International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Simulating Brute Force Attacks on Vigenère and AES Ciphers in Python: Measuring Key Size Impact on Security

Virgilio F. Tuga Jr.[1], Lloyd Vincent P. Cubero[2]

[1,2]Faculty, College of Information Technology Education, North Eastern Mindanao State University – Tandag Campus

ABSTRACT

The security of modern digital communications depends heavily on the robustness of cryptographic algorithms against brute force attacks. This study investigates the impact of key size on the brute force resistance of two representative symmetric ciphers: the classical Vigenère cipher and the Advanced Encryption Standard (AES). Using Python, the research implements a systematic simulation that incrementally increases key sizes and measures the corresponding computational effort required to recover plaintext through exhaustive key search. The simulation spans key lengths of 2–5 characters for Vigenère and 12–20 bits for AES (with reduced keyspace for feasibility), recording both the average time and the number of attempts needed to break each cipher. Results show an exponential growth in attack effort with increasing key size for both algorithms: even modest increments in key length yield orders-of-magnitude improvements in brute force resistance. These findings provide concrete, empirical evidence supporting cryptographic best practices that mandate adequate key lengths for secure communications. The study underscores the continued relevance of key size as a fundamental defense mechanism in both legacy and modern symmetric encryption systems, offering valuable insights for security practitioners and educators alike.

Keywords: Advanced Encryption Standard, brute force attack, key size, Vigenère cipher

1.INTRODUCTION

The increasing reliance on cryptographic algorithms to secure both classical and modern communications has heightened the importance of evaluating their resilience against brute force attacks. This study specifically investigates the vulnerability of two prominent symmetric ciphers—the Vigenère cipher, representative of classical encryption methods, and the Advanced Encryption Standard (AES), which is widely adopted in modern applications. By simulating brute force attacks using Python, this research aims to systematically measure and analyze the effect of key size on the security strength of these algorithms, providing empirical insight into their respective levels of resistance [1], [2], [3].

A substantial body of literature has examined the weaknesses of cryptographic schemes under brute force methodologies. The Vigenère cipher has been well-documented for its susceptibility, particularly when shorter keys are used [1]. In contrast, AES, though currently regarded as robust, still faces potential threats from exhaustive key search, especially as advances in computational power continue to accelerate [2]. Foundational studies by Alkhwaja et al. [3] and others [4], [5] have demonstrated that incremental

increases in key size can exponentially raise the computational complexity required for successful attacks. Further, research on brute force techniques has encompassed scenarios such as attacks on secured shell servers [6] and password breach detection [7], while innovative strategies like all-or-nothing transforms have been proposed to mitigate the risks of partial decryption through brute force means [8].

Despite the breadth of existing research, there remains a gap in comparative simulation-based analysis of brute force attacks on both classical and modern symmetric ciphers using a unified Python framework. Many studies focus on either theoretical aspects or case-specific attacks, often overlooking the direct, side-by-side measurement of key size effects across fundamentally different cryptographic paradigms. Additionally, while password cracking and server attack simulations abound, empirical modeling of brute force resistance in Vigenère and AES—using a replicable, accessible tool like Python—remains underexplored, particularly with respect to quantifying the practical computational demands involved [3], [9], [10].

The significance of this study lies in its practical approach to elucidating the real-world implications of key size choices in cryptographic design. By providing quantifiable evidence on how key length directly impacts the resistance of Vigenère and AES ciphers to brute force attacks, the research contributes actionable insights for both security practitioners and educators. The findings not only reinforce the centrality of key space expansion as a defense mechanism [5], but also inform future cryptographic implementations in diverse environments where balancing security and efficiency is paramount. Ultimately, this work aims to deepen understanding of the dynamic interplay between key size and brute force resilience, thereby supporting the development of more robust and future-ready encryption systems.

2. METHODOLOGY

This study adopts a simulation-based experimental approach to quantitatively measure the impact of key size on the brute force resistance of two symmetric ciphers: the Vigenère cipher and the Advanced Encryption Standard (AES). The research systematically increases the key size for each cipher and records both the time and the number of attempts required to successfully recover the original plaintext through brute force attack.

2.1 Simulation Environment

All simulations were conducted on a personal computer with the following specifications:

1. Processor: Intel(R) Core(TM) i5 or equivalent, 2.60 GHz, 8GB RAM
2. Operating System: Windows 10
3. Python Version: Python 3.10+
4. Libraries:

a. **pycryptodome** (for AES operations)
b. **matplotlib** and **numpy** (for data visualization and processing)
c. Standard Python libraries (**itertools**, **string**, **random**, **time**)

2.2 Cipher Implementation and Attack Procedure

- **Vigenère Cipher**
- The Vigenère cipher was implemented with the uppercase English alphabet as the keyspace (A-Z).
- The plaintext "**THEQUICKBROWNFOX**" was encrypted using a randomly generated key for each key length tested.
- Key lengths of 2, 3, 4, and 5 characters were selected.
- For each key length, a brute force attack was simulated by exhaustively generating and testing every

possible key of that length, checking the decrypted output against the known plaintext to identify the correct key.

- The time taken and the number of key attempts were recorded for each attack.
- **AES Cipher (with Reduced Key Size)**
- AES was implemented in ECB mode using the **pycryptodome** library.
- For simulation feasibility, the keyspace was deliberately reduced: key sizes of 12, 14, 16, 18, and 20 bits were tested (real AES uses 128 bits or higher, but this is computationally infeasible to brute force in a lab setting).
- For each key size, a random integer in the specified range was selected as the secret key, and expanded/padded to 16 bytes.
- The plaintext "**THE QUICK BROWN**" was encrypted and brute-forced by iterating through every possible key value in the keyspace, comparing the decrypted result to the original plaintext.
- The time required and number of attempts were measured for each setting.

2.4 Experimental Procedure

▪ ***For each key size (Vigenère: 2–5 chars; AES: 12–20 bits)***

○ Conduct three independent trials using different randomly selected keys.
○ Encrypt the chosen plaintext with the generated key.
○ Run the brute force attack routine to recover the key.
○ Record the time taken and the number of key guesses/attempts for each trial.

▪ ***Aggregate the results***

○ Calculate the average time and average number of attempts required to recover the plaintext for each key size.

▪ ***Visualization***

○ Plot key size against both the average brute force time and the average number of attempts, including logarithmic (log2) scales to highlight exponential growth.
○ Present all results in both summary tables and graphical plots.

2.5 Limitations

For AES, only severely reduced key sizes (up to 20 bits) could be simulated due to computational constraints. While this does not reflect real-world security, the approach illustrates the underlying exponential growth in brute force complexity as key size increases. Results for Vigenère are also limited to feasible key lengths for exhaustive search on standard hardware.

3. RESULTS AND ANALYSIS

The simulation systematically measured the brute force effort required for different key sizes in both the Vigenère and AES ciphers. Table 1 summarizes the average time and number of key attempts needed to successfully recover the original plaintext for each tested key size. Results are averaged over three independent trials per setting.

Cipher	Key Size	Avg Time (s)	Avg Attempts
Vigenère	2 chars	0.00	393
Vigenère	3 chars	0.07	12,797
Vigenère	4 chars	0.41	80,907
Vigenère	5 chars	38.50	7,539,059

AES	12 bits	0.01	1,755
AES	14 bits	0.03	4,683
AES	16 bits	0.21	29,174
AES	18 bits	1.21	179,775
AES	20 bits	4.09	607,826

Table 1: Brute Force Attack Results for Vigenère and AES Ciphers

3.1 Key Size Impact: Exponential Growth

Both ciphers exhibited exponential increases in brute force effort as the key size increased:

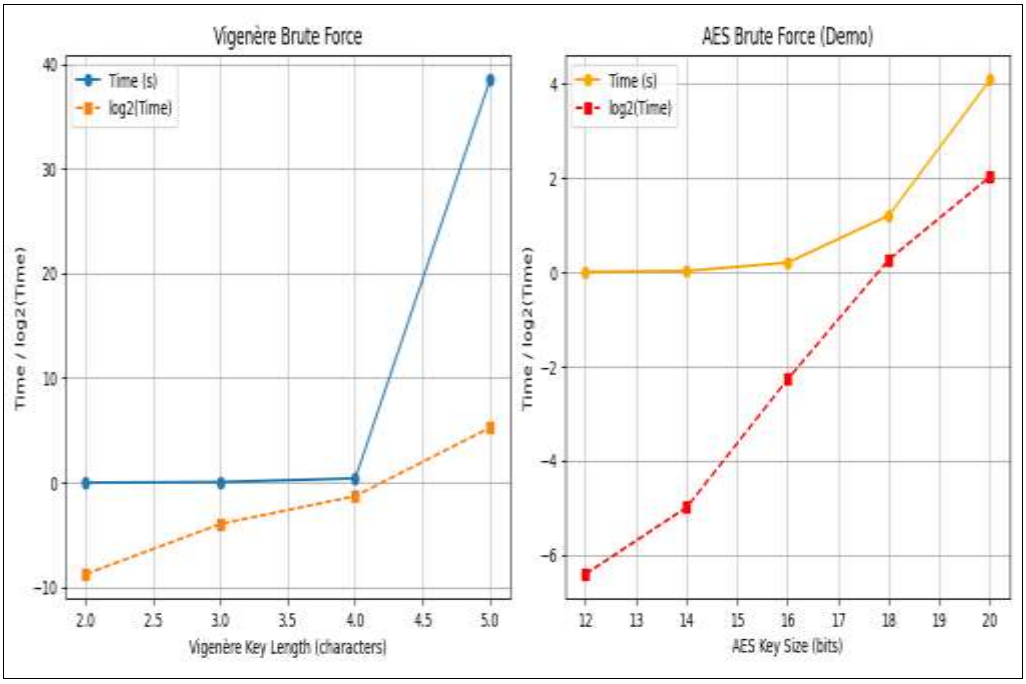
A. Vigenère Cipher

Doubling the key length from 2 to 4 characters caused the number of brute force attempts to soar from just under 400 to over 80,000, with the time required rising from a fraction of a second to nearly half a second. At a key length of 5, the attack required more than 7.5 million attempts and an average of 38.5 seconds to complete. This exponential trend is visually confirmed in Figure 1 (left), where both time and log2(time) escalate sharply with key size.

A. AES Cipher (with Reduced Key Size)

Increasing the AES key size from 12 to 20 bits led to a similar exponential pattern: attempts rose from 1,755 to over 600,000, while the average attack time grew from just 0.01 seconds to over 4 seconds. While these are not real-world key sizes, the results clearly demonstrate the relationship between key length and brute force resistance, as reflected in Figure 1 (right).

Figure 1: Brute Force Attack Time vs. Key Size for Vigenère and AES (logarithmic scale)



3.2 Security Implications

The simulation confirms that key size is a critical factor in determining a cipher's resistance to brute force attacks. Each incremental increase in key length results in an exponential growth in the number of possible keys, and thus in the computational effort required for an attacker to succeed. For example, increasing the Vigenère key by just one character multiplies the keyspace by 26; for AES, each additional bit doubles the keyspace.

In real-world systems, this is why modern cryptographic recommendations mandate key lengths that make brute force attacks computationally infeasible—such as 128 bits or higher for AES. The demonstrated results provide concrete, empirical support for these best practices.

4. CONCLUSIONS

This study simulated brute force attacks on both a classical (Vigenère) and a modern (AES) symmetric cipher using Python, systematically measuring the impact of key size on cryptographic security. The results provide clear, quantitative evidence that brute force resistance increases exponentially with key size for both types of ciphers. Even modest increases in key length yield orders-of-magnitude improvements in security, quickly rendering exhaustive key search attacks impractical for sufficiently long keys.

The findings highlight the enduring importance of appropriate key size selection in both legacy and modern encryption systems. While Vigenère's security becomes meaningful only at impractically long key lengths, AES's security—when using recommended key sizes—is effectively unbreakable by brute force. These results underscore the fundamental principle that key size remains one of the most crucial defenses against brute force attacks in symmetric cryptography.

For practitioners and educators, this simulation demonstrates and justifies, through direct measurement and visualization, why cryptographic standards emphasize adequate key length as a cornerstone of digital security.

5. APPENDIX

5.1. Python Source Code

```
import time
import itertools
import string
import random
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import matplotlib.pyplot as plt
import numpy as np

# Vigenère Cipher implementation
def vigenere_encrypt(plaintext, key):
    alphabet = string.ascii_uppercase
    plaintext = plaintext.upper()
    key = key.upper()
    encrypted = ''
```

```
    for i, char in enumerate(plaintext):
        if char in alphabet:
            idx = (alphabet.index(char) + alphabet.index(key[i % len(key)])) % 26
            encrypted += alphabet[idx]
        else:
            encrypted += char
    return encrypted

def vigenere_decrypt(ciphertext, key):
    alphabet = string.ascii_uppercase
    key = key.upper()
    decrypted = ''
    for i, char in enumerate(ciphertext):
        if char in alphabet:
            idx = (alphabet.index(char) - alphabet.index(key[i % len(key)])) % 26
            decrypted += alphabet[idx]
        else:
            decrypted += char
    return decrypted

def brute_force_vigenere(ciphertext, key_length, known_plaintext):
    alphabet = string.ascii_uppercase
    attempts = 0
    for key_tuple in itertools.product(alphabet, repeat=key_length):
        key = ''.join(key_tuple)
        candidate = vigenere_decrypt(ciphertext, key)
        attempts += 1
        if candidate == known_plaintext:
            return key, attempts
    return None, attempts

# AES with short keys
def aes_encrypt(plaintext, key_bytes):
    cipher = AES.new(key_bytes, AES.MODE_ECB)
    return cipher.encrypt(pad(plaintext, AES.block_size))

def aes_decrypt(ciphertext, key_bytes):
    cipher = AES.new(key_bytes, AES.MODE_ECB)
    return unpad(cipher.decrypt(ciphertext), AES.block_size)

def brute_force_aes(ciphertext, known_plaintext, key_bits):
    attempts = 0
    found_key = None
```

```
    for k in range(2 ** key_bits):
        key_bytes = k.to_bytes(16, byteorder='big')
        attempts += 1
        try:
            pt = aes_decrypt(ciphertext, key_bytes)
            if pt == known_plaintext:
                found_key = key_bytes
                break
        except Exception:
            continue
    return found_key, attempts

def log2safe(x):
    return np.log2(x) if x > 0 else 0

# Main Simulation
def main():
    # Settings
    n_trials = 3
    # Vigenère settings
    vigenere_key_lengths = [2, 3, 4, 5]
    plaintext = "THEQUICKBROWNFOX"
    # AES settings
    aes_key_bits = [12, 14, 16, 18, 20]  # Up to 2^20 = 1,048,576 attempts
    aes_plaintext = b'THE QUICK BROWN'

    # Data storage
    v_times, v_attempts = [], []
    aes_times, aes_attempts = [], []

    print("=== Vigenère Brute Force Simulation ===")
    for key_length in vigenere_key_lengths:
        times, attempts = [], []
        for _ in range(n_trials):
            key = ''.join(random.choices(string.ascii_uppercase, k=key_length))
            ciphertext = vigenere_encrypt(plaintext, key)
            start = time.time()
            result_key, n_attempts = brute_force_vigenere(ciphertext, key_length, plaintext)
            elapsed = time.time() - start
            times.append(elapsed)
            attempts.append(n_attempts)
        avg_time = sum(times) / n_trials
        avg_attempts = sum(attempts) / n_trials
```

```
        v_times.append(avg_time)
        v_attempts.append(avg_attempts)
        print(f"Key length {key_length}: Avg time = {avg_time:.2f}s, Avg attempts =
{avg_attempts:.0f}")

    print("\n=== AES Brute Force Simulation ===")
    for bits in aes_key_bits:
        times, attempts = [], []
        for _ in range(n_trials):
            secret_int = random.randint(0, 2 ** bits - 1)
            key_bytes = secret_int.to_bytes(16, byteorder='big')
            ciphertext = aes_encrypt(aes_plaintext, key_bytes)
            start = time.time()
            result, n_attempts = brute_force_aes(ciphertext, aes_plaintext, bits)
            elapsed = time.time() - start
            times.append(elapsed)
            attempts.append(n_attempts)
        avg_time = sum(times) / n_trials
        avg_attempts = sum(attempts) / n_trials
        aes_times.append(avg_time)
        aes_attempts.append(avg_attempts)
        print(f"Key bits {bits}: Avg time = {avg_time:.2f}s, Avg attempts = {avg_attempts:.0f}")

    # Print Summary Table
    print("\n--- Summary Table ---")
    print("Vigenère KeyLen\tTime(s)\tAttempts\tAES KeyBits\tTime(s)\tAttempts")
    for i in range(max(len(vigenere_key_lengths), len(aes_key_bits))):
        vkl = vigenere_key_lengths[i] if i < len(vigenere_key_lengths) else ''
        vt = f"{v_times[i]:.2f}" if i < len(v_times) else ''
        va = f"{v_attempts[i]:.0f}" if i < len(v_attempts) else ''
        akb = aes_key_bits[i] if i < len(aes_key_bits) else ''
        at = f"{aes_times[i]:.2f}" if i < len(aes_times) else ''
        aa = f"{aes_attempts[i]:.0f}" if i < len(aes_attempts) else ''
        print(f"{vkl}\t\t{vt}\t{va}\t\t{akb}\t\t{at}\t{aa}")

    # Visualization
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(vigenere_key_lengths, v_times, marker='o', label="Time (s)")
    plt.plot(vigenere_key_lengths, [log2safe(x) for x in v_times], marker='s', linestyle='--',
label="log2(Time)")
    plt.xlabel('Vigenère Key Length (characters)')
    plt.ylabel('Time / log2(Time)')
```

```
    plt.title('Vigenère Brute Force')
    plt.grid(True)
    plt.legend()

    plt.subplot(1,2,2)
    plt.plot(aes_key_bits, aes_times, marker='o', color='orange', label="Time (s)")
    plt.plot(aes_key_bits, [log2safe(x) for x in aes_times], marker='s', linestyle='--', color='red',
label="log2(Time)")
    plt.xlabel('AES Key Size (bits)')
    plt.ylabel('Time / log2(Time)')
    plt.title('AES Brute Force (Demo)')
    plt.grid(True)
    plt.legend()

    plt.tight_layout()
    plt.show()

    # Extra: Attempts visualization
    plt.figure(figsize=(10,5))
    plt.plot(vigenere_key_lengths, v_attempts, marker='o', label="Vigenère Attempts")
    plt.plot(aes_key_bits, aes_attempts, marker='o', color='orange', label="AES Attempts")
    plt.yscale('log')
    plt.xlabel('Key Size (chars/bits)')
    plt.ylabel('Number of Attempts (log scale)')
    plt.title('Brute Force Attempts vs. Key Size')
    plt.grid(True)
    plt.legend()
    plt.show()

if __name__ == "__main__":
    main()
```

REFERENCES

1. P. Purwanti, S. Nurcahya, & D. Nazelliana, "Message security in classical cryptography using the vigenere cipher method", International Journal Software Engineering and Computer Science (Ijsecs), vol. 4, no. 1, p. 350-357, 2024. https://doi.org/10.35870/ijsecs.v4i1.2263
2. W. Putra, S. Suyanto, & M. Zarlis, "Performance analysis of the combination of advanced encryption standard cryptography algorithms with luc for text security", Sinkron, vol. 8, no. 2, p. 890-897, 2023. https://doi.org/10.33395/sinkron.v8i2.12202
3. I. Alkhwaja, M. Albugami, A. Alkhwaja, M. Alghamdi, H. Abahussain, F. Alfawazet al., "Password cracking with brute force algorithm and dictionary attack using parallel programming", Applied Sciences, vol. 13, no. 10, p. 5979, 2023. https://doi.org/10.3390/app13105979
4. X. Liu and X. Hu, "Research on techniques for detecting brute-force attacks on corporate email",

Journal of Computational Methods in Sciences and Engineering, vol. 24, no. 3, p. 1379-1393, 2024. https://doi.org/10.3233/jcm-247147
5. M. Hoobi, "Multilevel cryptography model using rc5, twofish, and modified serpent algorithms", Iraqi Journal of Science, p. 3434-3450, 2024. https://doi.org/10.24996/ijs.2024.65.6.37
6. A. Raj, M. Chauhan, V. Chhoker, M. Rani, B. Singh, R. D'Souzaet al., "Brute forcing on secured shell servers emphasising the role of cyber forensics – a quali-quantitative study", Medico-Legal Journal, vol. 92, no. 3, p. 152-157, 2024. https://doi.org/10.1177/00258172241236269
7. H. TAŞÇI, S. Gönen, M. BARIŞKAN, G. KARACAYILMAZ, B. ALHAN, & E. Yılmaz, "Password attack analysis over honeypot using machine learning password attack analysis", Turkish Journal of Mathematics and Computer Science, vol. 13, no. 2, p. 388-402, 2021. https://doi.org/10.47000/tjmcs.971141
8. Y. Gu, S. Akao, N. Esfahani, Y. Miao, & K. Sakurai, "On the information-theoretic security of combinatorial all-or-nothing transforms",, 2022. https://doi.org/10.48550/arxiv.2202.10280
9. N. Hs, R. Ma, & P. Magadum, "Detection of brute force attacks on iot networks", International Research Journal of Modernization in Engineering Technology and Science, 2024. https://doi.org/10.56726/irjmets50643
10. E. Dogruluk, J. Macedo, & A. Costa, "A countermeasure approach for brute-force timing attacks on cache privacy in named data networking architectures", Electronics, vol. 11, no. 8, p. 1265, 2022. https://doi.org/10.3390/electronics11081265