Proposal of a New Block Cipher reasonably Non-Vulnerable against Cryptanalytic Attacks

Abhijit Chowdhury¹, Angshu Kumar Sinha¹, Saurabh Dutta²

¹ Department of Computer Applications, NSHM College of Management and Technology, Durgapur, West Bengal, INDIA
² Dr. B. C. Roy Engineering College, Durgapur-713206, West Bengal, INDIA

Abstract
This paper proposes a new block cipher termed as “Modular Arithmetic based Block Cipher with Varying Key-Spaces (MABCVK)” that uses private key-spaces of varying lengths to encrypt data files. There is a simple but intelligent use of theory of modular arithmetic in the scheme of the cipher. Based on observed implementation of the proposed cipher on a set of real data files of several types, all results are tabulated and analyzed. The schematic strength of the cipher and the freedom of using a long key-space expectedly can make it reasonably non-vulnerable against possible cryptanalytic attacks. As a part of the future scope of the work, it is also intended to formulate and implement an enhanced scheme that will use a carrier image to have a secure transmission of the private key.

Keywords: Cryptography, Modular Arithmetic, Carrier Image, Data Security

1. Introduction

Module arithmetic [1], a strong backbone in numerous existing ciphering protocols [2], has been applied in an intelligent manner to develop “Modular Arithmetic based Block Cipher with Varying Key-Spaces” with the acronym as MABCVK. Although the encryption of data using MABCVK results in encrypted data of extended dimension, its schematic strength and the liberty of using a long key-space while encrypting using it expectedly can make MABCVK reasonably non-vulnerable against possible cryptanalytic attacks.

Section 2 presents the scheme followed in MABCVK. A report of implementation of MABCVK on a sample plaintext is demonstrated in section 3. Results of a set of real implementations are presented in section 4. Section 5 is an analytical observation to MABCVK. A conclusion is drawn in section 6.

2. The Scheme

The data to be encrypted is divided in blocks each of L-bits. All these blocks are converted into equivalent set of integers of base-10. Next, two sufficiently large prime integers K₁ and K₂ are taken as keys satisfying the conditions:

1. K₁, K₂ being sufficiently bigger than the number or numbers to be encrypted
2. K₁ < K₂

A variable α with its value being in the range of 0.1 and 0.9 is taken to be used in due course of time to shift the sequence of encrypted set of integers. Sections 2.1 and section 2.2 respectively describe the encryption and the decryption processes with conventionally Alice being considered as the sender and Bob being considered as the receiver.

2.1 Encryption

The set of computations to be done in the encryption end for P being a positive integer to be encrypted is as follows:

1. Set K₃ = φ (K₁) – 1.
2. Set A = Chosen random number from the set of numbers, which when used as quotient to divide K₂, leaves P as remainder.
3. Set B = A^K₃ mod K₁
4. Set C = Product of B and K₂
5. Set \( D = C^{K_3} \mod K_1 \)
6. Set \( E = \) Result of shifting \( D \) by \( \alpha \) %, which is the encrypted form for \( P \).

The function \( \varphi \) (\( K_1 \)) is a totient function [3]. The totient \( \varphi \) (\( K_1 \)) of a positive integer \( K_1 \) defines the number of integers less than or equal to \( K_1 \) that are relatively prime to \( K_1 \). \( \varphi \) (\( K_1 \)) = \( K_1 - 1 \), when \( K_1 \) is a prime number.

2.2 Decryption

Based on the set of values for \( K_1, K_2, \alpha \) extracted from the private key, the following set of operations are carried out in the decryption end.

1. Set \( K_3 = \varphi \) (\( K_1 \)) – 1.
2. Set \( D = \) Result of shifting \( E \) by \( (1- \alpha)\% \)
3. Set \( B = (D \cdot K_2)^{K_3} \mod K_1 \)
4. Set \( A = B^{K_3} \mod K_1 \)
5. Set \( P = K_2 \mod A \)

The function \( \varphi \) (\( K_1 \)) is a totient function as was described in section 2.1.

3. Implementation

This section describes the process of encryption and decryption using a sample tiny plaintext WORLD considering \( K_1 = 263, K_2 = 317 \) and \( \alpha = 0.1 \).

Table 1 enlists different intermediate and the final result of implementation following the encryption algorithm stated in section 2.1. Consequently, the cipher text “\( U \leq a^{\frac{1}{4}} E \)” is generated for the taken plaintext.

A reverse process is followed to employ the decryption algorithm stated in section 2.2.

4. Results

The implementation of the encryption / decryption algorithm described in section 2.1 and section 2.2 respectively was accomplished using C language and MIRACL (Multi precision Integer and Rational Arithmetic C/C++ Library) library [4] for big integer calculations. The plaintext to be encrypted was read as blocks of 8-bits, a pair of private keys KEY1 and KEY2, each 10-bits long, and a small value \( \alpha \) was taken for operation on the plaintext data.

Ciphertext data produced after encryption process was written as blocks of 32-bits i.e. size of integer type variable, to avoid overflow, as in some cases 8-bit data after encryption is to yield a value greater than 255. We have encrypted and also successfully decrypted file types .txt, .doc, .docx, .xls, .pdf, .mp4, .ocx, .jpg, .bmp, .dll, .gif, .sys etc.

A comparison of file type, source file size, decrypted file size, encryption time and decryption time is depicted in the table 2.

It is seen from data in Table 2 that the relation between encryption time and file size is linear. The average time, based on tabulated data in table 2, to encrypt 1 KB of data is remarkably found to be 0.028852538 seconds. Similarly, it was noted that the average time required for decryption of 1 KB of data is 0.007198586 seconds.

Each 8-bit source data is written as 32-bit encrypted data, so the encrypted data file size (in bytes) is approximately four times the source data file size (in bytes). Thus it was observed that the proposed cipher carries a schematic overhead of causing data expansion. Having computational complexities, an operational overhead of being marginally slow in execution is observed during its real implementation.

Figure 1 shows plotting of source file size (in KB) and encryption time (in Sec).
The average time required for such extensive search is given in Table 3 [5].

| Key Size (Bits) | Number of Alternative Keys | Time Required at 1 Encryption/Key | Time Required at 10^6 Encryption/Key |
|----------------|-----------------------------|------------------------------------|--------------------------------------|
| 56             | $2^{7.2 \times 10^9}$      | $1 142$ years                      | 10.01 hours                          |
| 128            | $2^{3.3 \times 10^9}$      | $5.4 \times 10^9$ years           | $5.4 \times 10^8$ years             |
| 256            | $2^{3.2 \times 10^{10}}$   | $5.9 \times 10^9$ years           | $5.9 \times 10^8$ years             |

Data expansion noted during encryption is expected as the proposed cipher uses exponentiation. We can adopt to write encrypted data as bit stream and device an appropriate padding scheme to identify separate blocks to reduce the encrypted file size considerably. The computational complexity of calculating large integer powers of integers can be simplified by adopting faster methods like exponentiation by squaring, square and multiply or binary exponentiation, which leads to reduction of both encryption and decryption time. Encryption process can be augmented to be faster if read block size (in bits) is increased but it should be less than the size of key (in bits). The proposed scheme restricts use of the read block size greater than the key size.

6. Conclusion

The proposed MABCVK is flexible enough for implementation with larger keys. The substitution of plaintext data with random integers produces different sets of encrypted data for the same private key pairs, which, in turn, makes the task of cryptanalysis a challenging one. Encryption and decryption using MABCVK require almost same execution time. If a pair of 128-bit keys is used for encryption and if brute-force attack is applied for decryption then at least all the primes in the range of integers from 1 to $2^{128}$ are to be tested as a probable key. There are $2^{128} / \log(2^{128}) = 38354127545935 \times 10^{12}$ prime numbers in the range of integers from 1 to $2^{128}$. It is observed that if one trillion of these numbers per second can be checked, more than $121,617,874,031,562 \times 10^3$ years is required to check all such primes. The requirement of time as shown in Table 3, to calculate the keys makes it reasonably non-vulnerable against cryptanalytic attacks. As a part of the future scope of the work, it is also intended to formulate and implement an enhanced scheme that will use a carrier image to have a secure transmission of the private keys.

References

[1] Menezes, P. van Oorschot, and S. Vanstone – “Handbook of Applied Cryptography”, CRC Press, 1996.
[2] Bruce Schneier - “Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C”, Publisher: John Wiley & Sons, Inc.

[3] Clay S. Turner, Euler's Totient Function and Public Key Cryptography - http://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/RSAmath.pdf.

[4] MIRACL Library web link: - http://www.shamus.ie

[5] Saurabh Dutta - An Approach to Design New Efficient Encryption Techniques, ISBN: 978-3-8443-0082-6, Lap Lambert Academic Publishing, Germany.

Abhijit Chowdhury, MCA, Assistant Professor, Department of Computer Applications – NSHM College of Management & Technology, Durgapur, West Bengal, INDIA. Member of IAENG (International Association of Engineers, http://www.iaeng.org). Research Area of interest is cryptography, network security.

Angshu Kumar Sinha, M.Sc.-Mathematics, Department of Computer Applications – NSHM College of Management & Technology, Durgapur, West Bengal, INDIA.

Dr. Saurabh Dutta, Ph.D. (Computer Science), Dr. B. C. Roy Engineering College, Durgapur-713206, West Bengal, INDIA.