Master's Degree in Cybersecurity

Course: Statistics

RSA Encoding: Exploring Statistical Vulnerabilities and Cryptographic Weaknesses in Modern Security Systems

Student:

Alessandro Polidori

Professor:

Prof. Tommaso Gastaldi

Academic Year 2025-2026

Abstract

The RSA cryptosystem remains one of the cornerstones of modern digital security, securing data transmission through an elegant application of number theory and computational complexity. However, the security of RSA is not absolute; it relies heavily on the statistical improbability of factoring large composite integers and the correct implementation of probabilistic algorithms during key generation.

This thesis explores the statistical foundations of RSA, moving from the axiomatic definitions of modular arithmetic to the probabilistic nature of primality testing using the Miller-Rabin algorithm. We analyze the critical role of the Law of Large Numbers and probability bounds in ensuring key validity. Furthermore, we investigate statistical vulnerabilities arising from improper parameter selection and side-channel leakage. Specifically, we examine how statistical inference techniques can be applied to timing variations and power consumption to recover private keys, effectively bypassing the mathematical hardness of the factorization problem.

By bridging theoretical statistics with practical cryptanalysis, this work demonstrates that secure encryption is not merely a product of robust algorithms but also of statistically sound implementation and random number generation.

Contents

Chapter 1

Introduction

The digital era is defined by the transmission of information. As the volume of global data traffic grows exponentially, the requirement to secure this data against unauthorized access has transitioned from a military necessity to a fundamental pillar of the modern internet. At the heart of this security infrastructure lies the RSA (Rivest–Shamir–Adleman) cryptosystem, an algorithm that revolutionized cryptography by solving the key distribution problem.

However, the robustness of RSA is frequently misunderstood as a purely algebraic certainty. While its foundation rests on Number Theory, its practical security is intrinsically tied to **Statistics** and **Probability Theory**. The generation of keys relies on probabilistic primality tests, and the resistance to attacks often depends on the statistical distribution of prime factors and the entropy of random number generators. This thesis aims to deconstruct RSA not merely as a cryptographic standard, but as a statistical subject, analyzing its dependencies on randomness and the specific vulnerabilities that arise when statistical assumptions fail.

1.1 Historical Context: From Symmetry to Asymmetry

Prior to the 1970s, cryptography was almost exclusively **symmetric**. In a symmetric system, two parties, Alice and Bob, must share a single secret key K to communicate securely. If Alice encrypts a message M using an encryption function E such that $C = E_K(M)$, Bob must use the corresponding decryption function D with the same key, $M = D_K(C)$.

While computationally efficient, symmetric cryptography suffers from a critical logistical flaw known as the *Key Distribution Problem*: How can Alice and Bob agree

on the secret key K without an adversary, Eve, intercepting it? If they already have a secure channel to exchange the key, they would not need encryption in the first place.

In 1976, Whitfield Diffie and Martin Hellman published *"New Directions in Cryptography"*, proposing the concept of **Public-Key Cryptography** (Asymmetric Cryptography). They theorized a system involving two keys:

- A **Public Key** (K_{pub}), used for encryption, which can be disseminated widely.
- A **Private Key** (K_{priv}), used for decryption, known only to the recipient.

While Diffie and Hellman provided the theoretical framework, they did not immediately produce a practical implementation of a trapdoor one-way function necessary to realize the system. That breakthrough came a year later, in 1977, at the Massachusetts Institute of Technology (MIT), when Ron Rivest, Adi Shamir, and Leonard Adleman proposed the RSA algorithm.

1.2 The Statistical Nature of Computational Security

RSA derives its security from the difficulty of the *Integer Factorization Problem*. Specifically, it is easy to multiply two large prime numbers p and q to obtain a modulus $n = p \cdot q$, but it is computationally infeasible to recover p and q given only n, provided n is sufficiently large.

From a statistical perspective, RSA relies on the concept of **Computational Security** rather than Information-Theoretic Security.

Definition 1.1 (Computational Security)**.** *A cryptosystem is computationally secure if the best known algorithm for breaking it requires N operations, where N is such that the computation cannot be completed within a reasonable time frame using available computing power.*

This definition implies that security is not binary (secure vs. insecure) but probabilistic. It relies on the assumption that:

1. The probability of an adversary guessing the private key is negligible.
2. The probability of a random number generation algorithm producing predictable primes is negligible.

3. The statistical distribution of the ciphertext is indistinguishable from random noise (semantic security).

Throughout this thesis, we will explore how violations of these statistical requirements—such as a poor source of entropy during key generation or statistical biases in side-channel leakages—can render the mathematical hardness of RSA irrelevant.

1.3 Thesis Objectives and Structure

The primary objective of this thesis is to analyze the intersection of Statistics and Cryptography within the RSA framework. We will demonstrate that implementing RSA is not simply a matter of coding mathematical formulas, but requires rigorous statistical validation.

The work is organized as follows:

- **Chapter 2: Mathematical and Statistical Foundations.** We review the necessary Number Theory, including Modular Arithmetic and Euler's Totient Theorem, establishing the ground rules for RSA.
- **Chapter 3: Probabilistic Primality Testing.** We analyze the algorithms used to generate prime numbers. We focus on the Miller-Rabin test, discussing the Law of Large Numbers and error bounds to ensure that "probable primes" are sufficiently prime for security purposes.
- **Chapter 4: Statistical Vulnerabilities.** We investigate attacks that exploit statistical weaknesses, such as the factorization of moduli sharing prime factors and attacks on low-entropy keys.
- **Chapter 5: Side-Channel Attacks.** We explore how physical implementations leak statistical data (timing, power), allowing attackers to infer private keys using statistical correlation techniques.
- **Chapter 6: Conclusion.** We summarize the findings and briefly discuss the future impact of Quantum Computing on these statistical assumptions.

Chapter 2

Mathematical and Statistical Foundations

To understand the vulnerabilities of RSA, one must first master the mathematical rigidities that define it. RSA is constructed upon the ring of integers modulo n, denoted as $\mathbb{Z}_n$. While the cryptographic strength relies on the factorization problem, the operational feasibility of RSA relies on the statistical abundance of prime numbers.

This chapter outlines the fundamental theorems of modular arithmetic used in RSA and introduces the Prime Number Theorem, which provides the statistical justification for the efficient generation of large keys.

2.1 Modular Arithmetic and Euler's Totient Function

The core operations of RSA take place in the multiplicative group of integers modulo n, specifically focusing on elements that are coprime to n.

Definition 2.1 (Euler's Totient Function)**.** *For a positive integer n, Euler's totient function, denoted as $\phi(n)$, counts the number of positive integers less than or equal to n that are relatively prime to n.*

The calculation of $\phi(n)$ is trivial if the prime factorization of n is known, but computationally hard if it is not. This asymmetry is the heart of RSA.

- If p is a prime number, then every integer from 1 to $p-1$ is coprime to p. Thus:

$$\phi(p) = p - 1 \tag{2.1}$$

- Since the function is multiplicative, if $n = p \cdot q$ where p and q are distinct primes:

$$\phi(n) = \phi(p) \cdot \phi(q) = (p-1)(q-1) \tag{2.2}$$

The correctness of the RSA algorithm is guaranteed by Euler's Theorem, a generalization of Fermat's Little Theorem.

Theorem 2.1 (Euler's Theorem)**.** *If a and n are coprime positive integers (i.e., $\gcd(a, n) = 1$), then:*

$$a^{\phi(n)} \equiv 1 \pmod{n} \tag{2.3}$$

This theorem allows for the "inversion" of the encryption process. If a message M is raised to a power e, we can recover M by raising the result to a power d, provided $e \cdot d \equiv 1 \pmod{\phi(n)}$.

2.2 The RSA Algorithm Structure

The RSA cryptosystem consists of three distinct phases: Key Generation, Encryption, and Decryption. From a statistical perspective, the Key Generation phase is the most critical, as it involves random sampling from the search space of prime numbers.

2.2.1 Key Generation Algorithm

1. **Choose two distinct large prime numbers** p and q. These are chosen at random and typically verified using probabilistic primality tests (discussed in Chapter 3).

2. **Compute the modulus** $n = p \cdot q$. The length of n in bits represents the key length (e.g., 2048 bits).

3. **Compute the totient** $\phi(n) = (p-1)(q-1)$.

4. **Choose an integer** e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$. This is the *public exponent*. A common choice is $e = 65537$ $(2^{16} + 1)$ due to its efficiency in binary exponentiation.

5. **Compute** d as the modular multiplicative inverse of e modulo $\phi(n)$:

$$d \equiv e^{-1} \pmod{\phi(n)} \tag{2.4}$$

This is the *private exponent*, calculated using the Extended Euclidean Algorithm.

The **Public Key** is (n, e), and the **Private Key** is (d, p, q).

2.2.2 Encryption and Decryption

Given a message M (represented as an integer $0 \leq M < n$):

- **Encryption:** $C \equiv M^e \pmod{n}$
- **Decryption:** $M \equiv C^d \pmod{n}$

2.3 The Statistical Distribution of Prime Numbers

A fundamental question in the implementation of RSA is: *How hard is it to find large prime numbers?* If primes were extremely rare, the Key Generation phase (Step 1) would be computationally prohibitive, rendering RSA impractical.

The answer lies in the **Prime Number Theorem (PNT)**, which describes the asymptotic distribution of primes.

Definition 2.2 (Prime Counting Function)**.** *Let $\pi(x)$ be the prime-counting function, defined as the number of prime numbers less than or equal to a real number x.*

Theorem 2.2 (Prime Number Theorem)**.** *As x approaches infinity, the prime-counting function $\pi(x)$ is asymptotic to $x/\ln x$:*

$$\lim_{x \to \infty} \frac{\pi(x)}{x/\ln x} = 1 \tag{2.5}$$

2.3.1 Statistical Implications for RSA

The Prime Number Theorem provides a probability model for finding primes. For a randomly chosen large integer x, the probability $P(x)$ that x is prime is approximately:

$$P(x \text{ is prime}) \approx \frac{1}{\ln x} \tag{2.6}$$

This statistical property is crucial for cybersecurity. Consider generating a 1024-bit prime (required for a 2048-bit RSA key). The value of such a number is approximately $x \approx 2^{1024}$. The probability that a random 1024-bit odd integer is prime is

roughly:

$$\frac{2}{\ln(2^{1024})} = \frac{2}{1024 \cdot \ln 2} \approx \frac{2}{1024 \cdot 0.693} \approx \frac{1}{355} \tag{2.7}$$

(We multiply by 2 because we only test odd integers).

This result is statistically significant: it implies that we do not need to search through billions of numbers. On average, we only need to test about 355 random odd integers to find a prime of this magnitude. This manageable statistical density enables the practical creation of RSA keys on standard hardware. However, it also implies that the "randomness" of the selection process is paramount; any statistical bias in the Random Number Generator (RNG) drastically reduces the search space for an attacker.

Chapter 3

Probabilistic Primality Testing

As established in the previous chapter, the security of RSA relies on the generation of large prime numbers. However, definitively proving that a number $n \approx 2^{2048}$ is prime using deterministic methods (such as Trial Division or the AKS primality test) is computationally expensive and, for practical purposes, too slow for real-time key generation.

Instead, modern cryptography relies on **Probabilistic Primality Tests**. These algorithms do not output a binary "Prime" or "Composite" truth; rather, they output "Composite" or "Probably Prime". This chapter analyzes the statistical mechanisms of these tests, focusing on the Miller-Rabin algorithm, and quantifies the probability of error—the risk that a composite number is statistically mistaken for a prime.

3.1 The Statistical Flaw of Fermat's Test

The earliest probabilistic tests were based on Fermat's Little Theorem.

Theorem 3.1 (Fermat's Little Theorem)**.** *If n is prime, then for any integer a such that $1 \leq a < n$:*

$$a^{n-1} \equiv 1 \pmod{n} \tag{3.1}$$

The *Fermat Primality Test* operates by picking a random base a and checking if the congruence holds. If $a^{n-1} \not\equiv 1 \pmod{n}$, n is certainly composite. However, if the congruence holds, n is only *likely* to be prime.

3.1.1 Carmichael Numbers and Statistical Failure

From a statistical viewpoint, a composite number n that satisfies Fermat's theorem is a **False Positive**, known as a *Fermat Pseudoprime*. A major statistical vul-

nerability in this approach is the existence of **Carmichael Numbers**. These are composite numbers n that satisfy $a^{n-1} \equiv 1 \pmod{n}$ for *all* bases a coprime to n. For example, the number $561 = 3 \cdot 11 \cdot 17$ is a Carmichael number. Against such numbers, the Fermat test fails to detect compositeness regardless of how many random bases a are tested (unless a happens to share a factor with n). This represents a structural statistical blind spot, necessitating a more robust test.

3.2 The Miller-Rabin Primality Test

The Miller-Rabin test improves upon Fermat's test by looking for non-trivial square roots of unity. It relies on the property that for a prime n, the equation $x^2 \equiv 1 \pmod{n}$ has only two solutions: $x \equiv 1$ and $x \equiv -1$ (which is $n-1$).

3.2.1 The Algorithm

Given an odd integer n to be tested:

1. Write $n-1$ as $2^s \cdot d$, where d is odd. This extracts the powers of 2 from $n-1$.
2. Choose a random base a uniformly from the range $[2, n-2]$.
3. Compute $x_0 = a^d \pmod{n}$.
4. If $x_0 \equiv 1$ or $x_0 \equiv n-1$, then n passes the test for base a.
5. Otherwise, repeat the squaring operation $s-1$ times:

$$x_i = x_{i-1}^2 \pmod{n} \tag{3.2}$$

6. If $x_i \equiv n-1$, then n passes. If the loop finishes without finding $n-1$, then n is composite.

If n passes the test, it is a *Strong Pseudoprime* to base a.

3.3 Statistical Analysis of Error Probability

The crucial aspect for a cybersecurity statistician is quantifying the confidence level of the generated keys. How likely is it that RSA generates a composite number by mistake?

Let n be an odd composite number. We define a "witness" as a base a that correctly reveals n to be composite. The strength of the Miller-Rabin test lies in the lower bound of the density of these witnesses.

Theorem 3.2 (Monier-Rabin Bound)**.** *If n is an odd composite number, then the number of bases $a \in [1, n-1]$ that are witnesses to the compositeness of n is at least $\frac{3}{4}(n-1)$.*

This theorem has profound statistical implications. It implies that for any composite n, at most $\frac{1}{4}$ of the possible bases are "liars" (bases that result in a False Positive).

3.3.1 Independent Events and Confidence Accumulation

Since the algorithm chooses the base a randomly and independently in each iteration, we can model the test as a sequence of Bernoulli trials. Let E be the event that a composite number n is declared "probably prime". If we perform the test k times with k independently chosen random bases:

$$P(E) \leq \left(\frac{1}{4}\right)^k = 4^{-k} \tag{3.3}$$

This formula demonstrates that the probability of error decreases exponentially with the number of rounds k.

3.3.2 Practical Application and Risk Assessment

In practical implementations (such as OpenSSL), k is typically set to 40. Let us calculate the probability of error for $k = 40$:

$$P(\text{Error}) \leq 4^{-40} = 2^{-80} \approx 10^{-24} \tag{3.4}$$

To put this statistical risk into perspective: the probability of the computer hardware failing due to a cosmic ray flipping a bit during the calculation is significantly higher than the probability of the Miller-Rabin test failing after 40 rounds.

Therefore, while the test is not deterministic in the strict mathematical sense, it is **statistically deterministic**. The Law of Large Numbers ensures that as we increase the sample size (number of rounds), the observed result converges to the truth with a certainty that surpasses most physical engineering constraints.

3.4 The Generation of Primes in Practice

Combining the Prime Number Theorem (Chapter 2) and the Miller-Rabin error bounds, the full algorithm for generating an RSA prime p is a statistical process:

1. **Random Sampling:** Generate a random odd n-bit integer x.
2. **Filtering:** Apply trial division with small primes (e.g., up to 2000) to quickly discard obvious composites. This is a statistical filter to improve average-case performance.
3. **Probabilistic Testing:** Run k iterations of Miller-Rabin.
4. **Decision:** If x passes all k tests, accept x as prime. Else, go to Step 1.

This process highlights that modern cryptography is not the search for absolute truth, but the management of statistical risk.

Chapter 4

Statistical Vulnerabilities and Factorization Attacks

The security of RSA is predicated on the assumption that the Integer Factorization Problem is intractable for large numbers. Specifically, given a modulus $n = p \cdot q$, there is no efficient algorithm to recover p and q in polynomial time.

However, cryptanalysis often bypasses the brute-force mathematical problem by exploiting statistical anomalies in how the system is implemented. If the parameters p, q, e, or d are chosen from statistical distributions that lack sufficient entropy or uniformity, the complexity of breaking RSA collapses. This chapter analyzes two major categories of statistical failures: the correlation of prime factors across large datasets (Batch GCD attack) and the improper selection of private exponents (Wiener's attack).

4.1 The Entropy Failure: Batch GCD Attack

In a statistically ideal RSA implementation, the prime numbers p and q are chosen uniformly at random from the set of all available primes of a given bit length. Let S be the space of 1024-bit primes. The size of this space is approximately $|S| \approx 10^{300}$. The probability of two independent users generating the same prime number p is:

$$P(\text{Collision}) \approx \frac{1}{|S|} \approx 10^{-300} \tag{4.1}$$

This probability is negligible. Therefore, under ideal statistical conditions, two different public moduli n_1 and n_2 should be coprime, i.e., $\gcd(n_1, n_2) = 1$.

4.1.1 The Statistical Anomaly

In 2012, researchers performed a massive statistical analysis of the Internet, collecting millions of public keys (X.509 certificates and SSH keys). They discovered that a significant percentage of these keys (approx. 0.2% to 0.5% of all keys) could be broken instantly.

The vulnerability was not in the RSA math, but in the **Random Number Generators (RNGs)** used by embedded devices (routers, firewalls, IoT devices). Due to low entropy at startup, these devices were generating prime numbers that were not uniformly distributed.

4.1.2 The Attack Mechanism

If two distinct moduli $n_A = p \cdot q_A$ and $n_B = p \cdot q_B$ share a common prime factor p (due to a statistical bias in the RNG), an attacker can recover p efficiently using the Euclidean Algorithm for the Greatest Common Divisor (GCD):

$$p = \gcd(n_A, n_B) \tag{4.2}$$

The computation of $\gcd(n_A, n_B)$ is extremely fast (logarithmic time), unlike factorization.

4.1.3 Statistical Significance

This attack demonstrates a critical lesson in statistical cybersecurity: **security assumes independence**. If we view the global population of RSA keys as a statistical dataset $\mathcal{N} = \{n_1, n_2, \ldots, n_k\}$, the naïve assumption is that all n_i are independent variables. The "Batch GCD" attack proved that in the real world, these variables are correlated due to poor entropy sources. By computing the GCD of pairs in a dataset of millions of keys, an attacker can compromise thousands of users who thought their keys were mathematically secure, simply because their random sampling process was statistically flawed.

4.2 Distributional Weakness: Wiener's Attack

Another statistical vulnerability arises from the selection of the private exponent d. In standard RSA, the public exponent e is often small (e.g., 65537). Consequently, the private exponent d is typically of the same order of magnitude as the modulus n ($d \approx n$).

However, for performance reasons (e.g., on smart cards with limited processing power), implementers might be tempted to choose a small d to speed up the decryption process $C^d \pmod{n}$. Michael Wiener proved that if d is chosen from the lower bound of the statistical distribution of possible values, the key is insecure.

4.2.1 The Theorem

Theorem 4.1 (Wiener's Theorem)**.** *Let $n = p \cdot q$ with $q < p < 2q$. If the private exponent d satisfies:*

$$d < \frac{1}{3} n^{1/4} \tag{4.3}$$

then d can be efficiently recovered from (n, e) using continued fractions.

4.2.2 Statistical Implication

The equation $e \cdot d \equiv 1 \pmod{\phi(n)}$ can be rewritten as:

$$e \cdot d - k \cdot \phi(n) = 1 \tag{4.4}$$

for some integer k. Dividing by $d \cdot \phi(n)$, we obtain:

$$\left| \frac{e}{\phi(n)} - \frac{k}{d} \right| < \frac{1}{d \cdot \phi(n)} \tag{4.5}$$

Since we do not know $\phi(n)$, we can approximate it with n. Wiener showed that the fraction $\frac{k}{d}$ appears in the *continued fraction expansion* of $\frac{e}{n}$.

From a statistical design perspective, this creates a "forbidden zone" in the distribution of d. If the random selection of d is not uniform over the entire range $[1, \phi(n)]$ but is heavily skewed towards the lower end (for efficiency), the system falls into this deterministic trap. This reinforces the principle that cryptographic parameters must be chosen from the entire available sample space to maximize entropy and resistance to analysis.

4.3 The General Number Field Sieve (GNFS)

Finally, we must acknowledge the primary tool used to gauge the strength of RSA against a standard statistical attack: the General Number Field Sieve. The GNFS is currently the most efficient classical algorithm for factoring large integers. Its time

complexity is sub-exponential:

$$O\left(\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln n)^{1/3}(\ln\ln n)^{2/3}\right)\right) \tag{4.6}$$

While this complexity grows rapidly, it is not exponential. The security of RSA depends on the gap between the speed of GNFS and the bit-length of n. This is a statistical arms race: as computing power increases and the heuristics of GNFS improve, the required bit-length for n must increase (from 1024 to 2048, and now 4096 bits) to maintain the probability of successful factorization below a negligible threshold.

Chapter 5

Side-Channel Attacks and Statistical Inference

In the previous chapters, we treated RSA as an abstract mathematical function $C = M^e \pmod{n}$. In reality, RSA is a physical process executed on silicon logic gates that consume time and energy.

A Side-Channel Attack (SCA) ignores the theoretical strength of the algorithm and focuses on the physical leakage of information. From a statistical perspective, an SCA is a problem of **Signal Detection Theory**: the private key is the hidden signal, and the physical measurements (time, power, sound) are noisy data points. The attacker uses statistical inference to isolate the signal from the noise.

5.1 Timing Attacks and Variance Analysis

The first major statistical side-channel attack was introduced by Paul Kocher in 1996. It targets the implementation of the modular exponentiation algorithm.

5.1.1 The Square-and-Multiply Algorithm

To compute $y = x^d \pmod{n}$ efficiently, systems use the binary expansion of the exponent d. Let the binary representation of the private key be $d = (d_k, d_{k-1}, \ldots, d_0)_2$. The algorithm proceeds as follows:

```
R = 1;
for i = k down to 0:
    R = (R * R) % n;       // Square (Always executed)
    if (d_i == 1):
        R = (R * x) % n;   // Multiply (Conditional)
```

5.1.2 Statistical Correlation

The "Multiply" step is executed only if the key bit d_i is 1. This introduces a statistical dependency between the value of the key bit and the total execution time T. Let T be the total time variable. We can model it as:

$$T = \sum_{i=0}^{k} (t_{square} + d_i \cdot t_{multiply}) + \epsilon \tag{5.1}$$

where ϵ represents measurement noise (variance from OS scheduling, thermal noise, etc.).

An attacker measures the execution time T_j for N different known messages M_j. By analyzing the variance and correlation of these times, the attacker can guess the bits d_i sequentially. If the attacker guesses a bit d_i correctly, the variance of the timing residuals (the time remaining after accounting for the guessed operations) decreases. If the guess is wrong, the variance increases (random noise). This is a classic application of **Variance Reduction** techniques.

5.2 Differential Power Analysis (DPA)

While timing attacks rely on aggregate data, Differential Power Analysis (DPA) is a more powerful statistical technique that uses hypothesis testing on power consumption traces.

5.2.1 The Statistical Model

When a transistor in a CPU switches state (from 0 to 1 or 1 to 0), it consumes a small amount of power. The attacker captures a set of N power traces $T_1, \ldots, T_N$ corresponding to the encryption of N different data blocks $D_1, \ldots, D_N$.

The attacker defines a **Selection Function** $D(C, b)$ which guesses the value of a specific intermediate bit b computed during the algorithm, based on a guess of a sub-segment of the key. This divides the dataset into two subpopulations:

- Set S_0: Traces where the target bit is predicted to be 0.
- Set S_1: Traces where the target bit is predicted to be 1.

5.2.2 Difference of Means Test

The attacker computes the average power trace for both sets and calculates the differential trace Δ:

$$\Delta[j] = \frac{1}{|S_1|} \sum_{i \in S_1} T_i[j] - \frac{1}{|S_0|} \sum_{i \in S_0} T_i[j] \tag{5.2}$$

where j represents the time sample.

Statistical Inference:

- **Null Hypothesis (H_0):** The key guess is wrong. The selection function D is uncorrelated with the actual computation. The partition S_0, S_1 is random regarding the power consumption. Thus, $\lim_{N \to \infty} \Delta[j] \approx 0$.
- **Alternative Hypothesis (H_1):** The key guess is correct. The partition reflects the actual bit value processed by the device. At the specific time instance j^* where that bit is processed, there will be a statistically significant "spike" in the differential trace $\Delta[j^*]$.

This method is extremely robust because, by the Law of Large Numbers, the noise (uncorrelated data) averages out to zero as N increases, while the signal (the correct key correlation) remains constant. An attacker can extract a key even if the noise is much louder than the signal, simply by increasing the sample size N.

5.3 Statistical Countermeasures: Blinding

To defend against these statistical inference attacks, we must break the correlation between the input data and the side-channel leakage. This is achieved through a technique called **Blinding**, which effectively introduces artificial statistical noise.

Before decrypting a ciphertext C, the system multiplies it by a random factor r^e:

$$C' = C \cdot r^e \pmod{n} \tag{5.3}$$

The system then decrypts C':

$$M' = (C')^d = (C \cdot r^e)^d = C^d \cdot r^{ed} = M \cdot r \pmod{n} \tag{5.4}$$

Finally, the system multiplies by r^{-1} to recover M.

From the attacker's statistical viewpoint, the device is now processing a random number C' rather than the target C. The power consumption and timing depend

on r, which is random and unknown. This destroys the correlation required for DPA, rendering the difference of means test statistically insignificant (the spike disappears).

Chapter 6

Conclusion and Future Outlook

This thesis has explored the RSA cryptosystem not merely as a set of algebraic rules, but as a complex statistical ecosystem. We have demonstrated that while the mathematical foundation of RSA—modular arithmetic and Euler's Theorem—is elegant and robust, the operational security of the system is entirely dependent on statistical implementation details.

6.1 Summary of Statistical Vulnerabilities

Our analysis has highlighted three critical areas where statistics dictate security:

1. **Key Generation and Primality:** The transition from deterministic certainty to probabilistic assurance is fundamental. The Miller-Rabin test relies on the Law of Large Numbers to reduce the probability of error to negligible levels (4^{-k}). We concluded that "security" is effectively a measure of statistical confidence, not absolute proof.

2. **Entropy and Independence:** The "Batch GCD" attack illustrated the catastrophic consequences of statistical correlation. The assumption that generated primes are independent variables is the single most critical point of failure in real-world deployments. When Random Number Generators (RNGs) exhibit bias, the search space for an attacker collapses, bypassing the factorization problem entirely.

3. **Side-Channel Inference:** We examined how physical leakages (time and power) transform cryptanalysis into a statistical inference problem. Attacks like Differential Power Analysis (DPA) utilize hypothesis testing and difference-of-means to extract keys from noise, proving that a mathematically secure algorithm can be statistically insecure in hardware.

The overarching conclusion is that a cryptographer must essentially be a statistician. Designing secure systems requires a rigorous understanding of probability distributions, variance analysis, and entropy sources.

6.2 The Quantum Threat: Shor's Algorithm

Looking to the future, the statistical assumptions underpinning RSA face an existential threat from Quantum Computing. Current security relies on the time complexity of the best classical factorization algorithm (GNFS) being sub-exponential.

However, in 1994, Peter Shor formulated a quantum algorithm that can factor an integer n in polynomial time:

$$O((\log n)^2(\log\log n)(\log\log\log n)) \tag{6.1}$$

From a statistical perspective, Shor's algorithm utilizes the *Quantum Fourier Transform* to detect the period of the function $f(x) = a^x \pmod n$ with high probability. Where classical algorithms search for factors in a vast probability space, quantum algorithms exploit quantum interference to amplify the probability of the correct answer (the period) while canceling out wrong answers.

If a sufficiently large quantum computer is built, the "statistical hardness" of RSA will vanish. The problem will shift from being computationally infeasible to trivially solvable. This impending reality is driving the field towards **Post-Quantum Cryptography (PQC)**, involving lattice-based and hash-based systems that rely on different statistical problems not susceptible to quantum period finding.

6.3 Final Remarks

RSA remains the gold standard for secure communication today, provided it is implemented with rigorous statistical validation—specifically, high-entropy RNGs, appropriate parameter selection (avoiding small d), and blinding countermeasures against side-channel analysis. However, as computational power evolves and new statistical attack vectors emerge, the reliance on the difficulty of factorization serves as a reminder that in cybersecurity, probability is the only reality.