Joe L. Rivas

Inst: Mehran Hassanpour, Ph.D.

Course: MATH 2414 P01

12 MAY-2025

Diverging the Partial Sums of Two Men of Calculus: An Encryptic Frontier

Abstract:

This paper presents a mathematical framework rooted in infinite series to analyze encrypted data streams and predict decryption complexity in cybersecurity systems. Drawing on tools from Calculus II, including geometric and logarithmic series, convergence and divergence tests, recurrence relations, and integral approximations; the study constructs predictive models that simulate brute-force attacks, measure entropy decay, and identify threshold-based defense triggers. These models are then used to design proactive, adaptive network defenses that can respond in real-time to intruder behavior. Beyond the technical dimension, this work is grounded in a human imperative: to protect the cognitively disabled, the socially isolated, the economically marginalized—those often overlooked by security architectures designed for the able and informed. In weaving analytical precision with ethical intent, this research honors the converging narratives of two lives; an Iranian professor shaped by revolution and exile, and a Hispanic student shaped by poverty and perseverance, who meet in the shared language of calculus, committed to safeguarding others through the mathematics of foresight, recursion, and hope.

I. Introduction:

In modern digital infrastructures, the mathematics of protection is no longer abstract—it is imperative. Passwords, encrypted data streams, and dynamic authentication systems are often the final safeguards between personal security and exploitation. For people with cognitive disabilities, retired middle-class users, isolated youth, and adults with long-term disabilities, these mechanisms are not merely optional technicalities; they are essential protections for autonomy, privacy, and digital participation. In response to escalating threats—from brute-force attacks to real-time credential phishing—security analysts must adopt mathematical models that offer both predictive accuracy and operational foresight. This paper proposes a framework grounded in infinite series theory to model decryption complexity, simulate intruder behavior, and engineer network defenses that remain steps ahead of potential threats. A central tool in this pursuit is the geometric series, defined as:

$$\sum_{n=1}^{\infty} ar^{n-1} = a + ar + ar^2 + ar^3 + \ ...$$

which converges when |r|<1, yielding:

$$\sum_{n=1}^{\infty} ar^{n-1} = \frac{a}{1-r}$$

This formula, while traditionally taught in theoretical contexts, has direct application in cybersecurity. Consider a password composed from an alphanumeric set (k characters), of variable length up to n. The number of total permutations (assuming repetition) is:

$$P(n) = \sum_{i=1}^{\infty} k^i = k + k^2 + \cdots + k^n$$
$$= \frac{k(k^n - 1)}{k-1}$$

This finite geometric series is used to estimate the keyspace entropy—a measure of unpredictability in password systems. For theoretical modeling, we may examine the limit as n→∞, observing divergence unless |k|<1, which never applies in practice. However, this mathematical tension offers insight: decryption becomes computationally infeasible not when the series converges, but when it diverges beyond the bounds of practical computation. Now, consider an attacker iterating through the keyspace with a fixed algorithm, each attempt represented by a time-dependent loop. We can describe the cumulative attack time, T(n) as:

$$T(n) = c\sum_{i=1}^{\infty} k^i = c \cdot \frac{k(k^n - 1)}{k - 1}$$

Where c is the average time per guess. The rapid exponential growth in T(n) illustrates the effectiveness of even modest increases in password length or complexity. For defenders, this model suggests that even a single character increase in length multiplies attack time exponentially, reinforcing the strategic necessity of complexity enforcement. Yet brute-force attacks are not the only concern. More sophisticated threats leverage entropy analysis, and for this, logarithmic series offer another layer of understanding. Recall the Maclaurin series for the natural logarithm:

$$ln(1 - x) = \sum_{n=1}^{\infty} (-1)^{n+1}\frac{x^n}{n}, -1 < x \leq 1$$

This expansion helps approximate entropy values derived from probability distributions:

$$H(X) = -\sum_{i=1}^{\infty} p_i log_2 \, p_i$$

By substituting log terms with their series expansions, defenders can compute entropy thresholds or information leakage estimations with greater precision. This becomes vital in real-time scenarios, where adaptive authentication systems must evaluate whether an observed login pattern is statistically probable or potentially malicious. Finally, the ethical dimension cannot be abstracted away. For the groups most at risk—those whose cognitive or physical conditions prevent fast reaction or technical comprehension, the failure of a mathematical model can mean the failure of a life support system, a compromised retirement account, or the exposure of private health data. Thus, the calculus of decryption is not merely academic, it is humanitarian. By employing infinite series as both a predictive and defensive apparatus, security analysts can build systems that not only repel intrusion but also prioritize the protection of those least able to defend themselves. This paper explores these models, proving that convergence is not just a mathematical result, it is a moral one.

II. Background:

In cybersecurity, the complexity of encryption and the feasibility of decryption are central to system resilience. To mathematically model these processes, we draw upon foundational topics in Calculus II—particularly infinite series, geometric growth, convergence criteria, and logarithmic expansions. This section introduces these tools and frames their relevance to encrypted data systems, brute-force decryption models, and entropy estimation.

1. Infinite Series and Geometric Growth

An infinite series is the summation of an infinite sequence of terms. In practical security modeling, infinite series help us analyze patterns that grow without bound—such as the total number of attempts an intruder might make to guess a password or break an encryption key.

The theoretical backbone of decryption complexity lies in the infinite geometric series:

$$\sum_{n=1}^{\infty} ar^{n-1} = \frac{a}{1-r}, for\ |r| < 1$$

While divergence ($|r| \geq 1$)typically signals computational impracticality, it also plays a constructive role in cybersecurity. For example, using a character set size of $r =$

62 andse guess time $c = 10^{-6}$ seconds, the total time for a brute-force attack can be modeled by:

$$T(n) = c\sum_{i=1}^{\infty} k^i = c \cdot \frac{k(k^n - 1)}{k - 1}$$

The explosive growth rate of r^n shows that even small increases in password length create decryption costs that quickly exceed the lifespan of computing hardware. Thus, we use divergence deliberately—*as a design goal* in strong encryption.

2. Comparison and Limit Tests in Threat Estimation

The Comparison Test provides a formal way to assess how quickly a series of attacks (e.g., probabilistic password guesses) grows relative to a benchmark series. If:

$$0 \leq a_n \leq b_n \; and \; \sum b_n \; converges$$

then $\sum a_n$ also converges. In network defense, if we let $a_n = \frac{1}{n^2}$ represent low-frequency threat attempts (e.g., login failures over time) and compare it to a known convergent p-series (p =2), we confirm that such events are containable. Alternatively, we use the Limit Comparison Test when direct comparison fails:

$$\lim_{n\to\infty} \frac{a_n}{b_n} = L \; where \; 0 < L < \infty$$

This helps assess whether a stochastic threat model matches the long-term behavior of a modeled intrusion pattern.

III. Modeling Decryption Complexity

While encryption algorithms are often treated as computationally secure due to their vast keyspaces and mathematical structure, this assumption only holds when we rigorously model and understand the complexity of decryption from the perspective of an attacker. Using tools from Calculus II—particularly infinite series, exponential growth models, and summation loops—we can build predictive frameworks to simulate brute-force attempts, quantify attack timelines, and identify critical intervention thresholds.

1. Password Complexity and Series Growth

The most direct way to visualize decryption complexity is through the size of the password space. Suppose a password is constructed using a character set of size k (e.g., alphanumeric, k=62) and the password length varies from 1 to n. Then the total number of unique passwords, allowing repetition, is:

$$P(n) \quad = \sum_{i=1}^{n} k^i = k + k^2 + k^3 + \cdots + k^n = \frac{k(k^n - 1)}{k - 1}$$

This is a finite geometric series. However, for modeling an attacker with unlimited time and compute power, we examine the limit as n→∞:

$$\sum_{i=1}^{\infty} k^i \to \infty$$

The divergence of this series reinforces a core security principle: when keyspaces grow exponentially, brute-force decryption becomes infeasible. But for a persistent intruder, feasibility is not binary, it's probabilistic, and that's where modeling matters.

2. Time-Based Attack Modeling

Assume the attacker requires an average time c to evaluate each password. Then the cumulative time to exhaustively search the space up to length n is:

$$T(n) = c \sum_{i=1}^{\infty} k^i = c \cdot \frac{k(k^n - 1)}{k - 1}$$

For example, let k = 62, $n = 8, and\ c = \ 10^{-6}\ seconds$:

$$T(8) = 10^{-6} \cdot \frac{62(62^8 - 1)}{61} \approx 218{,}000\, yrs$$

This model quantifies the deterrence value of increasing password length—demonstrating how the exponential rate of growth in k^n makes even small changes dramatically raise attacker cost.

3. Series Acceleration and Parallel Intrusions

In high-threat environments, attackers may implement parallelized attacks using multiple threads or distributed systems (botnets). In this case, total attack time is reduced by a factor m, where m is the number of nodes or processors:

$$T_{parallel}(n) = \frac{T(n)}{m}$$

While this increases feasibility for shorter passwords, the series growth remains exponential, and the time cost of longer passwords dominates any linear scaling of attacker infrastructure.

4. Entropy Decay and Logarithmic Series

If attackers are not guessing randomly, but using a weighted probability model (e.g., dictionary attacks or frequency analysis), the probability of successful guesses changes over time. The entropy of the password system is defined as:

$$H(X) = -\sum_{i=1}^{\infty} p_i log_2 \, p_i$$

To approximate the behavior of this entropy as more guesses are made, we use logarithmic series expansions:

$$log(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n}, for \ |x| < 1$$

Substituting $x = p$, this expansion allows defenders to model decaying entropy over time as attackers successfully eliminate improbable options. For systems with partial breaches or leaked hashes, this approach enables a quantitative model of how much entropy remains; crucial for determining whether immediate resets or full credential rotations are needed.

5. Attack Convergence and Defensive Thresholds

If we denote the probability of a successful attack attempt on the nth trial as a_n, we can model the cumulative breach probability as a series:

$$P_{breac} = \sum_{n=1}^{\infty} a_n$$

Using the Integral Test, suppose $a_n = \frac{1}{nlogn}$, a slowly decreasing function:

$$\int_2^{\infty} \frac{1}{x \log x} dx = \infty \rightarrow series\ diverges$$

Thus, even rare attack attempts eventually accumulate enough probability to guarantee success, unless system rules (timeouts, lockouts, MFA) interrupt the loop. This is the calculus of breach inevitability in passive defense systems.

6. Modeling Defensive Adaptation as a Step Function

Suppose a system adjusts its defense posture (e.g., rate limits, honeypots, CAPTCHA triggers) based on detected attack patterns. Let $f(n)$ be a stepwise function triggered after a convergence threshold N :

$$f(n) = \begin{cases} 0\ if\ n < N \\ 1\ if\ n \geq N \end{cases}$$

This introduces a non-differentiable defense function, yet integrable as a piecewise-defined function, which shifts the system from a passive to an active defense mode. These functions can also be constructed as Heaviside step functions $H(n-N)$embedded into system code to simulate automatic policy switching based on attack convergence models. Conversely, by modeling decryption complexity through exponential and logarithmic series, time-bound summations, and convergence behavior, we equip security analysts with predictive tools to simulate intrusion strategies and preconfigure defenses. These models are not speculative, they are mathematically grounded, computationally practical, and ethically necessary.

IV. Loop Simulation and Real-Time Emulation

In the realm of encryption attacks, especially brute-force and dictionary-based intrusions, an adversary's effort is algorithmic—reducible to looping structures that systematically traverse a keyspace. While these loops are written in code, they are governed by mathematical growth and predictability. In this section, we employ recurrence relations, series approximations, and differential-based emulation techniques to simulate how attackers behave and how defenders can intervene before breach convergence occurs.

1. Attacker Behavior as a Recursive Loop

Let us define a loop-based brute-force attack attempting to decrypt a password of length n from a character set of size k. The structure of the loop resembles:

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
std::string convertToPassword(int num, int k, int n) {
    std::string result(n, '0');
    for (int i = n - 1; i >= 0; --i) {
        result[i] = '0' + (num % k);
        num /= k;
    }
    return result;
}
int main() {
    int k = 4;
    int n = 3;
    std::string password = "203";
    int limit = std::pow(k, n);
    for (int i = 0; i < limit; ++i) {
        std::string guess = convertToPassword(i, k, n);
        if (guess == password) {
            std::cout << "Password found: " << guess << " at i = " << i
<< std::endl;
            break;
        }
    }
    return 0;
}
```

Mathematically, this is equivalent to evaluating a recursive process of the form:

$$a_n = k \cdot a_{n-1} \ , \qquad a_0 = 1$$

The closed-form solution is:

$$a_n = k^n$$

This recurrence relation reveals the exponential growth of computational cost and matches the total number of iterations required by the brute-force attacker. From a defender's point of view, knowing this function allows for precise forecasting of when an attacker will reach certain thresholds.

2. Series Simulation of Attack Progress

Suppose an attacker can test g guesses per second. The cumulative number of guesses made by time t is:

$$G(t) = \sum_{i=1}^{t} g = g \cdot t$$

Now, we model the cumulative probability of success over time as a growing geometric sequence, where the probability p of success per guess is uniformly distributed over the keyspace of size $N = k^n$:

$$P(t) = \sum_{i=1}^{g \cdot t} \frac{1}{N} = \frac{g \cdot t}{N}$$

Solving for the time when $P(t) = 0.5$ (expected success):

$$t = \frac{0.5 \cdot N}{g}$$

This estimation yields a precise expected breach time, allowing analysts to preemptively implement countermeasures just before this time elapses.

3. Real-Time Emulation Using Differentiable Functions

Attack behavior over time can also be represented as a continuous function $f(t)$ that models the fraction of the keyspace exhausted:

$$f(t) = \frac{g \cdot t}{k^n}$$

We then define a defensive threshold θ (e.g., 10% of keyspace explored) and solve:

$$f(t) = \theta \rightarrow t = \frac{\theta \cdot k^n}{g}$$

Taking the derivative gives the instantaneous rate of attack progression:

$$f'(t) = \frac{d}{dt}\left(\frac{g \cdot t}{k^n}\right) = \frac{g}{k^n}$$

Since this is a constant, defenders may program monitoring systems to issue alerts when cumulative login attempts reach certain inflection points, calculated analytically in advance.

4. Emulating Intrusion Behavior with Piecewise & Parametric Models

Some attacks evolve dynamically: initial brute-force becomes dictionary-based, then morphs into credential-stuffing. To model this, we define:

$$f(t) = \begin{cases} \dfrac{g_1 \cdot t}{k^n} & 0 \leq t < T_1 \\ \dfrac{g_2 \cdot (t \cdot T_1)}{k^m} + \alpha & T_1 \leq t < T_2 \\ \dots & \dots \end{cases}$$

Here, g_1 and g_2 are different guess rates; k^m is a refined dictionary space; α is the residual probability from earlier stages. The use of piecewise functions and parametric definitions allows the defender to simulate evolving attacks in real-time dashboards.

5. Counter-Loops and Predictive Defense

To defend against attack loops, analysts write counter-loops that monitor user behavior, IP address entropy, failed attempts, and token usage. These functions often loop as well, forming recursive detection scripts:

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <cmath>
struct LoginAttempt {
    std::string ip;
};
double entropy(const std::string& ip) {
    std::unordered_map<char, int> freq;
    for (char c : ip) {
        freq[c]++;
    }
    double ent = 0.0;
    int length = ip.size();
    for (const auto& pair : freq) {
        double p = static_cast<double>(pair.second) / length;
        ent -= p * std::log2(p);
    }
    return ent;
}
void flag_as_suspicious(const std::string& ip) {
    std::cout << "Suspicious IP flagged: " << ip << std::endl;
}
int main() {
    std::vector<LoginAttempt> login_attempts = {
        {"192.168.1.1"},
        {"000.000.000.000"},
        {"182.34.90.12"},
        {"111.111.111.111"}
    };
    double threshold = 2.5;
    for (const auto& attempt : login_attempts) {
        if (entropy(attempt.ip) < threshold) {
            flag_as_suspicious(attempt.ip);
        }
    }
    return 0;
}
```

We can model this using series:

$$S = \sum_{n=1}^{\infty} \chi E(x_n)$$

Where $\chi E(x)$ is the characteristic function for an entropy-failure event, and x_n are data points. The convergence of this series indicates successful filtering over time.

6. Visualization via Phase Space & Rate of Change

Finally, consider a phase diagram modeling guess rate vs. success rate:

Let $x(t)$ = fraction of keyspace attempted, and $y(t)$ = fraction of successful intrusion attempts. Then:

$$\frac{dy}{dx} = \frac{dy\ /\ dt}{dx\ /\ dt} = \frac{g_s}{g}$$

Where g_s is the attacker's success rate. Plotting this in the (x,y) plane yields a real-time profile of attacker efficiency, helping defenders visualize when attacks are escalating and shifting in type. Finally, modeling attacker behavior through recursive loops, differentiable functions, and adaptive series empowers analysts to move from reactive to proactive defense. These simulations provide precise estimates of when and how attacks might breach a system—enabling preemptive mitigation, especially for networks defending vulnerable populations who cannot afford to respond after the fact.

VI. Conclusion: The Convergence of Struggle and Precision

In every summation lies a limit. In every series, a story of steps taken—incremental, determined, divergent, convergent. This paper has traced how infinite series, logarithmic expansions, recursive loops, and parametric thresholds can be wielded in the defense of digital systems. But beneath these formal systems lies another series: the series of moments that carry a life across borders, through hunger, through isolation, and toward education. You, Professor, once crossed a line of convergence—one not measured in calculus, but in courage. From the fractured pathways of Iran's revolution, you arrived not merely in geography, but in purpose. In the dry silence of exile and reinvention, you built something infinite: a system of thought rooted in mathematical truth, yet expanding with ethical depth. The partial sums of your struggle—language, culture, survival—did not diverge. They converged into legacy.

I, your student, walk my own axis. Born to another hemisphere of hardship, in a *colonia* that endured unfathomable challenges, behind school walls that never saw renovation, let alone a decent algebra professor, I inherited not wealth, but momentum. My series begins with the clarion voices of Martin Luther King Jr., Cesar Chavez, and John F. Kennedy, whose resounding calls for justice promised that progress must include the forgotten. As Holy Scripture reminds us: *"The last shall be first, and the first last; for many are called, but few are chosen."* Just as the Río Grande flows ceaselessly, so does the divine assurance that our efforts for equality and justice are guided by a force greater than ourselves.

My recursion is in every exam I barely passed, every line of code I refused to give up on, every time I translated pain into precision. And in this paper, we meet—not merely as aspiring analyst and professor, or Hispanic and Iranian, but as men of calculus: sons of broken places who found structure in series, and defiance in definitions. The limits we once feared became the very boundary conditions that made our function solvable. So let it be written, that security is not only a technical field, it is a human promise. That password entropy and decryption complexity are not just defenses against intruders, but guardians of dignity for those too often forgotten in digital design: the disabled, the isolated, the cognitively vulnerable. Let our work be their convergence, the sum of many unseen lives made whole through predictive foresight, recursive care, and infinite resolve. Because, in the end, the function that defines us is not written in the syntax of computers, but in the mathematics of hope.