SAPIENZA
UNIVERSITÀ DI ROMA

The Law of Large Numbers: meaning, simulations and applications in cybersecurity and data analysis

Faculty of Information Engineering, Informatics, and Statistics
Master's Degree in Cybersecurity

Christian Capitani
ID number 1905211

Advisor
Prof. Gastaldi

Academic Year 2025/2026

The Law of Large Numbers: meaning, simulations and applications in cybersecurity and data analysis

Course Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: capitani.1905211@studenti.uniroma1.it

Abstract

This thesis explores the Law of Large Numbers (LLN), a fundamental theorem of probability theory that describes the result of performing the same experiment a large number of times. Through the development of an interactive simulator based on web technologies, the convergence of relative frequency to theoretical probability is visually analyzed. Finally, the paper discusses the practical implications of this theorem in the field of Data Analysis and, specifically, in Cybersecurity, analyzing how the LLN underpins Intrusion Detection Systems (Anomaly Detection).

Contents

Chapter 1

Introduction

In statistics and probability, intuition suggests that collecting a sufficient amount of data leads empirical observations to stabilize around a "true" value. This concept, formalized by the Law of Large Numbers (LLN), is the pillar upon which insurance companies, casinos, scientific research, and, as we shall see, cybersecurity rely.

The objective of this paper is twofold:

1. **Visualizing convergence:** Using a computational approach, we will demonstrate how empirical frequency converges to theoretical probability as the number of experiments increases ($n \to \infty$).
2. **Applying the concept to security:** We will analyze how deviation from this convergence can signal a cyber attack or a cryptographic vulnerability.

Chapter 2

Theoretical Foundations

Before stating the Law of Large Numbers, it is necessary to precisely define the quantities involved and the type of "approach" that exists between them. In statistics, concepts of frequency and convergence take on different nuances compared to classical mathematical analysis.

2.1 Preliminary Concepts

2.1.1 Absolute Frequency and Relative Frequency

In the analysis of a repeated random experiment (such as a coin toss or the analysis of network packets), we distinguish between two types of counts:

- **Absolute Frequency (n_k):** This is the simple count of times a specific event E has occurred over a total of n trials. It is an integer $n_k \in \mathbb{N}$.

$$n_k = \sum_{i=1}^{n} \mathbb{I}(X_i = E)$$

  Where $\mathbb{I}$ is the indicator function which equals 1 if the event occurs, and 0 otherwise.

- **Relative Frequency (f_n):** This is the ratio between the absolute frequency and the total number of trials n. It is a real number between 0 and 1.

$$f_n = \frac{n_k}{n}$$

Fundamental Note: The Law of Large Numbers concerns **relative frequency**. Absolute frequency, in fact, does not converge to a fixed value; on the contrary, it tends to diverge (for example, after a million coin tosses, the absolute difference between heads and tails can be very large, but their ratio f_n will be extremely close to 0.5).

2.1.2 Mathematical Convergence vs. Stochastic Convergence

The term "converge" can be misleading if interpreted in a purely deterministic sense.

- **Mathematical Convergence (Analytical):** Refers to deterministic numerical sequences. A sequence a_n converges to a limit L if, from a certain point onwards, the terms of the sequence become arbitrarily close to L. Example: The sequence $a_n = 1/n$ converges to 0. There is no uncertainty: for $n = 1000$, the value will *definitely* be 0.001.

$$\lim_{n\to\infty} a_n = L$$

- **Stochastic Convergence (in Probability):** Refers to random variables. Saying that the relative frequency f_n converges to the probability p does not mean that, upon reaching $n = 1000$, the result will *definitely* be p. It means that the **probability** that the result is different from p becomes infinitesimal. Even with a billion coin tosses, there is a non-zero (albeit tiny) probability of obtaining an anomalous sequence. Stochastic convergence describes the behavior of the probability distribution, not of the single number.

$$\text{plim}_{n\to\infty} X_n = X$$

This distinction is crucial in **Cybersecurity**: security systems based on statistics (like IDS) do not offer absolute mathematical certainties (100% security), but stochastic guarantees (security with probability tending to 1).

2.2 The Law of Large Numbers (LLN)

There are two main versions of this theorem that formalize the concept of convergence.

Since in our paper we deal with Bernoulli-type experiments (success/failure, 1/0), we can specialize the general notation. We will therefore replace the sample mean $\bar{X}_n$ with the **relative frequency** f_n and the expected value μ with the **theoretical probability** p.

2.2.1 Weak Law

The Weak Law (historically known as *Bernoulli's Theorem*) states that for every margin of error $\epsilon > 0$, the probability that the relative frequency deviates from the theoretical probability by an amount greater than ϵ tends to zero as the number of trials n tends to infinity:

$$\lim_{n\to\infty} P\left(|f_n - p| > \epsilon\right) = 0$$

This concept is known as *convergence in probability*. In practical terms, it does not guarantee that a single deviation will not occur, but it says that observing one becomes increasingly unlikely as n grows.

2.2.2 Strong Law

The Strong Law is stricter and states that the relative frequency converges to the theoretical probability with probability 1 (i.e., "almost surely"):

$$P\left(\lim_{n\to\infty} f_n = p\right) = 1$$

While the Weak Law speaks of the probability of being close to the target at a precise instant n, the Strong Law describes the behavior of the entire sequence: by increasing the number of simulations in our software, the trajectory of the graph will inevitably stabilize on the theoretical probability line, without any further large and persistent oscillations.

Chapter 3

Simulations and Graphic Representation

To empirically verify the exposed theory, a software simulator was developed using JavaScript. The experiment consists of simulating the toss of a coin ($p = 0.5$) or a biased coin ($p \neq 0.5$) a number n of times.

3.1 Experiment Implementation

The core of the simulation lies in the generation of pseudo-random numbers. Below is an extract of the code used to generate the convergence trajectories:

```
function generateSimulationData(numTrials, numSimulations) {
    // ... setup arrays ...
    for (let j = 0; j < numSimulations; j++) {
        let currentScore = 0;
        for (let i = 1; i <= numTrials; i++) {
            // Simulate Bernoulli event (0 or 1)
            const result = Math.round(Math.random());
            if (result === 1) currentScore++;

            // Calculate current relative frequency
            scoreData.push(currentScore / i);
        }
        // ... store data ...
    }
    return seriesData;
}
```

Listing 3.1. Simulation data generation

3.2 Results Analysis

3.2.1 Case 1: High Variance (Small Numbers)

Performing a single simulation with few tosses ($n = 50$), we observe a strong initial oscillation. The frequency is not stable and can deviate significantly from 0.5. This represents statistical "noise".

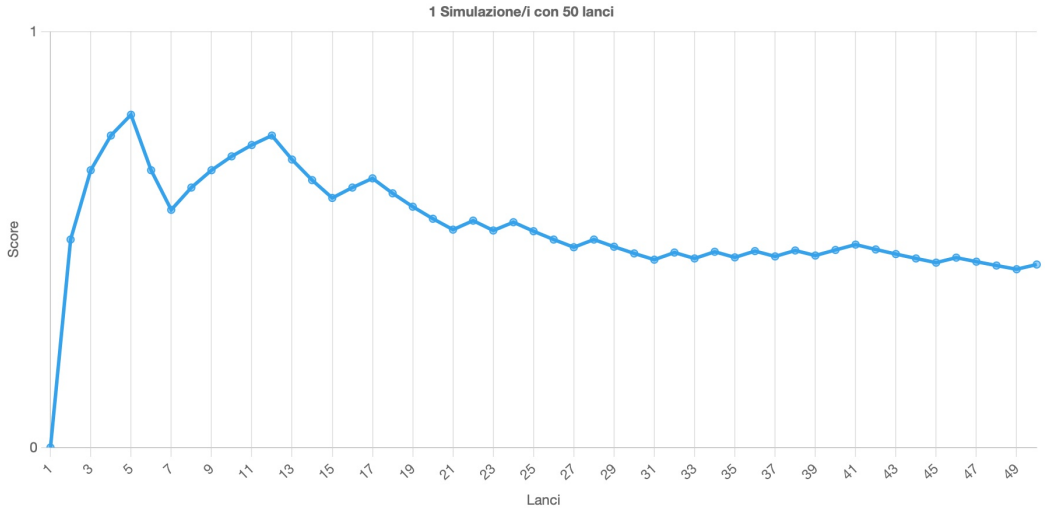

Figure 3.1. Single simulation with 50 tosses: strong initial oscillation.

3.2.2 Case 2: Convergence (Large Numbers)

By increasing the number of simulations ($m = 500$) and the number of tosses, a "funnel" shape clearly emerges. Despite individual fluctuations, the mass of trajectories converges towards the line $y = 0.5$.

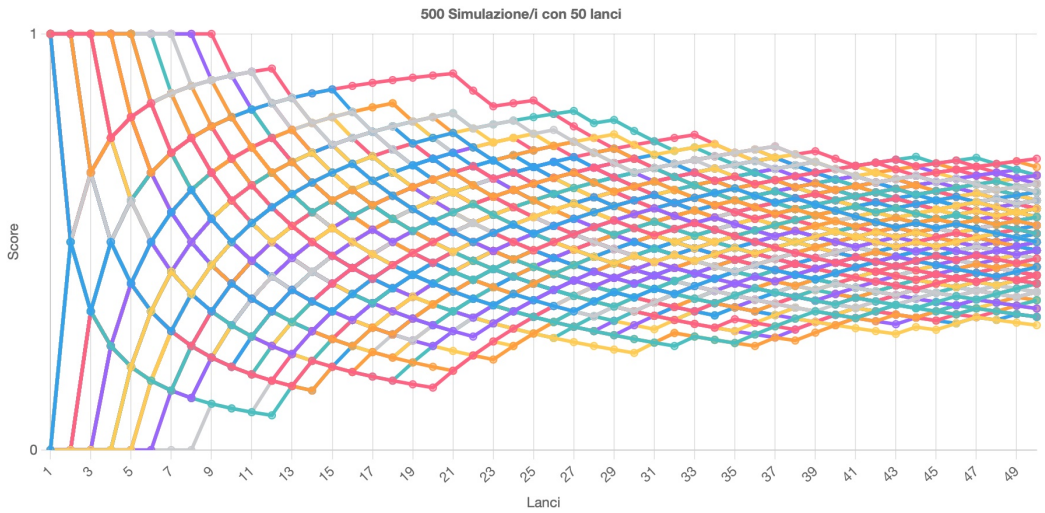

Figure 3.2. 500 simulations: visualization of the Law of Large Numbers.

3.2.3 Case 3: Bias (Biased Coin)

Modifying the code to simulate a coin with $p = 0.9$, convergence occurs nonetheless, but towards the new expected value. This demonstrates that the LLN is valid regardless of the value of p, provided the process is stationary.

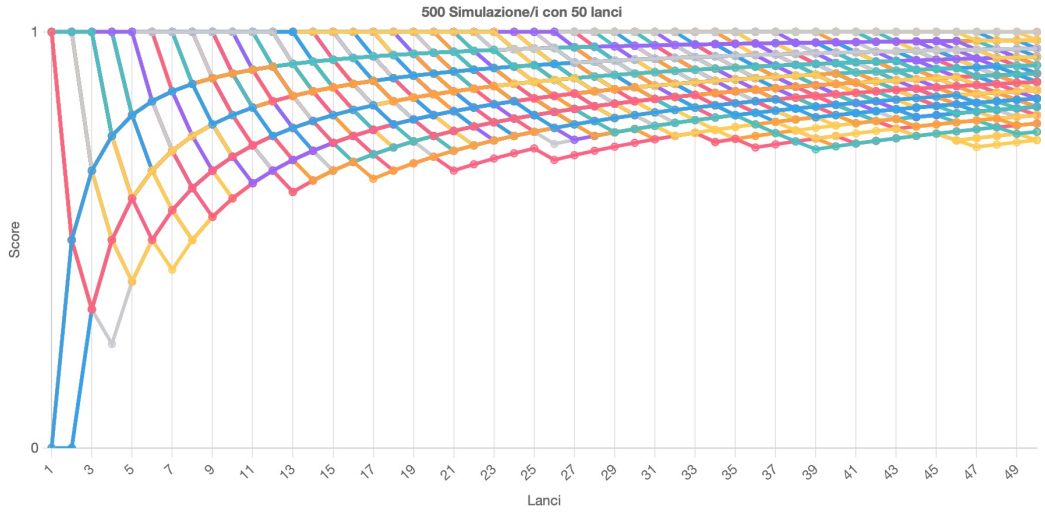

Figure 3.3. 500 simulations with biased coin ($p = 0.9$).

3.3 A Real-World Application: The Mathematics of Roulette

One of the most evident and commercially relevant applications of the Law of Large Numbers is found in the gambling industry. It is often said that "the house always wins". This statement is not the result of luck or secret manipulations, but is a direct consequence of the stochastic convergence guaranteed by the LLN.

3.3.1 Problem Modeling

Consider European Roulette, consisting of 37 numbers: from 0 to 36. A player betting on "Red" wins if one of the 18 red numbers comes up. If a black number (18) or zero (green) comes up, they lose.

We can model this bet using the developed simulator, treating it as a Bernoulli variable, but with a success parameter p slightly skewed against the player:

$$P(\text{Win}) = \frac{\text{Red Numbers}}{\text{Total Numbers}} = \frac{18}{37} \approx 0.4864 \quad (48.6\%)$$

$$P(\text{Loss}) = \frac{19}{37} \approx 0.5135 \quad (51.3\%)$$

3.3.2 Convergence Analysis and "House Edge"

The difference between a fair coin ($p = 0.5$) and roulette ($p \approx 0.486$) seems minimal (about 1.4%), but over large numbers, it is decisive.

- **Short Term (Small Numbers):** If a player places a few bets ($n = 10$ or $n = 50$), variance is high. As seen in the single simulation graph (Fig. 3.1), relative frequency can oscillate well above 0.5. The player may find themselves winning.

- **Long Term (Large Numbers):** The casino operates on millions of plays ($n \to \infty$). Due to the Strong Law of Large Numbers, the player's win frequency will *almost surely* converge to the theoretical value of 48.6%.

This systematic deviation from 50% is defined as the **House Edge**. Using our simulator by setting the probability to 48.6%, we can graphically observe how, as simulations increase, the score line inevitably stabilizes below the breakeven line, guaranteeing the casino a constant and predictable mathematical profit.

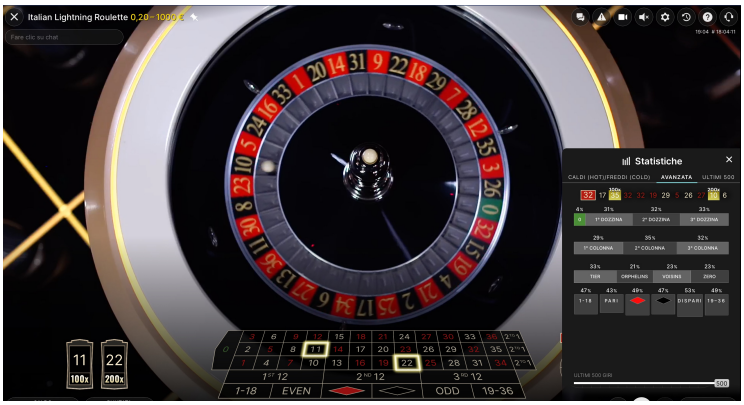

Figure 3.4. Roulette Layout.

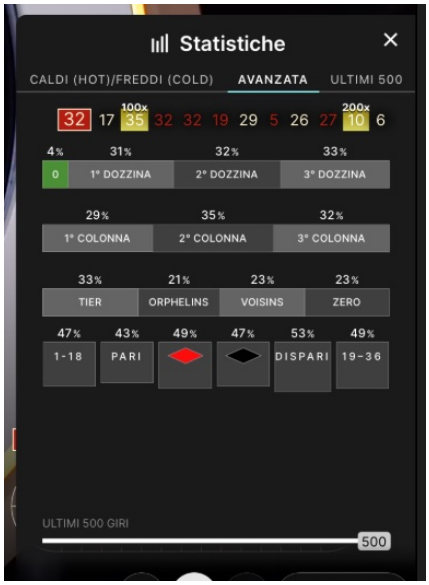

Figure 3.5. Statistical breakdown of Roulette outcomes.

Chapter 4

Applications in Data Analysis

The Law of Large Numbers is not just a mathematical abstraction, but the invisible engine that allows modern society to function. Without the guarantee that frequencies observed in a sample converge to real probabilities, entire industries such as insurance, finance, and digital marketing could not exist.

In this chapter, we will analyze how the LLN transforms the uncertainty of a single data point into a reliable prediction, through four fundamental practical examples.

4.1 Polls and Market Research

The main problem in social data analysis is the impossibility of interviewing the entire population (for example, all 60 million Italian citizens) to know an opinion or an electoral preference.

The analyst must rely on a *sample*. This is where the Law of Large Numbers comes into play:

- If we interview 10 people ($n = 10$), the frequency of opinions will be extremely volatile. A single "extreme" opinion can completely skew the average (like a series of 10 coin tosses yielding 8 heads).
- By increasing the sample to 1,000 or 2,000 people ($n \to \infty$), the LLN guarantees that the distribution of opinions in the sample faithfully reflects that of the entire population, with an increasingly reduced margin of error.

This principle allows predicting the outcome of national elections based on the responses of a few thousand individuals, saving enormous amounts of time and resources.

4.2 The Insurance Model and Risk Management

The insurance sector is perhaps the most direct application of the concept of convergence. Imagine a company insuring cars against accidents.

- **The Single Event (Uncertainty):** For the company, it is impossible to know if Mr. Smith will have an accident this year. It is a binary random event

(like a coin toss: 0 = no accident, 1 = accident). Insuring a single person is a gamble, not a business.

- **The Mass (Certainty):** If the company insures 100,000 drivers, the Law of Large Numbers kicks in. Even if we don't know *who* will have an accident, we know with extreme precision *how many* accidents will occur in total (frequency converges to the statistical probability of an accident).

Thanks to this statistical stability, the company can calculate the insurance premium to cover costs and generate profit. Without the LLN, the variance of costs would be unsustainable.

4.3 Industrial Quality Control

In production chains (e.g., microchips or pharmaceuticals), it is impossible to test every single product, as testing is often destructive or too expensive. A batch of samples is therefore taken from the production line.

- If out of 100 tested pieces, 2 are defective, the defect frequency is 2%.
- The LLN authorizes us to extend this result to the entire production of millions of pieces.

If the LLN were not valid, we would have no guarantee that the defect rate in the sample is representative of the total, making it impossible to guarantee high quality standards.

4.4 A/B Testing in Digital Marketing

In the digital world, decisions are not based on intuition but on data. A/B Testing is the standard technique for optimizing websites and apps. The experiment consists of showing Version A (e.g., blue button) to one group of users and Version B (e.g., red button) to another.

Let's use a numerical example linked to our simulations:

- **Scenario with few data ($n = 50$):** Version A gets 4 clicks, Version B gets 8. It seems B is twice as good (16% vs 8%). However, as seen in our 'chartOneSimulation' graph, with $n = 50$ oscillations are violent. This difference could be pure chance.
- **Scenario with many data ($n = 5,000$):** The LLN stabilizes frequencies. If A is now at 10% and B at 12%, we can confidently say that B is genuinely superior.

Those who ignore the Law of Large Numbers in A/B testing risk making business decisions based on statistical "noise" rather than real user behavior, leading to economic losses.

Chapter 5

Applications in Cybersecurity

In the context of modern cybersecurity, the main challenge is no longer the lack of information, but the overabundance of data (logs, network packets, system calls). The Law of Large Numbers (LLN) provides the mathematical foundation to filter this enormous information flow, allowing analysts and algorithms to distinguish what is "normal" (expected behavior converging to a stable mean) from what is "anomalous" (an attack or a malfunction).

In this chapter, we will delve into two pillars of operational security: anomaly-based intrusion detection systems and automated testing techniques (Fuzzing).

5.1 Anomaly Detection and Intrusion Detection Systems (IDS)

Intrusion detection systems fall into two main categories: *Signature-based* (looking for known attack fingerprints) and *Anomaly-based*. It is in the latter category that statistics and the LLN play a crucial role.

The basic assumption is that traffic on a healthy network follows a stationary distribution. According to the LLN, by monitoring the network for a sufficient time ($n \to \infty$), traffic metrics will converge towards a robust statistical "Baseline".

5.1.1 Baseline Construction and Statistical Thresholds

To detect a never-before-seen attack (Zero-Day), the IDS must first "learn" normality.

- **Training Phase (Large n):** The IDS analyzes millions of packets in the absence of attacks. It calculates the mean μ and variance σ^2 of various features (e.g., number of SYN requests per second, average packet size, session duration).
- **The Law of Large Numbers:** Guarantees that if the observation period is long enough, the sample mean calculated by the IDS is a faithful representation of the infrastructure's real behavior.

5.1.2 Attack Detection (Divergence)

A cyber attack alters the stochastic nature of traffic. Let's take a **DDoS (Distributed Denial of Service)** attack of the *HTTP Flood* type as an example.

1. Under normal conditions, HTTP requests arrive pseudo-randomly but with a stable average frequency μ_{norm}.

2. During the attack, thousands of bots send simultaneous requests.

3. The IDS calculates the moving average over a recent time window. Suddenly, the observed mean $\bar{X}_n$ diverges significantly from the baseline μ_{norm} established by the LLN.

4. If the deviation $|\bar{X}_n - \mu_{norm}|$ exceeds a critical threshold (usually defined as $k \cdot \sigma$), an alarm is triggered.

5.1.3 The False Positive Problem and Sample Size

A classic problem with IDS is the rate of false positives (unjustified alarms). This is directly explainable via the LLN:

- If the IDS makes decisions based on few packets (small n), variance is high (like in our simulation with 50 tosses). A normal spike in user traffic could be mistaken for an attack.

- By increasing the sampling window (large n), measurement becomes more precise and false positives decrease, but the system becomes slower to react. The engineering challenge lies in finding the right balance for n.

5.1.4 Case Study: Visualization of a DDoS Attack

To clarify the operation of a statistical IDS, we simulated a synthetic network traffic flow. As shown in Figure 5.1, the system learned a historical *Baseline* (green line) around 50 requests per second.

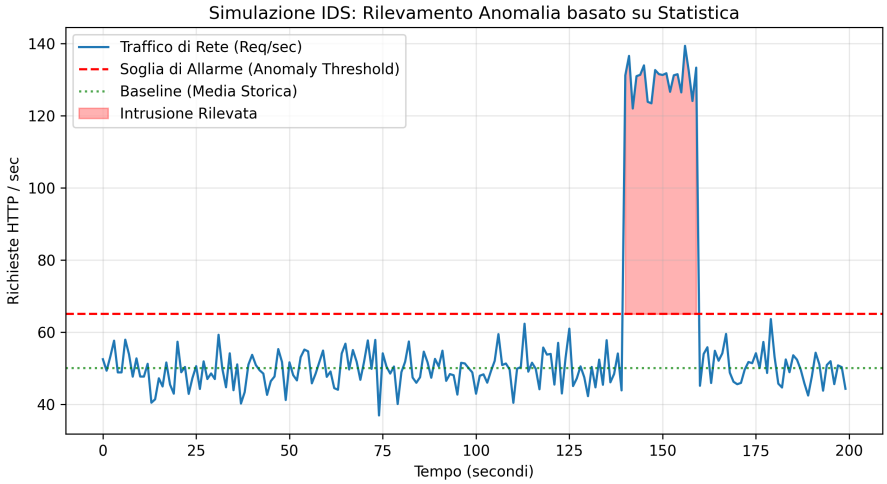

Figure 5.1. Detection of a statistical anomaly. The attack exceeds the confidence threshold established by historical variance.

Normal traffic physiologically oscillates around the mean (natural variance). However, at time $t = 140$, a simulated DDoS attack occurs. The local traffic mean instantly violates the *Alarm Threshold* (red dashed line), calculated as $\mu + 3\sigma$. This graph visually demonstrates how the Law of Large Numbers allows defining a mathematical boundary between "licit noise" and "illicit signal".

5.2 Fuzzing: The Brute Force of Large Numbers

Fuzzing (or Fuzz Testing) is a dynamic software analysis technique that involves inputting random, malformed, or unexpected data into a program to cause crashes or unexpected behaviors (memory leaks, buffer overflows).

While static code analysis looks for errors by reading the source code, Fuzzing relies on probability and computing power, leveraging the concept that "with enough attempts, every possible path will be explored".

5.2.1 State Space and Crash Probability

Complex software has a virtually infinite state space (set of all possible inputs and internal states). Critical bugs often reside in "Corner Cases", i.e., extremely rare input combinations that a human developer would hardly test.

We can model vulnerability discovery as a Bernoulli event with extremely low probability p (e.g., $p = 10^{-9}$).

- With few tests (small n), the probability of finding the bug is almost zero.
- The Law of Large Numbers tells us that the empirical crash frequency will converge to the real probability p.
- However, the most interesting aspect here is the complementary event. The probability of *not* finding the bug in n attempts is $(1 - p)^n$. Since $(1 - p) < 1$, as $n \to \infty$, this probability tends to 0.

In other words: if we run the fuzzer for a sufficiently long time (millions or billions of iterations), the mathematical certainty of finding the vulnerability (if it exists) tends to 100%.

5.2.2 Dumb Fuzzing vs. Smart Fuzzing

There are two main approaches, statistically interpretable:

- **Dumb Fuzzing (Black Box):** Generates purely random inputs. It relies exclusively on brute force and the LLN. It requires an enormous n to get results, as it explores the state space blindly.
- **Coverage-guided Fuzzing (Grey Box):** Modern tools like *AFL (American Fuzzy Lop)* use an evolutionary approach. When a random input discovers a new portion of code (increases Code Coverage), the tool uses it as a base for new mutations. Statistically, this equates to "loading the coin": the algorithm modifies the probability distribution of inputs to maximize the probability of

visiting rare states, drastically reducing the number n of attempts needed to find a crash compared to the purely random method.

In conclusion, Fuzzing is the industrial application of the "Infinite Monkey Theorem": thanks to the speed of modern processors, we can simulate the "infinity" needed for randomness to uncover human errors.

5.2.3 Practical Example: Fuzzing a Vulnerable Function

To demonstrate how randomness can explore complex code paths, consider the following Python example. Imagine a function `vulnerable_process` containing a critical bug nested in a specific sequence of conditions ("Magic Number").

A human might not notice the error in the code, but a *Dumb Fuzzer*, based on the Law of Large Numbers, will generate infinite inputs until, statistically, it stumbles upon the combination that satisfies the conditions.

```
import random
import string

def vulnerable_process(user_input):
    """
    Vulnerable function that crashes only if the input
    satisfies very specific conditions (Corner Case).
    """
    # Condition 1: Exact length
    if len(user_input) == 4:
        # Condition 2: First character 'F'
        if user_input[0] == 'F':
            # Condition 3: Last character 'Z'
            if user_input[3] == 'Z':
                # Condition 4: Internal calculation
                if user_input[1] == 'U' and user_input[2] == 'Z':
                    raise Exception("CRASH! Input 'FUZZ' broke the
                        system.")
    return "Input processed correctly."

# --- Fuzzer Simulation ---
print("Starting Fuzzing...")
attempts = 0

while True:
    attempts += 1
    # Generate a random string of 4 uppercase letters
    # Example: 'AAAA', 'XJKD', 'FUZZ'
    fuzz_input = ''.join(random.choices(string.ascii_uppercase, k=4))

    try:
        vulnerable_process(fuzz_input)
    except Exception as e:
        print(f"[-] Success! Bug found after {attempts} attempts.")
        print(f"[-] Guilty input: {fuzz_input}")
        break # Stop loop when bug is found

# Expected result:
# The loop will continue until the LLN guarantees the string 'FUZZ'
    appears.
```

Listing 5.1. Demonstrative Fuzzing Script

In this example, the state space is $26^4 = 456,976$ possible combinations. Although it seems large, for a modern computer generating them takes fractions of a second. The LLN assures us that, by keeping the loop active ($n \to \infty$), the probability of generating the string "FUZZ" tends to 1.

Chapter 6

Conclusions

In this paper, we analyzed the Law of Large Numbers starting from its mathematical definition up to its empirical verification through software simulation. We demonstrated that the concept of "convergence of frequency to probability" is the keystone for transforming the uncertainty of a single random event into statistical certainty over large volumes of data. This property is what makes Big Data analysis possible and enables the construction of robust cybersecurity systems capable of distinguishing between background noise and real threats, or ensuring the unpredictability necessary for secure cryptography.

Bibliography

[1] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, 1997.

[2] Sheldon M. Ross, *Introduction to Probability Models*, Academic Press, 2019.

[3] William Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson, 2016.