# AES And Merkle-Hellman Knapsack Hybrid Cryptosystem

**Maricris C.Castro, Edwin R. Arboleda, Reynaldo R. Corpuz**

**Abstract**: To develop an enhanced cryptosystem by combining two existing algorithm is the objective of this study.The encryption process is composed of the Merkle-Hellman key generation combined with the AES sub-byte transformation that utilized an S-box. The strength of the proposed algorithm is provided by the extended Euclidian division of the Merkle-Hellman and the S-box of the AES algorithm. The proposed algorithm was tested using an example proving the encryption and decryption process.

**Index Terms**: AES, ciphertext, cryptography, decryption, encryption, hybrid encryption, Merkle-Hellman knapsack

## 1. INTRODUCTION

Cryptography is used in a communication system to to guarantee the information's integrity and confidentiality[1]. Cryptography comes from the word "kryptos" of the Greeks which means "secret writing"[2]. In the communication system comprising of the sender and receiver of the information, it is of utmost importance that no third party is intruding[3], [4]. The objective of the cryptosystem is keep the intruder out of the sender-receiver information exchanges and this is done by using encryption and decryption[5], [6]. In order for cryptography algorithm to be implemented the message in plaintext format is combined with a key or keys[7]. In the encryption process, the message containing information in the plaintext format from the sender is transformed into a format that is unintelligible to anyone except the intended recipient. In the decryption, the encrypted message is transformed again into a format that is intelligible again[8], [9]. Hybrid cryptosystems combines two or more cryptosystems, to combine the strengths of each[10]. Most hybrid cryptosystems incorporates the convenience of public cryptosystem and the security of the symmetric cryptosystems[5][4]. The result of combined, modified or enhanced cryptosystem by combining two or more cryptosystems are increase in security and improvement of overall system performance[11]–[15].

In 1999, two Belgians, Joan Daemen and Vincent Rijmen developed a symmetric encryption system known as the Advanced Encryption Standard (AES)[16].The AES used varying key lengths in encryption, that is AES-128, AES-192 or AES-256 [17], [18]. The Merkle-Hellman cryptosystem was developed by Ralph Merkle and Martin Hellman which uses a super increasing knapsack vector s. In this cryptosystem, modular multiplication and permutation is used to create a second vector M to hide the super increasing property. The public key is the vector M and vector s is for decyption of message[1], [19], [20].


- *Maricris C. Castro is from the Department of Computer and Electronics Egineering, College of Engineering and Information Technology, Cavite State University, Indang, Cavite. E-mail: sircimar27@gmail.com*
- *Edwin R. Arboleda is from the Department of Computer and Electronics Engineering, College of Engineering and Information Technology, Cavite State University. Indang, Cavite. E-mail: edwin.r.arboleda@cvsu.edu.ph*
- *Reynaldo R. Corpuz is an affiliate of Isabela State University.Cauayan Campus. Email: reynaldo.r.corpuz@isu.edu.ph*


## 2. METHODOLOGY

### 2.1 Proposed Algorithm

The proposed algorithm architecture of the encryption method is shown in Figure 1

#### 2.1.1 Key Generation

Preprocessing key generation

1. Convert the message to its 8-bit binary ASCII equivalent.
2. Get the gray code of the binary form of Mi, denoted as ki (k1, k2, k3….kn).

Merkle-Hellman key Generation

1. With an n-bit message
2. Choose ai : {a1 , a2 ,…, an } , a super increasing vector.
3. Choose a number p such that p > ∑ ai for 1 ≤ i ≤ n. p is called the modulus
4. Choose a number r such that r and q are coprime: gcd (r, p) = 1. r is called the multiplier.
5. Compute the vector bi : (b1 , b2 ,…, bn ) such that: bi = r ai mod(p) , 0 ≤ bi < p
6. Using the Extended Euclidean Division, compute the r-1 inverse of r modulo q

   Public key: is bi

   Private key: is (ai, p, and r)

#### 2.1.2 Encryption Process

Preprocessing:

1. Convert the message to its 8-bit binary ASCII equivalent.
2. XOR the 8-bit binary with the corresponding ki
3. Get the 1's complement, denoted as wi.

Using AES Sub Byte transformation:

1. Convert the 1's complement to hexadecimal, denoted as si
2. Apply the S-box to each s. (note: the first digit will be the row x, and the second digit will be the column y).

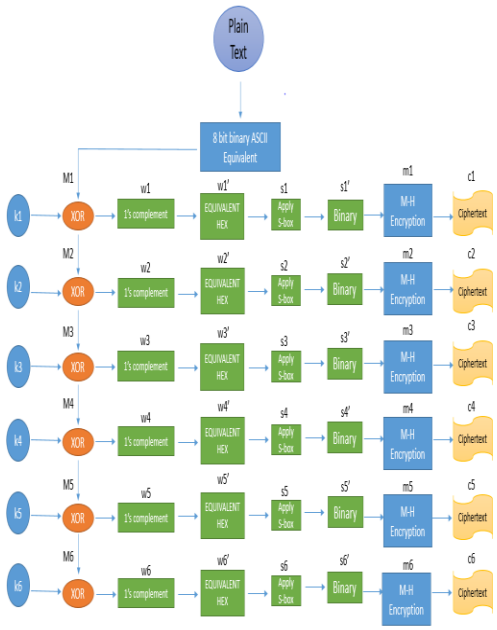


Figure 1. Encryption Process

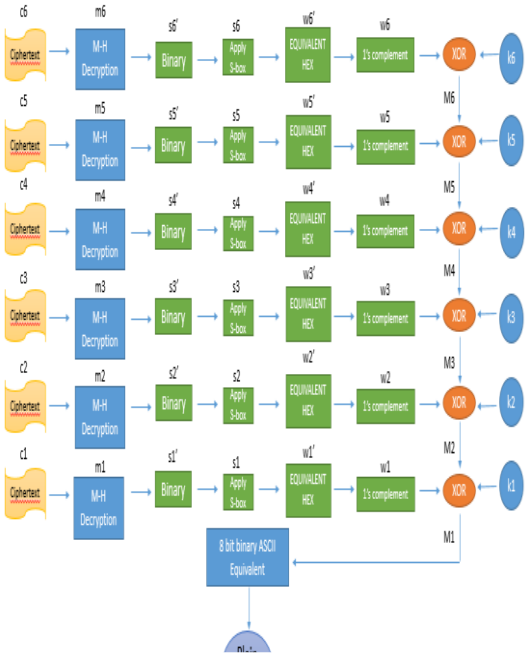


Figure 2. S-box

Merkle-Hellman Encryption:

1. n-bit message mi : { m1 , m2 ,…, mn }
2. Public key bi : {b1 , b2 ,…, bn }
3. Encrypted message is: c = ∑ mi bi (E) for 1 ≤ i ≤ n, with 0 ≤ c < p (E) is NP-Complete knapsack problem : bi is a hard-Knapsack

2.1.3. Decryption Process

Merkle-Hellman Decryption:

1. Private key: (ai, p, r).
2. Message integer c = ∑ mi bi for 1 ≤ i ≤ n.
3. Compute r-1 inverse of r modulo q using the Extended Euclidean Division
4. Solve the super increasing knapsack problem: for j = n down to 1 {If s ≥ ai then {xi = 1; s = s - ai ;} else xi = 0 ;} return (x1, x2... xn).

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 3B | 52 | D6 | B3 | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | **2A** | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

Figure 3. Decryption Process

Using AES inverse Sub Byte transformation:

1. Convert the binary to hexadecimal.
2. Apply the inverse S-box, denoted as w. (note: the first digit will be the row x, and the second digit will be the column y)
3. Covert to binary

Using the Decryption Final Process:

1. Get the 1's complement, denoted as wi'
2. XOR the 8-bit binary with the corresponding ki
3. Convert the 8-bit binary to its decimal equivalent then convert to its original form.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 09 | 6A | D5 | 30 | 36 | A5 | 38 | BF | 40 | A3 | 9E | 81 | F3 | D7 | FB |
| 1 | 7C | E3 | 39 | 82 | 9B | 2F | FF | 87 | 34 | 8E | 43 | 44 | C4 | DE | E9 | CB |
| 2 | 54 | 7B | 94 | 32 | A6 | C2 | 23 | 3D | EE | 4C | **95** | 0B | 42 | FA | C3 | 4E |
| 3 | 08 | 2E | A1 | 66 | 28 | D9 | 24 | B2 | 76 | 5B | A2 | 49 | 6D | 8B | D1 | 25 |
| 4 | 72 | F8 | F6 | 64 | 86 | 68 | 98 | 16 | D4 | A4 | 5C | CC | 5D | 65 | B6 | 92 |
| 5 | 6C | 70 | 48 | 50 | FD | ED | B9 | DA | 5E | 15 | 46 | 57 | A7 | 8D | 9D | 84 |
| 6 | 90 | D8 | AB | 00 | 8C | BC | D3 | 0A | F7 | E4 | 58 | 05 | B8 | B3 | 45 | 06 |
| 7 | D0 | 2C | 1E | 8F | CA | 3F | 0F | 02 | C1 | AF | BD | 03 | 01 | 13 | 8A | 6B |
| 8 | 3A | 91 | 11 | 41 | 4F | 67 | DC | EA | 97 | F2 | CF | CE | F0 | B4 | E6 | 73 |
| 9 | 96 | AC | 74 | 22 | E7 | AD | 35 | 85 | E2 | F9 | 37 | E8 | 1C | 75 | DF | 6E |
| A | 47 | F1 | 1A | 71 | 1D | 29 | C5 | 89 | 6F | B7 | 62 | 0E | AA | 18 | BE | 1B |
| B | FC | 56 | 3E | 4B | C6 | D2 | 79 | 20 | 9A | DB | C0 | FE | 78 | CD | 5A | F4 |
| C | 1F | DD | A8 | 33 | 88 | 07 | C7 | 31 | B1 | 12 | 10 | 59 | 27 | 80 | EC | 5F |
| D | 60 | 51 | 7F | A9 | 19 | B5 | 4A | 0D | 2D | E5 | 7A | 9F | 93 | C9 | 9C | EF |
| E | A0 | E0 | 3B | 4D | AE | 2A | F5 | B0 | C8 | EB | BB | 3C | 83 | 53 | 99 | 61 |
| F | 17 | 2B | 04 | 7E | BA | 77 | D6 | 26 | E1 | 69 | 14 | 63 | 55 | 21 | 0C | 7D |

Figure 3. Inverse S-box

## 3. RESULTS AND DISCUSSION

This implementation shows how the proposed algorithm works. The proposed algorithm is consists of AES and Merkle-Hellman knapsack.

Message: MANILA

***Table 1.*** *Plaintext Equivalent in Decimal and Binary*

| PLAIN TEXT | ASCII CODE: DECIMAL | ASCII CODE: BINARY |
|---|---|---|
| M | 77 | 01001101 |
| A | 65 | 01000001 |
| N | 78 | 01001110 |
| I | 73 | 01001001 |
| L | 76 | 01001100 |
| A | 65 | 01000001 |

Key Generation:

Preprocessing :

1. M1 = 01001101
   M2= 01000001
   M3= 01001110
   M4= 01001001
   M5= 01001100
   M6= 01000001

2. k1 = 01110110
   k2 = 01111110
   k3 = 01110100
   k4 = 01110001
   k5 = 01110111
   k6 = 01111110

Using Merkle-Hellman:

1. n = 8 bit binary
2. ai = { 1, 3, 7, 13, 26, 65, 119, 267} , i = 501
3. p = 523
4. r = 467
5. Using Euclidean algorithm $r^{-1} = 28$
   $467 \times r^{-1} \bmod 523$

Step 1: Euclidean Algorithm

523x + 467y = 1
523 = 1(467) + 56
467 = 8(56) + 19
56 = 2(19) + 18
19 = 1(18) +1

Step 2: Extended Euclidean Algorithm (Back Substitution)

1 = 19 – 1(18)
1 = 19 – 1[56 – 2(19)]
1 = 3(19) – 1(56)
1 = 3[467 – 8(56)] – 1(56)
1 = 3(467) – 24(56) – 1(56)
1 = 3(467) – 25(56)
1 = 3(467) – 25[523 – 1(467)]
1= 3(467) -25(523) + 25(467)
1= 28(467) - 25(523)

Since 28 is a positive number, we calculate
$r^{-1} = 28 \bmod 523$
$r^{-1} = 28$

6. bi1 = 467 x 1 mod(523) = 467
   bi2 = 467 x 3 mod(523) = 355
   bi3 = 467 x 7 mod(523) = 131
   bi4 = 467 x 13 mod(523) = 318
   bi5 = 467 x 26 mod(523) = 113
   bi6 = 467 x 65 mod(523) = 21
   bi7 = 467 x 119 mod(523) = 135
   bi8 = 467 x 267 mod(523) = 215

Public Key: bi { 467, 355, 131, 318, 113, 21, 135, 215 }
Private Keys : ai { 1, 3, 7, 13, 26, 65, 119, 267}
p= 523 r= 467

ENCRYPTION

Preprocessing:

1. M1 = 01001101 k1 = 01110110
   M2 = 01000001 k2 = 01111110
   M3 = 01001110 k3 = 01110100
   M4 = 01001001 k4 = 01110001
   M5 = 01001100 k5 = 01110111
   M6 = 01000001 k6 = 01111110

2. M1 ⊕ k1 = 01001101 ⊕ 01110110 = 00111011
   M2 ⊕ k2 w1 = 01000001 ⊕ 01111110 = 00111111
   M3 ⊕ k3 w2 = 01001110 ⊕ 01110100 = 00111010
   M4 ⊕ k4 w3 = 01001001 ⊕ 01110001 = 00111000
   M5 ⊕ k5 w4 = 01001100 ⊕ 01110111 = 00111011
   M6 ⊕ k6 w5 = 01000001 ⊕ 01111110 = 00111111

3. w1 = 11000100
   w2 = 11000000
   w3 = 11000101
   w4 = 11000111
   w5 = 11000100
   w6 = 11000000

Using AES inverse Sub Byte transformation:

1. w1' = 11000100 = c4
   w2' = 11000000 = c0
   w3' = 11000101 = c5
   w4' = 11000111 = c7
   w5' = 11000100 = c4
   w6' = 11000000 = c0

2. s1 = 1c
   s2 = ba
   s3 = a6
   s4 = c6
   s5 = 1c
   s6 = ba

3. s1' = 1c = 00011100
   s2' = ba = 10111010
   s3' = a6 = 10100110
   s4' = c6 = 11000110
   s5' = 1c = 00011100
   s6' = ba = 10111010

Using Merkle-Hellman Encryption:

1. s1' = 00011100
   s2' = 10111010
   s3' = 10100110
   s4' = 11000110
   s5' = 00011100
   s6' = 10111010

2. bi { 467, 355, 131, 318, 113, 21, 135, 215 }
   m1 = 318+113+21 = 452
   m2 = 467+131+318 +113+135 = 1,164
   m3 = 467 + 131 + 21 + 135 = 754
   m4 = 467 + 355 + 21 + 135 = 978
   m5 = 318 + 113 + 21 = 452
   m6 = 467 + 131 + 318 + 113 + 135 = 1,164

3. c1 = 452
   c2 = 1,164
   c3 = 754
   c4 = 978
   c5= 452
   c6 = 1,164

DECRYPTION

Using Merkle-Hellman Decryption:

1. c1' = 452(28) mod 523 = 104
   c2' = 1,164 (28) mod 523 = 166
   c3' = 754 (28) mod 523 = 192
   c4' = 978 (28) mod 523 = 188
   c5' = 452 (28) mod 523 = 104
   c6' = 1,164 (28) mod 523 = 166

2. ai { 1, 3, 7, 13, 26, 65, 119, 267}
   c1'= 104 – 65 = 39
   39 – 26 = 13
   13 – 13 = 0
   s1' = 00011100
   c2'= 166 – 119 = 47
   47 – 26 = 21
   21 – 13 = 8
   8 – 7 = 1
   1-1= 0
   s2' = 10111010
   c3'= 192 – 119 = 73
   73 – 65 = 8
   8 – 7 = 1
   1 – 1 = 0
   s3' = 10100110
   c4'= 188 – 119 = 73
   69 – 65 = 4
   4 – 3 = 1
   1 – 1 = 0
   S4' = 11000110
   c5'= 104 – 65 = 39
   39 – 26 = 13
   13 – 13 = 0
   s5' = 00011100
   c2'= 166 – 119 = 47
   47 – 26 = 21
   21 – 13 = 8
   8 – 7 = 1
   1 – 1 =0
   s6' = 10111010

Using AES inverse Sub Byte transformation:

1. s1' = 00011100 = 1c
   s2' = 10111010 = ba
   s3' = 10100110 = a6
   s4' = 11000110 = c6
   s5' = 00011100 = 1c
   s6' = 10111010 = ba

2. s1 = 1c = c4
   s2 = ba = c0
   s3 = a6 = c5
   s4 = c6 = c7
   s5 = 1c = c4
   s6 = ba = c0

3. w1' = c4 = 11000100
   w2' = c0 = 11000000
   w3' = c5 = 11000101
   w4' = c7 = 11000111
   w5' = c4 = 11000100
   w6' = c0 = 11000000

Decryption Final Process:

1. w1' = 11000100 = 00111011
   w2' = 11000000 = 00111111
   w3' = 11000101 = 00111010
   w4' = 11000111 = 00111000
   w5' = 11000100 = 00111011
   w6' = 11000000 = 00111111

2. w1 = w1' ⊕ k1
   = 00111011⊕01110110 = 01001101

   w2 = w2' ⊕ k2
   = 00111111⊕ 01111110 = 01000001

   w3 = w1' ⊕ k3
   =00111010⊕01110100 = 01001110

   w4 = w1' ⊕ k4
   =00111000⊕01110001= 01001001

   w5 = w1' ⊕ k5
   = 00111011⊕01110111 = 01001100

   w6 = w1' ⊕ k6
   =00111111⊕01111110 = 01000001

3. M1 = 01001101 = 77
   M2= 01000001 = 65
   M3= 01001110 = 78
   M4= 01001001 = 73
   M5= 01001100 = 76
   M6= 01000001 = 65

4. 77 = M
   65 = A
   78 = N
   73 = I
   76 = L
   65 = A

## 4. CONCLUSION

In this study, a new algorithm of selected asymmetric and symmetric encryption systems was created by combing the AES and Merkle-Hellman knapsack. First the key generation of this algorithm structure requires few keys. The encryption and decryption process does not require too much computation . As a whole, this new algorithm is as easy as to do. It does not require difficult computation. The developed hybrid algorithm's security resulted from combining the strengths of Merkle-Hellman Knapsack's extended Eucledian division and AES's S-box utilization.

## REFERENCES


[1] M. Thangavel and P. Varalakshmi, "A Novel Public Key Cryptosystem based on Merkle – Hellman Knapsack Cryptosystem," in 2016 IEEE Eighth International Conference on Advanced Computing (ICoAC) A, 2016, pp. 119–122.

[2] B. A. Forouzan and S. C. Fegan, Data Communications and Networking, 4th ed. Huga Media, 2007, 2000.

[3] E. R. Arboleda, C. E. R. Fenomeno, and J. Z. Jimenez, "KED-AES algorithm : combined key encryption decryption and advance encryption standard algorithm," vol. 8, no. 1, pp. 44–53, 2019.

[4] M. Enriquez, D. W. Garcia, and E. Arboleda, "Enhanced Hybrid Algorithm of Secure and Fast Chaos-based , AES , RSA and ElGamal Cryptosystems," Indian J. Sci. Technol., vol. 10, no. July, 2017.

[5] J. M. B. Espalmado and E. R. Arboleda, "DARE Algorithm : A New Security Protocol by Integration of Different Cryptographic Techniques," Int. J. Elecrical Comput. Eng., vol. 7, no. 2, pp. 1032–1041, 2017.

[6] E. R. Arboleda, "Secure and Fast Chaotic El Gamal Cryptosystem," Int. J. Eng. Adv. Technol., vol. 8, no. 5, pp. 1693–1699, 2019.

[7] Jhoanne Kris P. Alegro, E. R. Arboleda, M. R. Pereña, and R. M. Dellosa, "Hybrid Schnorr, RSA, and AES Cryptosystem," Int. J. Sci. Technol. Res., vol. 8, no. 10, pp. 1777–1781, 2019.

[8] R. R. Corpuz, B. D. Gerardo, and R. P. Medina, "Using a Modified Approach of Blowfish Algorithm for Data Security in Cloud Computing," in Proceedings of the 6th International Conference on Information Technology: IoT and Smart City, 2018, pp. 157–162.

[9] R. R. Corpuz and B. D. Gerardo, "A Modified Approach of Blowfish Algorithm Based On S- Box Permutation using Shuffle Algorithm," in Proceedings of the 2018 VII International Conference on Network, Communication and Computing, 2018, pp. 140–145.

[10] L. B. De Guzman and A. M. Sison, "Implementation of Enhanced MD5 Algorithm using SSL to Ensure Data Integrity," in Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, 2019, pp. 71–75.

[11] T. F. G. Quilala, A. M. Sison, and R. P. Medina, "Modified Blowfish Algorithm," Indones. J. Electr. Eng. Comput. Sci., vol. 12, no. 1, pp. 38–45, 2018.

[12] R. E. J. Paje and A. M. Sison, "Multidimensional Key RC6 Algorithm," in ICCSP 2019-Proceedings of the 3rd International Conference on Cryptography, Security and Privacy, 2019, no. July.

[13] R. B. Antonio and A. M. Sison, "A Modified Generation of S-Box for Advanced Encryption Standards," in PICISS 2019- Proceedings of the 2019 2nd International Conference on Information Science and Systems, 1999, pp. 280–283.

[14] G. L. Dulla and B. D. Gerardo, "An Enhanced BlowFish ( eBf ) Algorithm for Securing x64FileMessage Content," in 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM), 2018, pp. 1–6.

[15] G. L. Dulla, B. D. Gerardo, and R. P. Medina, "A unique message encryption technique based on enhanced blowfish algorithm A unique message encryption technique based on enhanced blowfish algorithm," in The International Conference on Information Technology and Digital Applications, 2019, pp. 1–12.

[16] J. Daemen, J. Daemen, V. Rijmen, V. Rijmen, and K. U. Leuven, "AES Proposal : Rijndael Authors :," The Rinjndael Block Cipher, p. 45, 1999.

[17] A. Nadjia and A. Mohamed, "AES IP for Hybrid Cryptosystem RSA-AES," in 2015 12th International Multi-Conference on Systems, Signals & Devices AES, 2015, pp. 1–6.

[18] N. Mathur and R. Bansode, "AES Based Text Encryption Using 12 Rounds with Dynamic Key," in Procedia - Procedia Computer Science, 2016, vol. 79, pp. 1036–1043.

[19] B. Padhmavathi, A. Ral, A. Anjum, and S. Bhat, "Improvement ofCBC Encryption Technique by Using the Merkle-Hellman Knapsack Cryptosystem B. Padhmavathi1, Arghya Ral, Alisha Anjum3 and Santhoshi Bhat4," in 2013 7th International Conference on Intelligent Systems and Control (ISCO), pp. 1–5.

[20] S. N. Sinha, S. Palit, M. A. Molla, A. Khanra, and M. Kule, "A Cryptanalytic Attack on Knapsack Cipher Using Differential Evolution Algorithm," in 2011 IEEE Recent Advances in Intelligent Computational Systems, 2011, pp. 317–320.