Vulnerability Analysis of Vandermonde Matrices in Shamir's Secret Sharing

Azri Arzaq Pohan - 13524139
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524139@std.stei.itb.ac.id, azri.pohan@gmail.com

***Abstract*— Shamir's Secret Sharing (SSS) is a widely used threshold cryptographic scheme that secures a secret by interpolating a polynomial of degree k − 1 among n participants. While typically analyzed through the lens of modular arithmetic, the reconstruction process fundamentally relies on solving a system of linear equations governed by the properties of the Vandermonde matrix. This paper analyzes the structural vulnerability of SSS when implemented as an overdetermined system (n > k) in the presence of data corruption. By modeling the distributed shares as a linear system, we demonstrate that the redundancy originally intended for data availability inadvertently functions as an information leak. We implement a combinatorial attack strategy utilizing Lagrange interpolation and majority voting to mathematically isolate corrupted shares from an "honest majority" in a simulated (3,5) threshold scheme. The experimental results prove that without supplementary integrity controls, such as cryptographic hashes or digital signatures, an overdetermined SSS implementation is susceptible to combinatorial error correction attacks, effectively compromising the scheme's confidentiality guarantees.**

***Keywords*—Shamir's Secret Sharing, Vandermonde Matrix, Overdetermined System, Combinatorial Error Correction, Linear Algebra, Data Integrity.**

I. INTRODUCTION

In today's cybersecurity landscape, protecting sensitive data remains critical. Adi Shamir introduced one of the most popular and robust cryptographic methods in 1979 that is Shamir's Secret Sharing scheme. This scheme divides a cryptographic key into distinct parts called shares. Users need a minimum number of these shares to recover the original secret.

However, while polynomial interpolation frequently introduces the scheme, a deeper assessment of its computation viability reveals that solving a System of Linear Equations fundamentally represents the reconstruction process. This transformation necessitates the utilization of the Vandermonde Matrix, a structure known for its elegant algebraic properties but also for its inherent numerical fragility. The Vandermonde matrix's ill-conditioning worsens as threshold or share values grow. The system risks succumbing to numerical instability where standard floating-point computations fail to yield precise results, thereby exposing the cryptographic scheme to potential vulnerabilities.

Also, the reconstruction process's trustworthiness relies on geometric agreement of shares in Euclidean vector space. Because of the absence of strong validation, "cheating shares" or malicious inputs can be introduced, which can tamper with the secret. Consequently, a thorough vulnerability analysis employing determinants and vector spaces is a key necessity for securing Shamir's Secret Sharing in practical uses.

II. THEORETICAL BASIS

A. System of Linear Equations and Matrix Representation

A System of Linear Equations is a collection of linear equations involving the same set of variables, where each variable appears only to the first degree. An SPL with m equations and n unknowns can be expressed as:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}$$

Fig. 1. System of Linear Equations.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf]

And also can be expressed in matrix form:

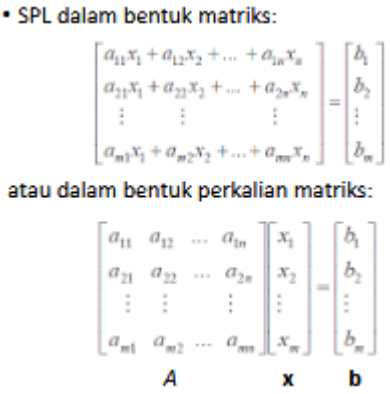

Fig. 2. System of Linear Equations in matrix form.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf]

Key properties of linear systems include:

1) Augmented Matrix Representation

An SPL can be represented as an augmented matrix, which combines the coefficient matrix and the constant vector into a single matrix:

$$[A \mid \mathbf{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

Fig. 3. Augmented Matrix Representation.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf]

2) Elementary Row Operations

Elementary Row Operations consist of:

- Swapping two rows,
- Multiplying a row by a nonzero scalar,
- Adding a multiple of one row to another.

The solution to an SPL is obtained by applying OBE to the augmented matrix until a row echelon matrix or reduced row echelon matrix is formed.

3) Gaussian Elimination Method

Gaussian Elimination is a procedure that applies Elementary Row Operations to transform an augmented matrix into row echelon form. The leading coefficients move progressively to the right, simplifying the structure of the system and enabling solution extraction.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_n \end{bmatrix} \sim \text{OBE} \sim \begin{bmatrix} 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & * \end{bmatrix}$$

Fig. 4. Gaussian Elimination Method
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf]

Once a system is in row echelon form, the values of unknowns can be determined sequentially by starting from the last equation also known as backward substitution and is guaranteed to yield a solution when the system has full rank.

B. Determinant

A determinant is a scalar value associated with a square matrix that captures algebraic and geometric information about the matrix.

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

Fig.5. $n \times n$ matrix.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf]

For an $n \times n$ matrix A, the determinant is denoted by $det(A)$.

$$\det(\mathrm{A}) = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$$

Fig. 6. Determinant for an $n \times n$ matrix.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf]

Key properties of determinant include:

1) Non-Zero Determinant and Invertibility

A square matrix A is invertible if and only if $det(A) \neq 0$. If $det(A) = 0$, the rows or columns of A are linearly dependent, and the matrix does not have an inverse.

2) Determinant of Triangular Matrices

For an upper or lower triangular matrix, the determinant is the product of its diagonal entries.

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \Longrightarrow \det(\mathrm{A}) = a_{11}a_{22}\,a_{33}a_{44}$$

Fig. 7. Upper Triangular Matrix.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf]

$$\mathrm{A} = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \Longrightarrow \det(\mathrm{A}) = a_{11}a_{22}\,a_{33}a_{44}$$

Fig. 8. Lower Triangular Matrix.
[Source:https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf]

3) Determinant Rules

- Swapping two rows multiplies the determinant by -1.
- Multiplying a row by a nonzero scalar k multiplies the Determinant by k.
- Adding a multiple of one row to another leaves the determinant unchanged.

A square matrix A is classified as singular (or non-invertible) if and only if its determinant equals zero ($det(A) = 0$). On solving a linear system $Ax = B$, a singular coefficient matrix implies that a unique solution does not exist, either no solution, or there are infinitely many solutions.

C. Vectors

A vector is a quantity possessing magnitude and direction, conventionally represented as an ordered list of numbers called components. In a 2-dimensional space, for example, a vector can be denoted as $v = \begin{bmatrix} v_1 \\ v_1 \end{bmatrix}$ and in a 3-dimensional space, a vector can be denoted as $v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$.

Key properties of vectors include:

1) Commutative Property

The order of vector addition does not alter the result. For any two vectors a and b, this expressed as:

$$a + b = b + a$$

2) Associative Property

When adding three or more vectors, the grouping of the vectors does not affect the final sum. For vectors a, b, and c:

$$(a + b) + c = a + (b + c)$$

3) Distributive Property

Vector addition is distributive over scalar multiplication. If λ is a scalar quantity, the operation holds that:

$$\lambda(a + b) = \lambda a + \lambda b$$

4) Additive Identity

There exists a zero vector (0) such that adding it to any vector a leaves the vector unchanged:

$$a + 0 = a$$

5) Additive Inverse

For every vector a, there exists a unique negative vector (−a) such that their sum results in the zero vector:

$$a + (-a) = 0$$

D. Vandermonde Matrix

The coefficient matrix A inherent to polynomial interpolation schemes manifests as a specific structured matrix known as the Vandermonde Matrix. Formally, for a set of distinct points $x_1, x_2, \ldots, x_n$, the matrix takes the following form:

$$V = \begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ & & \ldots & \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix}$$

Fig. 9. Vandermonde Matrix.
[Source: https://stevengong.co/notes/Vandermonde-System]

The most significant property of this matrix lies in its determinant. Unlike generic matrices that require computationally expensive decomposition to find the determinant, the Vandermonde determinant has a concise closed-form solution:

$$\det(A) = \prod_{1 \le i < j \le n} (x_j - x_i)$$

Fig. 10. Theorem DVM Determinant of a Vandermonde Matrix
[Source: http://linear.ups.edu/version3/scla/section-VM.html]

This formula proves that the matrix is invertible (non-singular) if and only if all x_i are distinct ($det(V) \neq 0$). As the dimension n increases or if the evaluation points x_i are chosen close to each other, the condition number of matrix grows exponentially creates a severe computational vulnerability that when solving the system $Va = y$ using standard floating-point arithmetic, even minute rounding errors can result in catasthropic inaccuracies in the reconstructed secret.

E. Shamir's Secret Sharing (SSS)

Shamir's Secret Sharing is a specific implementation of a threshold scheme developed by Adi Shamir in 1979 based on the mathematical concept of polynomial interpolation and denoted as (t, w) where t and w are positive integers where $t \leq w$. This scheme divides a secret message M among w participants in such a way that any subset of t participants can reconstruct M by combining their shares, whereas any group with fewer than t participants obtains no information about the secret.

The algorithm operates within a finite field defined by a prime number p, which must be larger than both the secret M and the number of participants w. To distribute the secret, the dealer constructs a random polynomial of degree $t - 1$:

$$s(x) \equiv M + s_1 x + s_2 x^2 + \cdots + s_{t-1} x^{t-1} \ (mod\ p)$$

The constant term is the secret itself ($s(0) \equiv M$), while the coefficients $s_1, s_2, \ldots, s_{t-1}$ are random integers chosen by the dealer. The dealer then distributes the shares by evaluating this polynomial at distinct integer points $x_1, x_2, \ldots, x_w$. Each participant receives a share consisting of the pair (x_i, y_i), where $y_i \equiv s(x_i)\ (mod\ p)$. When t participants combine their shares $(x_1, y_1), \ldots, (x_t, y_t)$ to recover the secret, they essentially substitute their coordinates into the polynomial equation. This substitution yields a system of t linear equations with the unknown coefficients $M, s_1, \ldots, s_{t-1}$.

$$\begin{pmatrix} 1 & x_1 & \ldots & x_1^{t-1} \\ 1 & x_2 & \ldots & x_2^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \ldots & x_t^{t-1} \end{pmatrix} \begin{pmatrix} M \\ s_1 \\ \vdots \\ s_{t-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{pmatrix} \pmod p$$

Fig. 11. Matrix Representation of the Reconstruction Phase in Shamir's Secret Sharing.
[Source: Author's own work]

The coefficient matrix on the left is a Vandermonde Matrix. To extract the secret M, one must resolve this system using Gauss-Jordan Elimination. Also, the Lagrange Interpolation method can solves the problem by generating a single unique polynomial of degree $t - 1$ that passes through the defined share vectors.

F. Lagrange Polynomial Interpolation

While the secret reconstruction process can be modeled as solving a system of linear equations involving a Vandermonde matrix, we can use an alternative and often more computationally direct approach is to use the Lagrange Polynomial method. The Lagrange polynomial is a unique polynomial of degree n that passes through a given set of $n + 1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ with $x_0, < x_1 < \cdots < x_n$.

The general formula for the lagrange interpolation is given by:

$$P(x) = \sum_{i=0}^{n} y_i L_i(x)$$

where each basis polynomial $L_i(x)$ is constructed using the product notation:

$$L_i(x) = \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

The term $L_i(x)$ evaluates to 1 when $x = x_i$ and 0 when $x = x_j$ (for $j \neq i$) that ensures the total polynomial $P(x)$ perfectly interpolates all the given data points, meaning $P(x_i) = y_i$ for all i.

III. STUDY CASE

A. Problem Definition

In this study case, we are going to analyze a secure data distribution that governed by Shamir's Secret Sharing Scheme (SSS). In this scheme, we called the "Dealer" as a trusted entity that intends to split sensitive secret S (represented as a numeric flag) among $n = 5$ distinct participants. The scheme is configured with a threshold parameter $k = 3$, implying that the secret is encoded into a polynomial $q(x)$ of degree $k - 1 = 2$. Any subset of 3 or more shares should suffice to reconstruct the unique polynomial and and recover S.

```
import random
import functools
from secrets import randbelow

# Config
FLAG = b"The system was successfully cracked."
PRIME = 2**384 - 317
THRESHOLD = 3
TOTAL_SHARES = 5

def text_to_int(text):
    return int.from_bytes(text, 'big')

def eval_poly(poly, x, prime):
    """Menghitung y = f(x) dalam Finite
Field"""
    y = 0
    for i, coeff in enumerate(poly):
        term = (coeff * pow(x, i, prime)) %
prime
        y = (y + term) % prime
    return y

# 1. Polynomial Setup
secret = text_to_int(FLAG)
poly = [secret] + [randbelow(PRIME) for _ in
range(THRESHOLD - 1)]

# 2. Real Shares
shares = []
for x in range(1, TOTAL_SHARES + 1):
    y = eval_poly(poly, x, PRIME)
    shares.append((x, y))

# 3. Fake Shares
traitor_idx = random.randint(0, TOTAL_SHARES -
1)
x_bad, y_good = shares[traitor_idx]
y_bad = (y_good + randbelow(PRIME)) % PRIME
shares[traitor_idx] = (x_bad, y_bad)

print("=== CHALLENGE OUTPUT ===")
print(f"Modulus P : {PRIME}")
print(f"Threshold : {THRESHOLD}")
print(f"Total Shares: {TOTAL_SHARES} (Warning:
One of them is CORRUPTED!)")
print("-" * 30)
print("Shares (x, y):")
for s in shares:
    print(s)
print("-" * 30)
print(f"(Pengkhianat ada di indeks
{traitor_idx}, ID {x_bad})")
```

The scenario assumes a compromised environment where the integrity of the distributed shares cannot be fully guaranteed. As simulated in the system configuration, there is exactly one share (x_c, y_c) among the five distributed shares has been corrupted due to transmission error or malicious modification. Consequently, the set of available shares $D = (x_1, y_1), \dots, (x_5, y_5)$ contains a mix of valid data points and an outlier.

We can see that this problem represents an overdetermined system of linear equations. Since the number of equations ($n = 5$) exceeds the number of unknowns ($k = 3$), the system will theoretically possesses redundancy. This means that the five points no longer lie on a single parabolic curve. The vector of share values y deviates from the column space of the corresponding Vandermonde matrix, making standard reconstruction like direct matrix inversion on the full set impossible.

B. Vulnerability Analysis

We observe that the vulnerability lies in the share generation mechanism within the **eval_poly** function.

```
def eval_poly(poly, x, prime):
    """Menghitung y = f(x) dalam Finite
Field"""
    y = 0
    for i, coeff in enumerate(poly):
        term = (coeff * pow(x, i, prime)) %
prime
        y = (y + term) % prime
    return y
```

The system treats shares as purely arithmetic objects points (x, y) on a polynomial curve without attaching any cryptographic integrity proofs such as Digital Signatures. In a secure protocol, each share should be a tuple (x, y, σ), where σ is a signature ensuring that y has not been altered. We found that the corrupted share (x_{bad}, y_{bad}) remains mathematically indistinguishable from valid shares when examined in isolation. This forces the system to rely on expensive combinatorial validation rather than immediate cryptographic rejection.

This vulnerability also significantly worsen by the system's architectural configuration. In the global variables, we identified a critical redundancy where the total number of shares ($n = 5$) exceeds the reconstruction threshold ($k = 3$). While redundancy is typically employed to ensure availability, in this context, it creates an overdetermined linear system that effectively leaks information about the validity of the data. We calculated that this configuration provides $\binom{5}{3} = 10$ potential reconstruction paths, allowing an observer to perform cross-verification among subsets that an operation would be impossible if the system had adhered to strict $n = k$ distribution. And lastly, we analyzed the fault injection logic and determined that the error rate is statistically insufficient to disrupt the consensus mechanism. The script corrupts exactly one share index **traitor_idx**, leaving four shares intact. This results in an honest-to-total ratio of 80% (4/5). We observe that for a majority voting attack to fail, the number of corrupted shares t must satisfy the condition:

$$t \geq n - k + 1$$

In this study case, $t = 1$, while the breaking point is $5 - 3 + 1 = 3$. Since the injected fault ($t = 1$) is far below the theoretical safety limit, the valid shares

overwhelmingly dominate the voting process. We conclude that the fault injection mechanism is too weak to prevent a brute-force combinatorial solver from identifying the outlier.

C. Attacking Strategy

We can use Combinatorial Error Correction via Majority Voting to attack the system. This attack exploits the mathematical redundancy inherent in an overdetermined Shamir's Secret Sharing scheme.

We observe that the system distributes $n = 5$ shares while only requiring a threshold of $k = 3$ for reconstruction. This information ($n > k$) provides us with a mechanism to distinguish between consistent shares and inconsistent shares without prior knowledge of the secret. The attack scheme is going to be like this:

1) Brute Force Combinations

Instead of relying on the entire dataset which we know contains inconsistencies, we can generate all possible unique subsets of size $k = 3$ from the $n = 5$ available shares.

$$\binom{5}{3} = 10\ combinations$$

2) Partial Reconstruction

For each generated subset, we can apply Lagrange Interpolation to derive a candidate secret. For consistent subsets, we hypothesize that subsets consisting entirely of valid shares that will mathematically converge to the identical secret value S. And for the subsets that contains the corrupted shared will yield divergent, randomized results due to the sensitivity of polynomial interpolation in finite fields.

3) Majority Voting

By aggregating the results from all 10 combinations, we can analyze the frequency distribution of the candidate secrets. We expect that the true secret to appear with a frequency of:

$$Frequency = \binom{n\ -\ 1}{k} = \binom{4}{3} = 4$$

The correct secret will be derived exactly 4 times (from the 4 subsets that exclude the corrupted share). Then we select the value with the highest frequency as the valid flag.

D. Solutions

To initiate the experiment, we executed the challenge script to generate the randomized dataset. As defined in the system configuration, the script produced a polynomial of degree 2 ($k = 3$) and evaluated it at 5 distinct points ($x = 1 \dots 5$). The fault injection mechanism selected a random index to corrupt.

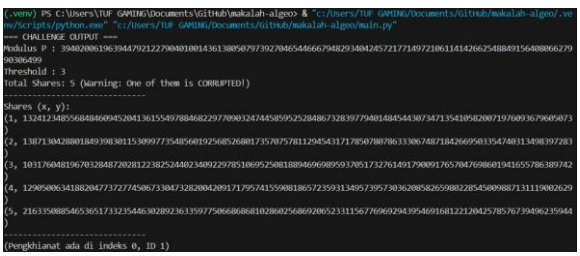

Fig. 12. Generation of 5 shares where one share is intentionally corrupted.

[Source: Author's own work]

Here's the solution script:

```
from itertools import combinations
from collections import Counter

PRIME =
394020061963944792122790401001436138050797392704654666794829340424572177149721061141426625488491564088066279
90306499

SHARES = [
    (1,
308676021034800564348764297968011543585925472678167397931458052756567652124519181543813738664444441721654282040257 61),
    (2,
194252297898787178368062941093887143275598904182575693677086288188735149363891961757141101144964767885250645426349 43),
    (3,
961340254219765736767691328914615926388443248691698140131254890745422733253576336141607341410733291003568109148819 7),
    (4,
300641858578995109684135341392221148581665466389536167152302895579378755867440190845950881361142920297926452645807 84),
    (5,
297448761543836560789111760992384268568644455356718269540799488468917143938232821804334077049326026259092973879084 6418)
]

THRESHOLD = 3

# Lagrange Interpolation
def inverse_mod(a, p):
    return pow(a, p - 2, p)

def lagrange_interpolate(shares, x_target, p):
    secret_sum = 0
    k = len(shares)

    for i in range(k):
        xi, yi = shares[i]

        # Hitung Basis Lagrange li(0)
        numerator = 1
        denominator = 1

        for j in range(k):
            if i == j: continue
            xj, yj = shares[j]

            # (0 - xj)
            num = (x_target - xj) % p
            numerator = (numerator * num) % p

            # (xi - xj)
            den = (xi - xj) % p
            denominator = (denominator * den)
% p

        # li(0) = numerator *
inverse(denominator)
        lagrange_poly = (numerator *
inverse_mod(denominator, p)) % p
        term = (yi * lagrange_poly) % p
        secret_sum = (secret_sum + term) % p

    return secret_sum

def int_to_text(number):
    try:
        h = hex(number)[2:]
        if len(h) % 2 != 0: h = '0' + h
```

```
        return bytes.fromhex(h).decode()
    except:
        return None

print(f"[*] Memulai Analisis pada
{len(SHARES)} titik...")
print(f"[*] Threshold sistem: {THRESHOLD}
(Butuh 3 titik konsisten)")

possible_secrets = []
debug_secrets = {}

for subset in combinations(SHARES, THRESHOLD):
    ids = tuple(s[0] for s in subset)

    res = lagrange_interpolate(subset, 0,
PRIME)
    possible_secrets.append(res)
    debug_secrets[ids] = res

counts = Counter(possible_secrets)
most_common_secret, frequency =
counts.most_common(1)[0]

print("\n=== LAPORAN HASIL ===")
print(f"Total Kombinasi Diuji:
{len(possible_secrets)}")
print(f"Rahasia Paling Konsisten Muncul:
{frequency} kali")

print("\n[DETEKSI ANOMALI]")
for ids, val in debug_secrets.items():
    status = "VALID" if val ==
most_common_secret else "RUSAK (Ada Share
Palsu)"
    print(f"  Kombinasi {ids} -> {status}")

print("\n=== DECODING FLAG ===")
flag = int_to_text(most_common_secret)
if flag:
    print(f"FLAG DITEMUKAN: {flag}")
else:
    print("Gagal decode flag.")
```

Lagrange interpolation over a finite field has a uniqueness property that is exploited in the solution. Because the related Vandermonde matrix is invertible, valid share combinations reliably reconstruct the identical secret by counting all threshold-sized subsets. Subsets with corrupted shares, on the other hand, provide different results. The original secret is thus revealed with high confidence by majority voting over the reconstructed secrets, illustrating a structural vulnerability resulting from redundancy without integrity checking.

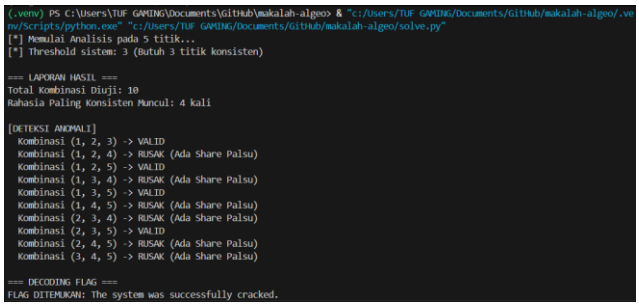

Fig. 13. Execution results of the solver script.
[Source: Author's own work]

IV. Conclusion

In this study, we have examined Shamir's Secret Sharing scheme's reconstruction phase from a linear algebraic standpoint, focusing on how it is formulated as a system of linear equations controlled by a Vandermonde matrix. We demonstrated how the algebraic consistency of the distributed shares is essential to the accuracy of secret reconstruction by looking at important ideas like matrix invertibility, determinants, vector spaces, and polynomial interpolation.

We showed that the system gets overdetermined when redundancy is added by allocating more shares than the reconstruction threshold. This redundancy can be used to find discrepancies between shares in the absence of cryptographic integrity safeguards. Even with a corrupted share, the original secret was successfully recovered using a combinatorial majority voting approach and Lagrange polynomial interpolation.In this work, we have examined Shamir's Secret Sharing scheme's reconstruction phase from a linear algebraic standpoint, focusing on how it is formulated as a system of linear equations controlled by a Vandermonde matrix. We emphasized how the algebraic consistency of the distributed shares is a vital requirement for the accuracy of secret reconstruction, using essential topics like matrix invertibility, determinants, vector spaces, and polynomial interpolation.

Because of the uniqueness property of polynomial interpolation and the invertibility of the accompanying Vandermonde matrices, the experimental results verify that legitimate share subsets consistently reconstruct the same secret, whereas subsets containing corrupted shares provide divergent results. This finding uncovers a structural flaw in Shamir's Secret Sharing when redundancy is not supported by sufficient integrity checking.

V. Appendix

The source code utilized to resolve study cases in this work is as follows.
https://github.com/AzriVz/makalah-algeo

VI. Acknowledgment

The author wishes to convey his sincere appreciation to Allah SWT for the unceasing strength, direction, and blessings that enabled him to finish this paper. Additionally, the author expresses profound gratitude to Dr. Ir. Rinaldi Munir, M.T. for his essential advice, perceptive instruction, and support during the course. Additionally, the author would like to express his sincere gratitude to his parents, siblings, and close friends for their steadfast moral and emotional support throughout his academic career at Institut Teknologi Bandung.

References

[1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: https://web.mit.edu/6.857/OldStuff/Fall03/ref/Shamir-HowToShareASecret.pdf. (Accessed: 22 December 2025).

[2] R. Munir, "Skema Pembagian Data Rahasia," *Kriptografi dan Koding (II4031)*, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2025. [Online]. Available:

https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2024-2025/35-Skema-Pembagian-Data-Rahasia-2025.pdf. (Accessed: 22 December 2025).
[3] R. Munir, “Bahan Kuliah IF2123 Aljabar Linier dan Geometri: Sistem Persamaan Linier (SPL)”, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung. Available : https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf (Accessed: 23 December 2025).
[4] R. Munir, "Determinan (Bagian 1)," *Aljabar Linier dan Geometri (IF2123)*, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf. (Accessed: 24 December 2025)
[5] Beezer, R. A. “A First Course in Linear Algebra”, University of Puget Sound. Available at: http://linear.ups.edu/html/section-VM.html (Accessed: 24 December 2025).
[6] GeeksforGeeks, "Properties of Vectors in Mathematics," *GeeksforGeeks*, 2024. [Online]. Available: https://www.geeksforgeeks.org/maths/properties-of-vectors/. (Accessed: 24 December 2025).
[7] Trefethen, L. N. “Vandermonde with Arnoldi”, Oxford University Mathematical Institute. Available at: https://people.maths.ox.ac.uk/trefethen/vander_revised.pdf (Accessed: 24 December 2025).
[8] "3.2: Polynomial Interpolation", *Mathematical Computing with Python*, LibreTexts. Available at: https://math.libretexts.org/Courses/Angelo_State_University/Mathematical_Computing_with_Python/3%3A_Interpolation_and_Curve_Fitting/3.2%3A_Polynomial_Interpolation (Accessed: 24 December 2025).
[9] S. Gong, "Vandermonde System," *Steven Gong Notes*. [Online]. Available: https://stevengong.co/notes/Vandermonde-System. (Accessed: 24 December 2025).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Azri Arzaq Pohan, 13524139