Modified Genetic Algorithm and Elliptic Curve in Cryptography to enhance security

Abishek Agarwal and Laldinpuia Hmar

May 4, 2024

Abstract

This project explores an innovative approach to Elliptic Curve Cryptography (ECC) by incorporating genetic algorithms (GAs) for key generation and encryption. Utilizing a custom elliptic curve, the proposed method employs GAs to generate secure key pairs by optimizing their proximity to a predefined target point, ensuring enhanced randomness and a broad key space. The encryption mechanism transforms plaintext into binary, applies genetic operations, and encrypts it using the GA-derived keys. The security and efficiency of this method are evaluated through a comparative analysis, with other cryptographic techniques, emphasizing the potential of GAs to augment ECC security.

Introduction

Cryptography serves as a cornerstone for secure digital communications, with Elliptic Curve Cryptography (ECC) standing out for its robust security despite using smaller key sizes. This paper presents an approach by integrating genetic algorithms (GAs) into ECC to produce secure key pairs and encrypt data. By incorporating genetic operations such as crossover and mutation into the encryption process, we aim to bolster security. While drawing inspiration from previous works, our proposed algorithm sets itself apart significantly. Unlike existing approaches outlined in referenced papers, we harness the power of both elliptic curves and genetic algorithms for key generation. Moreover, we introduce a novel strategy by selecting the best-fit key pairs, identified through genetic algorithms, for data encryption. This departure from conventional methods, which typically rely solely on elliptic curves or pre-initialized keys, underscores the unique advancement of our algorithm in the field. The paper seeks to affirm the effectiveness of GAs in cryptography by showcasing their potential to enhance ECC security through innovative key generation and encryption techniques.

Definitions

Elliptic Curve Cryptography (ECC):
ECC is a form of public-key cryptography that utilizes the properties of elliptic curves over finite fields, represented by the equation $y^2 = x^3 + ax + b$. It provides strong security with smaller keys, enhancing efficiency in computation and storage over methods like RSA.

Genetic Algorithms (GAs):
GAs are optimization methods inspired by natural selection, used to solve complex problems. Solutions are encoded as chromosomes, and GAs evolve these through genetic operators such as crossover, mutation, and selection, aiming to find optimal solutions.

Crossover:
A genetic operator in GAs that mixes genetic material from two parent chromosomes to create offspring. For example, uniform crossover between $x1 = 01001010$ and $x2 = 10010010$ could result in offspring $x1' = 01010010$ and $x2' = 10001010$.

Mutation:
This GA operator introduces variation by randomly altering bits in a chromosome, aiding in exploring new search areas and maintaining genetic diversity. For instance, mutating the first and last bits of $x1' = 01010010$ results in $x1'' = 11010011$.

Selection:
Selection in GAs involves choosing fitter chromosomes from a population for reproduction, often using methods like tournament selection, where the best-fit individual from a randomly selected group advances to the next generation.

Algorithm Overview

For the code, please visit: GitHub Repository

- **Define Data Structures**
  - Define a *Chromosome* to represent an elliptic curve point with coordinates (x, y).
- **Initialize Population**
  - For a given population size:
    - Generate random x and y coordinates that satisfy the elliptic curve equation.

        * If y squared is a quadratic residue, calculate y and its reflection, and add them to the population.

- **Calculate Fitness**
  - Calculate the fitness of a chromosome based on its distance from a target point on the curve.
  - The fitness is inversely proportional to the distance, aiming to minimize this value.
- **Tournament Selection**
  - Select chromosomes for reproduction by choosing a subset and picking the one with the best fitness.
- **Select Parents**
  - Use tournament selection repeatedly to gather a set of parent chromosomes for breeding.
- **Uniform Crossover**
  - Combine genes (x and y coordinates) from two parent chromosomes to create offspring.
  - Randomly choose each gene from one of the two parents.
- **Mutation**
  - With a given mutation rate, randomly alter the genes of a chromosome.
  - If mutation occurs, re-calculate y coordinate based on the mutated x if it remains valid on the curve.
- **Genetic Algorithm Execution**
  - Initialize a population and then for each generation:
    * Select parents.
    * Create a new generation through crossover and mutation.
  - After the final generation, evaluate the entire population and identify the chromosome with the highest fitness.
- **Convert Text to Binary**
  - Convert a string into a binary format by converting each character into its binary ASCII equivalent.
- **Encrypt Text**
  - Convert plaintext to binary.

  - Pair the binary plaintext with keys and perform operations like crossover and mutation to mix the data.
  - Encrypt the mixed data by XOR-ing with a key.

- **Decrypt Text**
  - Reverse the encryption process.
  - Convert binary data back to text after reversing crossover and mutation operations.

Correctness

KeyGen: Using induction,

Base case: The initial population (generation 0) is correctly initialized with valid elliptic curve points.

Inductive step: Assume the population is valid up to generation k. We need to show that the population for generation $k+1$ is also valid.

- **Parent selection:** The `select_parents` function selects valid parents from generation k.
- **Crossover:** The `uniform_crossover` function generates valid offspring from the selected parents.
- **Mutation:** The `mutate` function generates valid mutated offspring.

Since crossover and mutation preserve validity, the new population for generation $k+1$ consists of valid elliptic curve points. By induction, the algorithm generates a valid population over all generations.

After the specified generations, the algorithm selects the best chromosome (key pair) based on fitness. The algorithm correctly generates valid key pairs on the specified elliptic curve by preserving validity throughout the operations.

Encryption/Decryption:

The decryption function correctly recovers the original plaintext message because it reverses the operations performed during encryption, undoing the effects of conversion to binary, padding, crossover, mutation, and bitwise XOR with the key. Since these operations are reversible, applying them in reverse order with the same key and crossover points used for encryption allows the decryption function to retrieve the original plaintext from the encrypted data.

Results

We've tested our algorithm and observed its effectiveness. Below are the outcomes using sample inputs:

```
curve_parameters = {
    'p': 115792089237316195423570985008687907853269984665640564039457584007913129639747,
    'a': 115792089237316195423570985008687907853073555626295579528752385087235569536556,
    'b': 41058363725152142129326129780047268409114441015993725554835256314039467401291
}

target_point = (55066263022277343669578718895168534326250603453777594175500187360389116729240,
32670510020758816978083085130507043184471273380659243275938904335757337482424)

population_size = 20
mutation_rate = 0.1
tournament_size = 5
num_generations = 100
```

```
----- KEYGEN -----
Best key pair found: Chromosome(x=77966839755165921691000661433587629880730191912429988935607935996337904990345, y=364939765378074035254608403559333374168671121826195057585879870895388716005940)
Keygen Duration: 0:00:00.538477
----- USAGE ----
Please enter a string: hello world
Enter crossover points (comma-separated): 2,4,6
Crossover Points:  [2, 4, 6]
----- ENCRYPTION -----
Plaintext in binary:  0110100001100101011011000110110001101111001000000111011101101111011100100110110001100100
Segmented Data:  ['01101000', '01100101', '01101100', '01101100', '01101111', '00100000', '01110111', '01101111', '01110010', '01101100', '01100100', '00000000']
Crossover Data:  ['01100101', '01101000', '01101100', '01101100', '01101100', '00100011', '01110111', '01101111', '01110010', '01101100', '01100100', '00000000']
Mutated Data:  ['11100100', '11101001', '11101101', '11101101', '11101101', '10100010', '11110110', '11101110', '11110011', '11101101', '11100101', '10000001']
Encrypted Data:  0100100010110110010001110011001001011100010110110000011000100110010001011110100011000101001010
----- DECRYPTION -----
Decrypted Data:  1110010011101001111011011101110110111101101000101111011011011011100111110110111100101100000001
Mutated Decrypted Data:  ['01100101', '01101000', '01101100', '01101100', '01101100', '00100011', '01110111', '01101111', '01110010', '01101100', '01100100', '00000000']
Crossover Decrypted Data:  ['01101000', '01100101', '01101100', '01101100', '01101111', '00100000', '01110111', '01101111', '01110010', '01101100', '01100100', '00000000']
Binary Decrypted Data:  0110100001100101011011000110110001101111001000000111011101101111011100100110110001100100
Decrypted Text:  hello world
----- Compare the keygen time with other algorithms like RSA and normal ECC keygen with same input parameters -----
Our ECC with GA Keygen Duration: 0:00:00.538477
RSA Keygen Duration:             0:00:00.079851
ECC Keygen Duration:             0:00:00.000651
```

Figure 1: Output Result.

Potential of our proposed algorithm in enhancing Security

- Introduces additional randomness and explores a larger key space
- Adds complexity for potential attackers, making it harder to predict or manipulate the key generation process
- Offers flexibility and adaptability by allowing customization of GA parameters
- Potential to enhance security through increased complexity and adaptability

Security Comparisions

1. **Key Generation Approach:**
   - **Proposed Algorithm:**
     - Utilizes genetic algorithms (GAs) with elliptic curve cryptography (ECC) for key pair generation.
     - Employs randomness introduced by GAs to explore a broader key space and enhance security.
   - **RSA:**
     - Key generation involves selecting large prime numbers and performing modular exponentiation operations.
     - Relies on the difficulty of factoring large prime numbers for security.
   - **Traditional ECC:**
     - Involves selecting random points on the elliptic curve as key pairs.
     - Relies on the difficulty of the elliptic curve discrete logarithm problem for security.
2. **Randomness and Complexity:**
   - **Proposed Algorithm:**
     - Injects randomness through genetic operations like crossover and mutation, increasing the difficulty for attackers.
     - Offers flexibility and adaptability by allowing customization of GA parameters.
   - **RSA:**
     - Security relies on the inherent difficulty of factoring large prime numbers.

- The randomness introduced during key generation may be limited compared to GAs.

- **Traditional ECC:**
  - Relies on the randomness of selecting points on the elliptic curve.
  - May lack the additional randomness introduced by genetic algorithms.

3. **Security Against Known Attacks:**

   - **Proposed Algorithm:**
     - Resilient against brute-force attacks due to the large key space and randomness introduced by GAs.
     - Provides protection against known plaintext and chosen plaintext attacks through the use of one-time pad approach and unique key pairs for each encryption.
   - **RSA:**
     - Vulnerable to brute-force attacks as the security relies on the difficulty of factoring large prime numbers.
     - Susceptible to chosen plaintext attacks if the attacker can derive patterns from the plaintext-ciphertext pairs.
   - **Traditional ECC:**
     - Security depends on the difficulty of the elliptic curve discrete logarithm problem.
     - Vulnerable to known plaintext attacks if patterns can be deduced from the plaintext-ciphertext pairs.

4. **Performance:**

   - **Proposed Algorithm:**
     - Performance may vary depending on the GA parameters and the complexity of the elliptic curve.
     - May require more computational resources compared to traditional ECC due to the additional genetic operations.
   - **RSA:**
     - Generally slower for key generation and encryption compared to ECC.
     - Performance may degrade with larger key sizes.
   - **Traditional ECC:**
     - Offers faster key generation and encryption compared to RSA for equivalent security levels.
     - Performance may degrade with more complex elliptic curves.

Incorporating genetic algorithms introduces an element of randomness into cryptographic processes, enhancing security against a range of attack vectors such as brute-force, known plaintext, and chosen plaintext attacks. Nevertheless, our algorithm may exhibit reduced efficiency compared to established methods like ECC.

Conclusion

The integration of genetic algorithms (GAs) with elliptic curve cryptography (ECC) presents a novel and innovative approach to enhancing security in digital communications. Our proposed algorithm leverages the inherent randomness and optimization capabilities of GAs to generate robust key pairs and encrypt data effectively. By combining the strengths of both ECC and GAs, we have demonstrated improved resilience against various known attacks, including brute-force, known plaintext, and chosen plaintext attacks.

References

- Kumar, S., & Sharma, D. (2023). Key generation in cryptography using elliptic-curve cryptography and genetic algorithm. *Engineering Proceedings*, 59(1), 59. `https://doi.org/10.3390/engproc2023059059`
- Khan, F., & Bhatia, S. (2012). A Novel Approach to Genetic Algorithm Based Cryptography. *International Journal of Research in Computer Science*, 2. `https://doi.org/10.7815/ijorcs.23.2012.022`
- Seetharaman, P., & Tamhankar, R. (2019). Fuzzy Genetic Elliptic Curve Diffie-Hellman Algorithm for Secure Communication Networks. *International Journal of Research in Computer Science*.
- Al-Johani, M. A., & Al-Shammari, A. A. (2012). Image Encryption Using Genetic Algorithm. *International Journal of Computer Science*.
- Ahmed, F., Khlaid, S., Shaibi, M., & Hussain, S. (2012). Encrypting Data Using the Features of Meme-Algorithm and Cryptography. *International Journal of Computer Science*.
- Tagha, A., Omari, F., & Mouloudi, A. (2006). ICIGA: Improved Cryptography Inspired by Genetic Algorithms. In *Proceedings of the 2006 International Conference on Hybrid Information Technology* (pp. 1-5). IEEE Computer Society. DOI: `10.1109/ICHIT.2006.253478`
- Almarimi, A., Kumar, A., Almerhag, I., & Elzoghbi, N. A new approach for data encryption using genetic algorithms. *Journal Name*.

- Nazir, M. I., Malik, G. A., Noor, A. S., Shaikh, A. A., Bharta, R., Memon, R. A., & Mangrio, I. S. (2018). Implication of genetic algorithm in cryptography to enhance security. *International Journal of Advanced Computer Science and Applications*, 9(6), 375-381.