Application of Boolean Algebra in Frequency Hopping Spread Spectrum (FHSS) for UAV Communication Resilience Against Electronic Attack

Muhammad Naufal Hilmi - 13525029
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: m.naufalhilmi2007@gmail.com , 13525029@std.stei.itb.ac.id

***Abstract*—UAV communication links are uniquely vulnerable to radio jamming, since a UAV has no onboard pilot to take over when the wireless connection is disrupted. Frequency Hopping Spread Spectrum (FHSS) addresses this by rapidly switching the transmission across many different radio channels in a pattern that appears random to any outside observer, making it extremely difficult for an attacker to reliably interfere. The engine behind this switching pattern is a simple digital circuit called a Linear Feedback Shift Register (LFSR), which generates long, unpredictable sequences using only the most basic logic operation in digital systems. This paper examines how that mathematical foundation drives every part of FHSS, from generating the hopping sequence to selecting channels and keeping both the transmitter and receiver in sync. A worked example and a Karnaugh map simplification are provided to illustrate the design process, and the system is then tested against three realistic attack scenarios in a Python simulation: a jammer that stays on fixed channels, a jammer that chases the signal, and an attacker trying to guess the hopping pattern by brute force. The results confirm that resilience against each attack type is controlled by a single tunable parameter, while also revealing that the simplicity of the LFSR is a double-edged sword — the same property that makes it fast and cheap to implement also makes it mathematically breakable given enough observed output.**

Keywords—boolean algebra; fhss; lfsr; uav; anti-jamming

I. Introduction

The rapid advancement of technology and manufactoring capability of Unmanned Aerial Vehicles (UAVs) has transformed their role across both civilian and military sectors. Central to the operation of any UAVs system is its wireless communication link, which serves as the primary channel for command, control, dan telemetry data between the aircraft and its ground station.

However, the important communication environment in which UAVs operated are becoming more contested. The usage of electromagnetic spectrum to attack or surveil advesarial system pose a critical threat to UAV communication integrity. Among the most of prevalent forms of electronic attacks are radiofrequency jamming technique, which utilize electronic signal to disrupt or deny UAV communication link by overwhelming it with interference. Commercial-grade UAVs are particularly vunerable, as they commonly operate on publicly known frequency that could trivially be attacked using a low-cost jammer.

Frequency Hopping Spread Spectrum (FHSS) is an established countermeasure agaisnt such threats. Rather than transmitting on a fixed frequency, a pesudorandomized sequence synzhronized between the transmitter and receiver are used. An adversary without knowledge of the hopping sequenced cannot efectively jam or intercept the communication, as the signal occupies any given frequency only for a brief dwell period before jumping elsewhere.

At the core of FHSS is boolean algebra. The pseudorandom hopping sequence is generated by a Linear Feedback Shift Register (LFSR). As such, this paper is dedicated to the analysis of boolean algebra as the mathematical foundation of FHSS, and to evaluating how effectively an LFSR-based FHSS system holds up against realistic electronic attack scenarios.

II. Theoritical Background

A. Boolean Algebra

Boolean algebra is a branch of algebra that deals with variables restricted to two values, such as 0 and 1. The application of boolean algebra are diverse, from integrated circuit design to encryption.

These are the axiom in which boolean algebra are built upon:

1. Identity
   i. $a + 0 = a$
   ii. $a \cdot 1 = a$
2. Commutativity
   i. $a + b = b + a$
   ii. $a \cdot b = b \cdot a$
3. Complement

for every a there exist a unique a' such that $a + a' = 1$ and $a \cdot a' = 0$.

4. Distributivity

   i. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

   ii. $a + (b \cdot c) = (a + b) \cdot (a + c)$

5. Asociativity

   i. $a \cdot (b + c) = (a + b) + c$

   ii. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

From these axioms, the secondary laws follow: idempotence, absorption, domination, involution, and De Morgan's laws. Further derived from these laws and axioms are the logic gates that serves as the foundation of boolean algebra. The four most important logic gates areas follow:

1. AND Gate

TABLE I. NOT GATE TRUTH TABLE

Input		Output
A	**B**	
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

TABLE II. OR GATE TRUTH TABLE

Input		Output
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate

TABLE III. NOT GATE TRUTH TABLE

Input	Output
0	1
1	0

4. XOR Gate

TABLE IV. XOR GATE TRUTH TABLE

Input		Output
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	0

The most important operation for this paper is XOR. Looking at the table, XOR outputs 1 when its two inputs differ and 0 when they are the same. This is exactly the behavior of addition modulo 2: 0+0=0, 0+1=1, 1+0=1, 1+1=0 (mod 2). This equivalence means XOR is the addition operation of the binary field $GF(2) = \{0, 1\}$. Every time an LFSR computes its feedback bit using XOR, it is performing polynomial arithmetic over this field, which is why GF(2) is the algebraic setting in which LFSR theory is developed.

Any boolean function of n variables can be expressed in canonical Sum-of-Products (SOP) forms, as a disjunctions of minterms for which the function evaluates to 1, or in canonical Product-of-Sums (POS), as a conjunction of maxterms. To optimalized SOP or POS expression, Karnaugh Maps provide a graphical tool of grouping adjacent minterms or maxterms.

B. Linear Feedback Shift Register (LSFR)

An LFSR is a shift register, a chain of n single-bit storage cells, with a feedback connection. At every clock step, each cell copies the value of the cell to its left, the rightmost cell's value is output, and a new bit is computed from selected cells (*taps*) using XOR and fed into the leftmost cell. Because XOR is the only operation used in the feedback, the next state is a linear boolean function of the current state.

If the register holds bits $b_{n-1}, b_{n-2}, .., b_0$ and the taps are at positions $i_1, i_2, \ldots, i_k$, then the new feedback bit is:

$$f = b_1 \oplus b_2 \oplus \ldots \oplus b_k \quad (1)$$

This expression is just XOR applied to a subset of the current state bits, which is why the operation is called linear feedback.

The set of taps determines how long the LFSR's sequence will be before it repeats. The sequence length is determined by the corresponding feedback polynomial $P(x) = x^n + x^1 + x^2 + \ldots + 1$, where the exponents of the non-zero terms correspond to the tap positions. An n-bit LFSR achieves its maximum possible period of $2^n - 1$ steps if and only if this feedback polynomial is primitive over GF(2) [2]. A primitive polynomial is an irreducible polynomial whose roots generate the full multiplicative group of the extension field $GF(2^n)$. The practical meaning is if the polynomial is primitive, the LFSR

visits all $2^n - 1$ possible non-zero states exactly once before cycling back to the start. A sequence with this maximum-length property is called an m-sequence.

C. Frequency Hopping Spread Spectrum (FHSS)

FHSS is a wireless communication technique in which the carrier frequency is switched rapidly among N different channels following a pseudorandom sequence known to both the transmitter and receiver. Because an adversary without knowledge of this sequence cannot predict which channel will be used at any given moment, the adversary cannot reliably jam the signal without covering every channel at once.

An FHSS system has three main components. First, a pseudorandom sequence generator produces the hopping pattern. Second, a frequency synthesizer receives a channel index from the generator and tunes the radio to that frequency. Third, a hopping table maps each generator output value to one of the N available channel frequencies. All three of these components operate in lockstep on both the transmitting and receiving ends, so both sides are always listening and transmitting on the same channel at the same time.

Two key design parameters control the anti-jamming effectiveness of FHSS. The first is the hop-set size N, the total number of channels available. A larger N means a jammer has to spread its power across more frequencies to have any reliable effect. The second is the hop rate, the number of times per second the channel changes. A higher hop rate means the signal spends less time on each channel, reducing the window of time a reactive jammer has to detect and respond to the current channel before it has already moved.

D. Electronic Warfare and UAV Communication Threat

Electronic Warfare is the use of the electromagnetic spectrum as a weapon, either to attack an adversary's communications (Electronic Attack, EA), to protect one's own communications from attack (Electronic Protection, EP), or to gather information about the adversary's use of the spectrum (Electronic Support, ES). FHSS is an Electronic Protection measure.

UAVs are especially vulnerable to EA for a structural reason: the entire platform depends on a wireless link for command and control, and there is no human pilot on board who can fly the aircraft manually if that link is lost. Yu et al. note that commercial drones compound this vulnerability by using unhardened civilian communication systems that were not designed with adversarial environments in mind [5].

This paper evaluates FHSS against three specific EA scenarios:

1. Narrowband Jamming,
   The jammer transmits continous noise on M fixed channels and does nothing else. It does not track the hopping signal, it simply denies those M channels all the time.
2. Follower Jamming.
   The jammer listens to the spectrum and detects which channel the target is currently using. It then returns its own transmitter to that channel. However, this detection and retuning takes time (Δt) so it can only jam the current channel for the fraction of the dwell time that remains after it has reacted.
3. Brute-force Interception,
   The adversary does not jam at all. Instead, it tries every possible LFSR seed in turn until it finds the one that matches the observed hopping pattern. Once found, the adversary can predict all future hops, making a precise follower attack or selective interception trivial.

III. Boolean algebra in FHSS

A. LFSR as a Boolean Sequence Engine

The hopping sequence in an FHSS system is produced entirely by boolean operations. At every hop, the LFSR evaluates its feedback function, which consist of a chain of XOR gates coonecting the tap positions, and shifts to a new state. That new state is then mapped to a channel index, and the transmitter and receiver both jump to that channel. The entire process, from one hop to the next, consists of XOR operations and a bit shift. No other computation is involved.

The reason this simple mechanism produces a useful pseudorandom sequence is the primitive polynomial. When the feedback polynomial P(x) is primitive over GF(2), the LFSR state space is traversed completely before any state repeats. This means the channel indices produced over one full period are as evenly spread across the N available channels as possible, which is exactly what prevents a jammer from identifying a ‘favorite’ channel to target. If the polynomial were not primitive, the LFSR would cycle through only a fraction of the state space, producing a shorter, more predictable pattern that is easier to exploit. This shows that the unpredictability at the heart of FHSS is not incidental but a direct mathematical consequence of XOR being the additive operation of GF(2), and of the choice of a primitive polynomial in that field.

B. Frequency Selection Logic and Karnaugh Map Minimization

Once the LFSR has produced a new state, that state must be converted into a channel index. In the simplest case, a fixed number of bits from the LFSR output are read directly as a binary channel number. In practice, some channels may need to be excluded, such as channels reserved for a separate beacon or control signal that must remain on a fixed frequency.

To illustrate how boolean minimization handles this, consider a case where 4 LFSR output bits b_0, b_1, b_2, b_3 are used to address 16 channels, but the four channels 0 (0000), 5 (0101), 10 (1010), and 15 (1111) are reserved and must never be selected. A validity function $F(b_0, b_1, b_2, b_3)$ is needed: F = 1 when the addressed channel is available, and F = 0 when it is reserved.The Karnaugh map for F, shown in Table III, has 12 cells equal to 1 (non-reserved channels) and 4 cells equal to 0.

TABLE V. Karnaugh Map of Function F

b_0b_1,\b_2b_3	00	01	11	10
00	0	1	1	1
01	1	0	1	1
11	1	1	0	1
10	1	1	1	0

Grouping the 1-cells in the map yields the minimized expression:

$$F = b_0b_2' + b_0,'b_2 + b_1b_3' + b_1'b_3 \quad (2)$$

or

$$F = (b_0 \oplus b_2) + (b_1 \oplus b_3) \quad (3)$$

C. Synchronization as a Shared Boolean State

For FHSS to work, the transmitter (TX) and receiver (RX) must be on the same channel at the same time. Since both are driven by independent LFSR circuits, the only way to guarantee this is to ensure that both LFSRs hold the same internal state at all times. This is achieved by loading both registers with the same initial seed before communication begins.

From that shared starting point, both LFSRs advance through their state sequences in parallel. Because the next state of an LFSR is a deterministic booelan function of the current state, and because both registers start from the same state, they will always produce the same next state, and so on indefinitely. The synchronization condition can be stated simply: the link is synchronized if and only if the TX and RX LFSR states are equal bit by bit at every step.

This has an important consequence for robustness. If a sustained Electronic Attack causes one side to miss clock pulses or to reset its state, the two LFSRs will no longer hold the same state. Since both continue advancing deterministically from different starting points, they will never converge again on their own. The UAV's link is broken until an explicit resynchronization procedure restores a common state. This means that desynchronization, rather than jamming of any individual channel, is the true failure mode of FHSS under a sustained attack.

D. Security Bound of the LFSR Seed Space

An adversary who wants to break an FHSS system by guessing its seed must search through all possible seeds until the correct one is found. For an n-bit LFSR, there are $2^n - 1$ possible non-zero seeds (the all-zero seed is excluded because an LFSR initialized to all zeros stays at all zeros forever). Without any additional information, an exhaustive search must try, on average, half of these, giving an attack complexity of $O(2^n)$. Doubling n squares the cost of the search.

However, this $O(2^n)$ bound holds only against an attacker who has no information about the structure of the sequence. In practice, an LFSR-based sequence has a serious weakness. Because its feedback function is linear (XOR only), the sequence itself is linear. Massey proved that given a segment of an LFSR's output of length roughly 2n, the Berlekamp–Massey algorithm can efficiently reconstruct the smallest LFSR that produces that output, recovering both the feedback polynomial and the initial seed [7]. This means an adversary who can passively observe even a short run of the hopping pattern can, in principle, predict all future hops without ever performing an exhaustive seed search. The LFSR's linearity, which is what makes it so fast and cheap to implement, is also what makes it cryptographically fragile. This limitation motivates the improvements discussed in Section V.

IV. Simulation

A. Simulation Environment

B. LFSR Sequence Verification

An 8-bit Galois LFSR was run from a non-zero seed for its full theoretical period of $2^8 - 1 = 255$ steps. The simulation produced exactly 255 distinct states with no repetition before returning to the seed. This confirms the maximal-length property: every non-zero 8-bit pattern appears exactly once per cycle, consistent with the primitive-polynomial condition discussed in Section II-B.

The low-order 4 bits of each LFSR state were then used to select one of N = 16 hopping channels. The channel-usage frequency over the full 255-step period is shown in Table IV.

```python
from collections import Counter

# LFSR core

def lfsr_step(state, tap_mask):
    """Advance a Galois LFSR by one clock step."""
    lsb = state & 1
    state >>= 1
    if lsb:
        state ^= tap_mask
    return state

def run_lfsr(seed, n, tap_mask):
    """Run the LFSR for its full period (2^n - 1 steps) and return all states."""
    state = seed
    sequence = []
    for _ in range(2**n - 1):
        sequence.append(state)
        state = lfsr_step(state, tap_mask)
    return sequence

# Parameters

n        = 8
tap_mask = 0b10110100
seed     = 1
N        = 16

# Run and verify

sequence = run_lfsr(seed, n, tap_mask)

distinct_states = len(set(sequence))
expected_period = 2**n - 1

print(f"LFSR degree         : n = {n}")
print(f"Expected period     : 2^{n} - 1 = {expected_period}")
print(f"Distinct states seen: {distinct_states}")
print(f"Maximal-length?     : {distinct_states == expected_period}")
```

Fig 1. LFSR Sequence Verification
(Source: Author)

TABLE VI. Channel Usage Distribution Over One Full LFSR Period (N=8,N=16)

Statistic	Value
Total steps	255
Distinct LSFR states produced	255
Minimum channel-visit count	15
Maximum channel-visit count	16

Statistic	Value
Expected count if perfectly uniform	255 / 16 ≈ 15.94

Each of the 16 channels was visited either 15 or 16 times, almost a perfect distribution. This confirms that the m-sequence produced by the LFSR spreads the hopping pattern across all available channels with no significant bias, which is what FHSS requires to deny a jammer any single predictable target.

C. *Narrowband Jamming*

In this scenario, a static jammer permanently occupies M = 2 channels out of N total. The jamming success rate is defined as the fraction of all hops that land on a jammed channel: rate = M / N. Since the LFSR distributes hops uniformly (as shown in IV-B), this fraction is simply M divided by N. The hop-set size N was varied from 8 to 64 to evaluate how increasing the number of available channels reduces the jammer's effectiveness.

```python
import matplotlib.pyplot as plt

# Parameters

M           = 2
hop_set_sizes = [8, 16, 32, 64]

# Calculate jamming success rate

print(f"Jammer occupies M = {M} fixed channels.\n")
print(f"{'N':>6}  {'Jamming success rate':>22}")
print("-" * 32)

rates = []
for N in hop_set_sizes:
    rate = M / N
    rates.append(rate * 100)
    print(f"{N:>6}  {rate * 100:>20.2f}%")
```

Fig 2. Narrowband Jamming Simulaton
(Source: Author)

TABLE VII. NARROWBAND JAMMING SUCCES RATE

N	Jamming success rate (%)
8	25.00
16	12.50
32	6.25
64	3.12

Each time N doubles, the jamming success rate halves. This is because the jammer's M = 2 occupied channels become a smaller and smaller fraction of a larger channel set, while the hopping pattern continues to spread uniformly. A jammer that blocks 25% of hops on an 8- channel system blocks only 3.12% of hops if the same system is upgraded to 64 channels, without any change to the jammer's hardware or power output.

D. *Follower Jamming*

The follower jammer reacts to each hop by sensing the current channel and retuning its own transmitter to that channel. The reaction takes a fixed delay of Δt = 2 ms. After the jammer has reacted, it can interfere for the remaining fraction of the dwell time T_{dwell} = 1 / hop rate. The vulnerability windowis the fraction of each dwell period that the jammer successfully covers, its formula is:

$$\text{vulnerability window} = \max(0, T_{dwell} - \Delta t) / T_{dwell} \quad (4)$$

The hop rate was varied to show how faster hopping shrinks this window.

```python
import matplotlib.pyplot as plt

# Parameters

delta_t   = 0.002
hop_rates = [50, 100, 200, 500, 1000]

# Calculate vulnerability windows
windows = []
for hop_rate in hop_rates:
    T_dwell = 1 / hop_rate
    window  = max(0, T_dwell - delta_t) / T_dwell
    windows.append(window * 100)
    print(f"{hop_rate:>20}  {T_dwell * 1000:>15.1f}  {window * 100:>21.1f}%")
```

Fig 3. Follower Jamming Simulaton
(Source: Author)

TABLE VIII. FOLLOWER JAMMING VULNERABILITY WINDOW

Hop rate (hops/s)	T_{dwell} (ms)	Vulnerability window (%)
50	20.000	90.00
100	10.000	80.00
200	5.000	60.00
500	2.000	0.00
1000	1.000	0.00

At low hop rates, the jammer has plenty of time to react and still cover most of each dwell period. As the hop rate increases, the dwell time shrinks. Once the dwell time is equal to or shorter than the jammer's reaction delay (at 500 hops/s in this example, where T_{dwell} = Δt = 2 ms), the vulnerability window reaches zero where the jammer finishes retuning just as the signal has already moved on to the next channel. Beyond that threshold, the follower jammer is completely defeated regardless of how much power it uses.

E. *Brute-Force Interception*

In the worst case for the attacker, the adversary has no prior information about the seed and must try every possible value. The size of the search space for an n-bit LFSR is $2^n - 1$. The LFSR degree n was varied to show how quickly the search space grows.

```python
import matplotlib.pyplot as plt

# Parameters

lfsr_degrees = [4, 8, 16, 32, 64]

# Calculate search spaces

print(f"{'n (bits)':>10}  {'Search space (2^n - 1)':>25}")
print("-" * 38)

search_spaces = []
for n in lfsr_degrees:
    space = 2**n - 1
    search_spaces.append(space)
    print(f"{n:>10}  {space:>25,}")
```

Fig 4. Brute-force interception Simulation
(Source: Author)

TABLE IX. TABLE STYLES

n (bits)	Search space ($2^n - 1$)
4	15
8	255
16	65,535
32	4,294,967,295
64	18,446,744,073,709,551,615

Going from n = 4 to n = 8 multiplies the search space by 17, from n = 16 to n = 32 multiplies it by roughly 65,536, and from n = 32 to n = 64 multiplies it by 4,294,967,297. The growth is exponential. Each additional bit of LFSR state doubles the brute-force cost, so a modest increase in register size produces a dramatic increase in the cost of a naive exhaustive attack. As noted in Section III-D, however, this exponential bound does not protect against the Berlekamp–Massey algorithm, which can break an LFSR-based sequence in polynomial time given a sufficient sample of observed outputs.

V. CONCLUSION

This paper has demonstrated the application of boolean algebra as the direct operating mechanism of Frequency Hopping Spread Spectrum. The XOR operation over GF(2), governed by a primitive feedback polynomial, produces the maximum-length pseudorandom sequence that allows FHSS to deny jammers any predictable target. Karnaugh map minimization then converts that sequence into an efficient channel-selector circuit, and the shared LFSR seed functions as the system's secret key. However, FHSS has the downside of forced mismatch between transmitter and receiver states breaks the link irreversibly until explicit resynchronization.

Four Python simulations quantified these properties against three Electronic attack scenarios. Narrowband jamming success fell from 25% to 3.12% as the hop-set grew from 8 to 64 channels. Follower jamming vulnerability collapsed to zero once the hop rate exceeded the jammer's reaction threshold. Brute-force search complexity grew from 15 to over 4 billion as LFSR degree increased from 4 to 32 bits. Notably, each result is controlled by a single independently tunable parameter, that is hop-set size N, hop rate, and register degree n respectively, which makes FHSS a modular and computationally lightweight defense that can be strengthened along one dimension without disturbing the others.

One fundamental limitation nevertheless remains. Because LFSR feedback is purely linear, the Berlekamp–Massey algorithm can recover the complete register state from a short observed output segment, bypassing the exponential brute-force bound entirely. Addressing this requires moving beyond linear sequence generators, whether through Nonlinear Feedback Shift Registers, AES-encrypted output sequences, or Index Modulation-based FHSS extensions [4]. All three directions build on the same boolean-algebraic foundation analyzed in this paper and represent natural next steps for future work.

ATTACHMENT

Google CoLab: Source Code of "Application of Boolean Algebra in Frequency Hopping Spread Spectrum (FHSS) for UAV Communication Resilience Against Electronic Attack"

ACKNOWLEDGMENT

The author is grateful to God Almighty for the guidance and strength to complete this work, to family for their constant support, to the Dr. Ir. Rinaldi Munir, M.T., lecturer of IF1220 Discrete Mathematics for providing the knowledge that made this paper possible, and to all others who contributed in any way to its completion.

REFERENCES

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] R. Munir, "Aljabar Boolean," Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025- 2026/21-Graf-Bagian2-2026.pdf. [Accessed: Jun. 12, 2026].

[2] S. W. Golomb, Shift Register Sequences, Aegean Park Press, 1982.

[3] R. A. Poisel, Modern Communication Jamming Principles and Techniques, 2nd ed., Artech House, 2011..

[4] Y. Shi et al., "Efficient Index Modulation Based FHSS: A Unified Anti-Jamming Perspective," IEEE Transactions on Communications, 2023.

[5] Yu et al., "Electronic Warfare Cyberattacks, Countermeasures and Modern Defensive Strategies of UAVAvionics: A Survey," arXiv preprint, 2025.

[6] A. Benhallam et al., "Frequency Hopping Spread Spectrum Security Improvement with Encrypted Spreading Codes," SCIRP Journal of Information Security, 2013.

[7] J. Massey, "Shift-Register Synthesis and BCH Decoding," IEEE Transactions on Information Theory, vol. 15, no. 1, pp. 122–127, 1969.

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, 19 Juni 2025

Muhammad Naufal Hilmi, 13525029