessary for passive security (IND-CPA).
- **SPRP (Strong Pseudo-Random Permutation):** A Block Cipher is an SPRP if it remains indistinguishable even when the adversary has access to **both** Encryption and Decryption oracles. *Implication:* Necessary for resistance against active attacks (IND-CCA).

3 Formal Security Models

3.1 Confidentiality Models

IND-CPA (Indistinguishability under Chosen Plaintext Attack): The adversary chooses P_0, P_1. Challenger returns $C_b = E_K(P_b)$. Adversary must guess b. *Requirement:* Randomized encryption (ECB fails this).

IND-CCA (Indistinguishability under Chosen Ciphertext Attack): The adversary can also query a decryption oracle for any ciphertext $C \neq C_b$. This models active attackers who can modify traffic.

3.2 Authenticity Model: EU-CMA

Existential Unforgeability under Chosen Message Attack.

- **The Game:** Adversary requests tags T_i for messages M_i.
- **Win Condition:** Output a valid pair (M^*, T^*) for a **new** message M^*.
- **Generic Bound:** For a tag of length τ, the best generic attack is random guessing: $P(\text{success}) \approx 2^{-\tau}$.

3.3 Indifferentiability Framework (Hashing Context)

(From INFOF537-Hashing.pdf) For hash functions and sponge constructions, standard indistinguishability is often insufficient because hash functions have no key. We use the **Indifferentiability Framework** (Maurer et al.).

Definition: A construction C (using ideal primitive P) is indifferentiable from an ideal Random Oracle $\mathcal{R}$ if there exists a simulator S such that no distinguisher can tell apart the pair (C^P, P) from $(\mathcal{R}, S^{\mathcal{R}})$.

- **Implication:** If C is indifferentiable from a Random Oracle, it can replace a Random Oracle in any protocol without loss of security (up to the bound).
- **Sponge Bound:** The Sponge construction is indifferentiable from a RO up to $N^2/2^{c+1}$.

—

4 Deep Dive: The Missing Difference Problem

(Based on course notes: Leurent & Sibleyras context)

While often confusingly named, the **Missing Difference Problem** is a generic algorithmic tool used in cryptanalysis, particularly effective against 64-bit block ciphers (like 3DES or Blowfish) in modes like CBC or CTR. It relies on the birthday paradox to find collisions or specific difference patterns in a large dataset.

4.1 Definition

Let f be a function (e.g., an encryption oracle E_k) and Δ be a target constant. The problem is to find two inputs x and y such that:

$$f(x) \oplus f(y) = \Delta \tag{1}$$

If $\Delta = 0$, this reduces to the classic **Collision Problem**.

4.2 Application: Attacking 64-bit Block Ciphers

In the context of Leurent & Sibleyras (Sweet32 attack), this problem is used to recover a secret value S (e.g., an internal state or key-dependent value) by observing many plaintext/ciphertext pairs.

The Attack Logic

1. **Setup:** The attacker collects a large set of data pairs (D, i), where D is some data and i is a counter or index.
2. **Observation:** The attacker computes or observes values derived from the encryption:

$$C_{D,i} = E_k(D||i) \oplus S$$

   Here, $C_{D,i}$ is the observed ciphertext block, E_k is the block cipher encryption, and S is the unknown secret target.

3. **The "Missing Difference" Search:** The attacker looks for two distinct inputs (D, i) and (D', i') such that the encryption outputs collide:

$$E_k(D||i) = E_k(D'||i')$$

If such a collision occurs, then the XOR sum of the corresponding observed ciphertexts reveals information:

$$\begin{aligned} C_{D,i} \oplus C_{D',i'} &= (E_k(D||i) \oplus S) \oplus (E_k(D'||i') \oplus S) \\ &= 0 \oplus (S \oplus S) \\ &= 0 \end{aligned}$$

Wait! If the result is 0, we found a collision. But the "Missing Difference" problem generalizes this: we might look for a specific non-zero difference Δ that allows us to cancel out terms or verify a guess for S.

In many practical scenarios (like CBC collision attacks), finding the collision ($f(x) \oplus f(y) = 0$) allows the attacker to deduce the XOR of the corresponding plaintexts:

$$P_i \oplus P_j = C_{i-1} \oplus C_{j-1}$$

This is because the internal state collision eliminates the unknown key-dependent permutation, leaving only known values.

4.3 Complexity

The power of this attack lies in its generic nature. It does not require analyzing the S-boxes of the cipher.

- **Data Complexity:** $O(2^{n/2})$ blocks.
- **Time Complexity:** $O(2^{n/2})$ operations.

For a 64-bit block cipher ($n = 64$), the attack becomes feasible after collecting $\approx 2^{32}$ blocks (about 32 GB of data). This is the "Sweet32" threshold.

—

5 Generic Attacks on Block Cipher Modes

5.1 The General Security Claim (AES-CBC)

For a mode like AES-CBC, the security against a generic adversary is bounded by:

$$\varepsilon(t, d) \leq \underbrace{\frac{t}{2^k}}_{\text{Key Search}} + \underbrace{\frac{d^2}{2^n}}_{\text{Birthday Bound}} \tag{2}$$

5.2 Analysis of Terms

$\frac{t}{2^k}$ **(Exhaustive Key Search):** Trying all keys. Linear success probability. Goal: Key Recovery.

$\frac{d^2}{2^n}$ **(Block Collision / Birthday Bound):** The probability of finding a collision in n-bit blocks.

- In **CBC** ($C_i = E_K(P_i \oplus C_{i-1})$), a collision in the inputs to the block cipher allows distinguishing the scheme.
- **Limit:** For AES ($n = 128$), security degrades when $d \approx 2^{64}$ blocks.

5.3 Specific Vulnerabilities

- **ECB Mode:** Deterministic. Fails IND-CPA ($O(1)$ complexity).
- **CBC Padding Oracle:** Exploits decryption errors ("Invalid Padding") to recover plaintext. Breaks IND-CCA.
- **CTR Nonce Reuse:** If $(Key, Nonce)$ is reused, $C_1 \oplus C_2 = P_1 \oplus P_2$. Catastrophic loss of confidentiality.

—

6 Generic Attacks on Hash Functions

(Based on INFOF537-Hashing.pdf)

For a hash function $h : \{0,1\}^* \to \{0,1\}^n$, generic attacks depend solely on the output length n.

6.1 Security Definitions

1. **Preimage Resistance (One-Wayness):** Given y, find x such that $h(x) = y$.
2. **Second Preimage Resistance:** Given x, find $x' \neq x$ such that $h(x) = h(x')$.
3. **Collision Resistance:** Find any pair x, x' such that $h(x) = h(x')$.

6.2 Generic Bounds

- **Preimage / 2nd Preimage:** Requires $\approx 2^n$ operations (Exhaustive Search).
- **Collision:** Requires $\approx 2^{n/2}$ operations (Birthday Paradox).

6.3 Specific Hashing Constructions

- **Merkle-Damgård:** Iterative construction (used in SHA-1, SHA-2). Vulnerable to *Length Extension Attacks* if used as a MAC ($H(K||M)$) without proper padding.
- **Sponge Functions:** (See next section). Resistant to Length Extension by design.

—

7 Generic Attacks on Sponge Functions

(Updated with info from Hashing.pdf)

7.1 The Sponge Construction

Built on a permutation f of width $b = r + c$.

- **Rate (r):** Visible part (input/output).
- **Capacity (c):** Hidden part (Security Parameter).

7.2 Generic Security Bound (Theorem)

The advantage of a generic adversary differentiating a random sponge from a random oracle is bounded by:

$$\varepsilon \leq \frac{N^2}{2^{c+1}} \tag{3}$$

Where N is the number of calls to the permutation f.

Implications for Hashing (SHA-3):

- **Collision Resistance:** Limited by $2^{c/2}$ (Birthday bound on capacity).
- **Preimage Resistance:** Limited by $\min(2^n, 2^c)$ (where n is output length).
- **Indifferentiability:** Holds as long as $N < 2^{c/2}$.

7.3 Deep Dive: Duplex & SpongeWrap (AEAD)

SpongeWrap uses the Duplex construction for Authenticated Encryption.

Exam Tip: The Duplexing-Sponge Lemma: "An attack on a duplex object is also an attack on the corresponding sponge function." This means if c is large enough for the Sponge, the Duplex mode is secure.

Security Claim for SpongeWrap:

$$\varepsilon(t, d, q_{forge}) \leq \frac{t}{2^k} + \frac{d^2}{2^{c+1}} + \frac{q_{forge}}{2^\tau} \tag{4}$$

$\frac{t}{2^k}$**:** Brute force on Key.

$\frac{d^2}{2^{c+1}}$ **(State Recovery):** If data processed $D \approx 2^{c/2}$, an internal collision occurs. The attacker learns the state and recovers the Key.

$\frac{q_{forge}}{2^\tau}$ **(EU-CMA Term):** Probability of forging a MAC by guessing the tag τ.

—

8 Case Study: When Primitives Fail (RC4 Biases)

Based on course slides (AlFardan et al., 2013).

While generic attacks assume the primitive is perfect, the RC4 stream cipher provides a famous counter-example where the primitive exhibits Statistical Biases.

8.1 The Flaw

RC4 generates a keystream byte z_i. For an ideal random generator, the probability of any byte value x is 1/256.

$$\Pr[z_i = x] = \frac{1}{256}$$

However, analysis shows significant deviations (biases) for specific positions (e.g., z_1, z_2). The graph of $256 \times \Pr[z_i = x]$ shows sharp peaks > 1.0.

8.2 The Attack (Distinguishing & Plaintext Recovery)

This bias allows a Broadcast Attack (or Multi-session attack).

- **Scenario:** The same plaintext P is sent multiple times encrypted with different keys/nonces.
- **Mechanism:** Since $C = P \oplus z$, and z is biased, C leaks info about P.
- **Result:** By collecting enough ciphertexts (2^{30}), an attacker recovers the plaintext byte-by-byte using Bayesian statistics.

—

9 Summary & Oral Exam Pitch

Feature	**Block Cipher (CBC)**	**Sponge (KMAC)**
Ideal Primitive	PRP (E_K)	Random Permutation (f)
Critical Parameter	Block size n	Capacity c
Generic Bound	$d^2/2^n$ (Birthday on blocks)	$d^2/2^c$ (Birthday on state)
Key Recovery	$t/2^k$	$t/2^k$ (or State Recovery)

Exam Tip: How to explain in 1 minute: "Generic attacks treat the primitive as a black box. For Block Ciphers, security is limited by the block size n (Birthday limit $2^{n/2}$). For Sponge functions, security is limited by the Capacity c (Birthday limit $2^{c/2}$). We model security using IND-CPA/CCA for encryption and EU-CMA for authentication."

—

10 Self-Assessment Questions

Try to answer these questions aloud to simulate the exam environment.

Level 1: Definitions & Concepts

Q: Explain why Merkle–Damgård is vulnerable to length extension and how HMAC fixes it.
Answer: The student should recall that the chaining value of h is reused as internal state, so knowing $h(K\|M)$ lets an attacker continue the iteration. HMAC wraps the key inside two hash calls to break simple extension.

Q: Compare collision resistance and preimage resistance for an n-bit hash from a generic point of view.
Answer: The student should state the generic complexities $2^{n/2}$ vs 2^n and relate them to the birthday paradox vs exhaustive search.

Q: What is the fundamental difference between a Generic Attack and a Shortcut Attack?
Answer: A generic attack works even if the primitive (AES/Keccak) is perfect, exploiting only the mode/structure. A shortcut attack exploits internal mathematical flaws of the primitive.

Q: What is the difference between PRP and SPRP?
Answer: PRP requires indistinguishability from random under encryption queries only (IND-CPA). SPRP requires indistinguishability even with access to a decryption oracle (needed for IND-CCA).

Q: What is the EU-CMA model?
Answer: Existential Unforgeability under Chosen Message Attack. The attacker wins if they can generate a valid (Message, Tag) pair for a message they never queried.

Level 2: Analyzing Formulas

Q: In the AES-CBC claim $\varepsilon \leq \frac{t}{2^k} + \frac{d^2}{2^n}$, why does the second term use n instead of k?
Answer: Because it represents the Birthday Bound on block collisions. If we encrypt $d \approx 2^{n/2}$ blocks, collisions in the ciphertext occur, compromising security regardless of the key size k.

Q: Why is the security bound for Sponge functions often $2^{c/2}$?
Answer: Due to the Birthday Paradox applied to the internal state. The attacker needs to find an internal collision in the Capacity c. This takes $\sqrt{2^c} = 2^{c/2}$ operations.

Level 3: Advanced & Curveballs

Q: How do the RC4 biases violate the definition of a secure Stream Cipher?
Answer: A secure Stream Cipher implies the keystream is indistinguishable from random noise. The graphs show that $256 \times \Pr[z_i = x] \neq 1$, meaning some bytes are more probable than others, allowing plaintext recovery.

Q: In the Missing Difference Problem notes, we see $E_k(D||i) \oplus S$. What does this resemble?
Answer: It resembles a generic attack on a construction (like CBC-MAC collision search) where the attacker tries to eliminate the secret S by finding two inputs that produce the same mask or difference, reducing security to the birthday bound.

Q: What is Indifferentiability and why do we use it for Hash Functions?
Answer: Indifferentiability is a framework to prove that a construction (like Sponge) behaves like a Random Oracle even when the internal primitive is known. We use it because standard indistinguishability games don't apply well to keyless hash functions.