Lesson 9
Cryptographic attacks

A. Sorokin (Input | Output)

Self paced course

July 31, 2025

Cryptographic Attack

An **attack** is a specific method, process, or technique used to exploit a known (or suspected) vulnerability of a cryptosystem.

Goal of attack: to break the security of a cryptosystem

By performing a succesful attack an adversary can recover the key, decrypt a message, forge a signature, etc.

Attacks and vulnerability interaction

The relationship between attacks and vulnerabilities is a continuous cycle of discovery, exploitation, and mitigation.

(1) *A vulnerability enables an attack*: an attacker discovers or learns about a vulnerability and then develops or chooses a specific attack method to exploit it.

(2) *An attack reveals a vulnerability*: a successful demonstration of an attack proves that a vulnerability exists.

Attacks based on plaintext and ciphertext

The interaction between cryptosystem designers and their adversaries creates a constant feedback loop that drives the entire field of cryptography forward.

Step 1: a new cryptosystem (e.g., a cipher, a hash function) is proposed.

Step 2: Researchers and attackers try to find vulnerabilities in its design.

Step 3: If a vulnerability is found, they develop a practical attack to demonstrate the severity.

Step 4: The system is deemed broken and is deprecated (e.g., MD5, SHA-1, DES).

Step 5: A new, stronger system is designed to resist those specific attacks and vulnerabilities (e.g., SHA-2, SHA-3, AES).

Attacks based on plaintext and ciphertext

Attacks are classified based on what the attacker knows or can do:

Ciphertext-only attack: the attacker only has access to encrypted messages in a symmetric cryptosystem.

Known-plaintext attack (KPA): the attacker has access to both some plaintext and its corresponding ciphertext in a symmetric cryptosystem, and the goal is to deduce the key to decrypt other messages.

Chosen-plaintext attack (CPA): the attacker can choose any plaintext and get the corresponding ciphertext in a public-key cryptosystem.

Chosen-ciphertext attack (CCA): the attacker can choose any ciphertext and get its decrypted plaintext in a public-key cryptosystem, and the goal is to deduce information about the key or decrypt a different, target ciphertext.

Brute force attack

We describe how a standard **brute force** attack is performed for breaking a password.

Goal: find a password that matches a stolen password hash.

Method: the attacker generates every possible character combination (a **guess**) in a defined keyspace (e.g., all lowercase letters, all alphanumeric characters, etc.), hashes each guess using the same algorithm, and compares the resulting hash to the target hash.

Brute force attack

Problem: brute force is computationally expensive and incredibly slow for long, complex passwords; moreover, for each new target hash, the attacker must repeat the entire process from scratch.

Analogy. You need to find a specific word in a dictionary, but you don't know the word. A brute force approach would be to start at "A" and read every single word in the dictionary until you find the right one. You'd have to do this for every new word you need to find.

Improved brute force: precomputed tables

To speed up the brute force process, an attacker can **precompute** all hashes beforehand and store them.

Method: an attacker precomputes the hashes for every possible password in a given keyspace (e.g., all passwords up to 8 characters using alphanumeric symbols), then store these pairs in a giant database.

Attack: when the attacker gets a target hash, it is simply looked up in the precomputed database, and if it's there, the password is recovered.

Improved brute force: precomputed tables

Problem: the database becomes astronomically huge, which is impractical. (e.g., hashes for all 8-character passwords would require terabytes of storage).

Analogy. Instead of reading the dictionary every time, you write down every word and its page number on a gigantic piece of paper. Finding a word is now fast, but storing that gigantic piece of paper is a nightmare.

Optimized brute force: rainbow tables

Rainbow tables solve the storage problem of precomputed tables through a clever time-memory trade-off: they use less storage at the cost of a little more computation during the attack.

Method: instead of storing every single password-hash pair, a rainbow table stores only start and end of chains of passwords, where each chain starts with a random **seed password**.

Benefit: a single chain of 100000 passwords is represented by just two values, which is a massive reduction in storage.

Optimized brute force: rainbow tables

Attack: when an attacker has a target hash to crack, a process starts using the **reduction function** to see if the target hash can be *linked to any of the final hashes* stored in the table, and if so, the attacker can recreate that specific chain from the start to find the password that produced the original target hash.

Analogy: writing down only the first and last word of every chapter in the dictionary. To find a word, you figure out which chapter it might be in by working backwards from the last word, and then you only have to read that single chapter to find your word. You've traded the gigantic storage of the full index for the smaller storage of chapter headers and a bit of extra reading.

Problem: salting passwords (adding a unique value to each password before it is hashed) prevents using rainbow tables.

Collision attack

Hashing is a process that converts data into a fixed-length string, called a **digest**, that acts as a unique identifier of the original data.

The process of hashing is irreversible, but there are always inputs sharing the same output due to the pigeonhole principle. Such situation is called a **collision**.

If an attacker can find two messages with the same digest, they can:

- substitute a legitimate file by a malicious one without changing its verified hash;
- undermine digital signatures for messages sharing a digest;
- compromise certificate authorities and public key infrastructure.

Collision attack

Types of Collision Attacks:

classical collision attack: the attacker finds any two arbitrary messages that collide (computationally infeasible task);

chosen-prefix collision attack: the attacker finds two words sharing the same prefix (quite dangerous as it allows the colliding messages to have meaningful beginnings);

SHAttered attack (2017)

researchers demonstrated a practical collision for the SHA-1 hash function by producing two different PDF files sharing SHA-1 digest.

Length extension attack

A **length extension attack** is an attack against *Merkle-Damgård-based hash functions* (like MD5, SHA-1, SHA-256) when they are used for message authentication in a vulnerable way.

It allows an attacker who knows the hash of a message padded with a secret (*message authentication code (MAC)*) and the length of the secret to forge a valid MAC for a message that extends the original one.

Flickr API problem

This vulnerability was found in the Flickr API, which used the vulnerable MD5 scheme.

Collision and length extension attack comparison

A **hash collision** is a fundamental weakness in the hash function itself, showing it fails to provide unique fingerprints.

A **length extension attack** is a weakness in how the hash function is used (e.g., for authentication). The function itself can still be collision-resistant, but the way it's applied (e.g., a naive MAC) is insecure.

Solution: always use *modern, community recommended hash functions* (SHA-256, SHA-3) and *proper cryptographic constructions* (like HMAC) instead of inventing your own.

Padding oracle attack

Encryption of a long message is performed by breaking it into **blocks** of a fixed length, and if a block is less than that, the leftover need to be **padded**.

In cryptography, a **padding oracle** is a system that provides information about encrypted data without revealing the encryption key.

Padding oracle detection

(1) A web server encrypts cookies before sending them to a client.

(2) When the web server receives them back, it will decrypt them in order to process them and remove the padding.

(3) If invalid padding causes an error message or response latency, then you have a padding oracle to work with.

Padding oracle attack

Padding oracle scheme:

(1) the attacker sends modified ciphertexts to a server;

(2) server responds with “Padding Error” or “Decryption Failed”;

(3) the attacker uses these errors to guess plaintext byte-by-byte.

POODLE (2014)

Forced servers to downgrade to SSL 3.0, enabling padding oracle attacks.

Solution: always use authenticated encryption (AES-GCM, ChaCha20-Poly1305).

Side-channel attack

A **side-channel attack** is a type of security exploit that leverages information inadvertently leaked by a system (such as timing, power consumption, or electromagnetic or acoustic emissions) to gain unauthorized access to sensitive information.

Side-channel attack

General classes of side-channel attacks include:

(i) **cache attack** is based on attacker's ability to monitor cache accesses made by the victim in a shared physical system as in virtualized environment or a type of cloud service;

(ii) **timing attack** is based on measuring how much time various computations take to perform;

(iii) **power-monitoring attack (power analysis)** makes use of varying power consumption by the hardware during computation;

Side-channel attack

(iv) **electromagnetic attack** is based on leaked electromagnetic radiation, which can directly provide plaintexts and other information;

(v) **acoustic cryptanalysis** is an attack that exploits sound produced during a computation;

(vi) in **differential fault analysis** secrets are discovered by introducing faults in a computation.

(vii) in **data remanence attack** sensitive data are read after supposedly having been deleted (e.g. cold boot attack).

Side-channel attack

Examples

(1) Timing attack: a login system rejects passwords faster if the first character is wrong.

(2) Power analysis: breaking AES keys from a stolen IoT device.

Spectre/Meltdown (2018)

CPU cache timing was used to steal data from other processes.

Solution: Use constant-time algorithms, hardware mitigations.

Man-in-the-middle attack

In **man-in-the-middle (MITM) attack** an attacker positions themselves between two communicating parties (e.g., a user and a website).

Method: the attacker captures traffic (e.g., HTTP requests, unencrypted emails) and if the encryption is poor or absent, the attacker decrypts data and/or alters messages.

MITM attacks happen in unsecured Wi-Fi networks, compromised routers, or DNS hijacking.

Man-in-the-middle attack

Superfish (2015)

Lenovo pre-installed adware injected HTTPS-breaking certificates, enabling MITM attacks.

Solution: disable weak cipher suites, use certificate pinning and authentication.

Chess grandmaster attack

Chess grandmaster attack (parallel session hijacking) Named after chess hustlers who play multiple games simultaneously, using moves from one game in another. Such attack exploits authentication protocols where two sessions run in parallel:

(1) attacker initiates two sessions (e.g., one with a bank, one with the victim);

(2) bank sends a challenge (e.g., nonce) in Session 1;

(3) attacker forwards the challenge to the victim in Session 2;

(4) victim's response is used to authenticate Session 1.

Chess grandmaster attack

Early SSH

Early SSH implementations were vulnerable to chess grandmaster aattacks if session IDs were predictable.

To protect a cryptosystem from a chess grandmaster attack the following layered strategy should be implementated

- prevent initial access with patching and training;
- limit lateral movement via network segmentation and multi-factor authentication;
- detect activity with centralized monitoring;
- ensure resilience with tested, offline backups.

Mafia fraud attack

Mafia fraud attack (relay attack) is a form of real-time MITM attack where the attacker relays communications between two parties without decrypting anything. One of its possible forms is:

(1) a victim initiates action (e.g., tapping a credit card at a terminal);

(2) an attacker relays the signal to a second device (e.g., a fake terminal controlled by the attacker);

(3) a transaction is approved because the attacker never breaks crypto but just forwards messages.

Succesful mafia fraud attacks can lead to unauthorized payments (e.g., relayed NFC transactions), car theft (e.g., amplifying a key fob signal to unlock a vehicle).