International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT25113346

Strengthening AES Encryption: A Novel Approach Using Prime Number-Based Passwords

Dr. G. Sreedhar

Professor, Department of Computer Science, National Sanskrit University, Tirupati, Andhra Pradesh, India

Article Info

Publication Issue :
Volume 8, Issue 4
July-August-2022
Page Number : 418-421
Article History
Accepted: 20 July 2022
Published: 14 Aug 2022

ABSTRACT

This paper explores the enhancement of Advanced Encryption Standard (AES) encryption by integrating prime number sequences into password generation. Prime numbers, known for their mathematical properties, are utilized to create robust passwords that augment the security of AES encryption. The research demonstrates the implementation of this approach using the Web Crypto API, highlighting its effectiveness in strengthening cryptographic operations in web browsers.

Keywords: Advanced Encryption Standard, Prime numbers, Encryption, Decryption.

1. INTRODUCTION

The security of digital communications relies heavily on encryption algorithms, with AES being a widely adopted standard due to its efficiency and strength. However, the security of AES is contingent upon the strength of the encryption key. Traditional methods of key generation may not provide sufficient randomness, making them susceptible to attacks. This paper proposes an innovative approach to enhance AES encryption by incorporating prime number sequences into password generation, thereby increasing the entropy and unpredictability of the encryption keys.

2. BACK GROUND

2.1 Advanced Encryption Standard (AES)

AES is a symmetric key encryption algorithm that operates on fixed block sizes and supports key lengths of 128, 192, or 256 bits. It employs a series of transformation rounds, including substitution, permutation, and mixing operations, to convert plaintext into ciphertext. The strength of AES is largely determined by the key size and the complexity of the key schedule, which generates round keys from the original encryption key.

2.2 Prime Numbers in Cryptography

Prime numbers play a crucial role in various cryptographic algorithms, such as RSA, due to their mathematical properties that facilitate secure key generation and encryption processes. Incorporating prime numbers into password generation can enhance the randomness and complexity of the keys, making them more resistant to attacks.

3. METHODOLOGY

3.1 Prime Number Sequence Generation

A sequence of prime numbers is generated using an efficient primality test algorithm. These prime

numbers are then mapped to characters to form a password. The length of the password is determined based on the desired security level, with longer passwords providing greater security.

3.2 Password Generation Process

1. Generate a list of prime numbers up to a specified limit.
2. Map each prime number to a corresponding character in a predefined character set.
3. Concatenate the characters to form a password of the desired length.
4. Use this password as the key for AES encryption.

3.3. AES Algorithm Overview

AES operates on fixed-size blocks of 128 bits and supports key sizes of 128, 192, or 256 bits. The number of rounds in AES depends on the key size:

- **AES-128**: 10 rounds
- **AES-192**: 12 rounds
- **AES-256**: 14 rounds

Each round involves a series of transformations to ensure the security of the encrypted data.

3.4. AES Encryption Process

The AES encryption process involves several steps:

1. **Initial Round**:
   - **AddRoundKey**: XOR the plaintext block with the first round key.
2. **Main Rounds** (repeated for 9, 11, or 13 times depending on key size):
   - **SubBytes**: Substitute each byte in the state using the S-box.
   - **ShiftRows**: Cyclically shift the rows of the state.
   - **MixColumns**: Mix the columns of the state to provide diffusion.
   - **AddRoundKey**: XOR the state with the current round key.
3. **Final Round**:
   - **SubBytes**
   - **ShiftRows**
   - **AddRoundKey**: No MixColumns in the final round.

After the final round, the state contains the ciphertext.

3.4.1 AES Encryption Algorithm (Pseudocode)

```
function AES_Encrypt(plaintext, key):
state = ConvertToState(plaintext)
  state = AddRoundKey(state, key[0..3])
  for round = 1 to Nr-1:
    state = SubBytes(state)
    state = ShiftRows(state)
    state = MixColumns(state)
    state = AddRoundKey(state, key[4*round..4*round+3])
  end for
  state = SubBytes(state)
  state = ShiftRows(state)
  state = AddRoundKey(state, key[4*Nr..4*Nr+3])
  return state
end function
```

Figure 1: AES Encryption using Prime Numbers based password

3.5 AES Decryption Process

The AES decryption process is the reverse of encryption, applying the inverse of each transformation in reverse order:

1. **Initial Round**:
   - **AddRoundKey**: XOR the ciphertext block with the last round key.
2. **Main Rounds** (repeated for 9, 11, or 13 times depending on key size):
   - **InvSubBytes**: Substitute each byte in the state using the inverse S-box.
   - **InvShiftRows**: Cyclically shift the rows of the state in the opposite direction.
   - **InvMixColumns**: Apply the inverse of the MixColumns transformation.
   - **AddRoundKey**: XOR the state with the current round key.
3. **Final Round**:
   - **InvSubBytes**
   - **InvShiftRows**
   - **AddRoundKey**: No InvMixColumns in the final round.

After the final round, the state contains the original plaintext.

3.5.1 AES Decryption Algorithm (Pseudocode)

```
function AES_Decrypt(ciphertext, key):
  state = ConvertToState(ciphertext)
  state = AddRoundKey(state, key[4*Nr..4*Nr+3])
  for round = Nr-1 downto 1:
    state = InvShiftRows(state)
    state = InvSubBytes(state)
    state = AddRoundKey(state, key[4*round..4*round+3])
    state = InvMixColumns(state)
  end for
  state = InvShiftRows(state)
  state = InvSubBytes(state)
  state = AddRoundKey(state, key[0..3])
  return state
end function
```

Figure 2: AES Decryption using Prime Numbers based Password

4. CONCLUSION

The implementation of prime number-based password generation demonstrates a significant improvement in the security of AES encryption. By increasing the entropy of the encryption key, the system becomes more resistant to brute-force and dictionary attacks. Integrating prime number sequences into password generation offers a novel approach to enhancing the security of AES encryption. This method increases the unpredictability and complexity of the encryption keys, thereby strengthening the overall security of cryptographic systems. Future research may explore the application of this approach in other cryptographic algorithms and its potential in post-quantum cryptography scenarios. The integration of prime numbers into password generation not only enhances security but also aligns with modern cryptographic practices that emphasize the importance of key unpredictability and resistance to attacks. Future work may explore the

application of this method in various cryptographic protocols and its potential in post-quantum cryptography scenarios.

REFERENCES

[1]. Maurer, U. (1995). Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters. Journal of Cryptology, 8(3), 123–155.(crypto.ethz.ch)

[2]. Bach, E. (1988). How to generate factored random numbers. SIAM Journal on Computing, 17(5), 904–916. (Wikipedia)

[3]. Daemen, J., & Rijmen, V. (2002). The Design of Rijndael: AES – The Advanced Encryption Standard. Springer.(TechNewsWorld)

[4]. Elliptic Curve Primality Proving (ECPP). (1986). Mathematics of Computation. (Wikipedia)

[5]. Koblitz, N. (1987). Elliptic curve cryptosystems. Mathematics of Computation, 48(177), 203–209.

[6]. Miller, V. S. (1986). Use of elliptic curves in cryptography. In Advances in Cryptology – CRYPTO '85 Proceedings (pp. 417–426). Springer.

[7]. NIST. (2001). FIPS PUB 197: Announcing the Advanced Encryption Standard (AES). National Institute of Standards and Technology.(Wikipedia)

[8]. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120–126.

[9]. Shoup, V. (2008). A Computational Introduction to Number Theory and Algebra. Cambridge University Press.(Wikipedia)