Ensuring Cloud Data Security Using the Soldier Ant Algorithm

John Kwao Dawson*, Ben Beklisi Kwame Ayawli, Sylvester Agyemang, Philemon Baah, and Samuel Akyeramfo-Sam

Sunyani Technical University, Sunyani, Ghana; Email: bbkaywli@stu.edu.gh, revsly@stu.edu.gh, pbaah@stu.edu.gh, atosam@stu.edu.gh
*Correspondence: Kwaodawson@stu.edu.gh, kwaodawson1@yahoo.com

***Abstract*—Cloud computing is evolving as a firsthand archetype of large-scale distributed computing which adds more power to internet technologies. It is a framework aiming at convenient and on-request network access to an organized and shared pool of resources. Security of data transmitted and stored by a third party happens to be the greatest challenge for organizations embracing the technology. In this work, a proposed hybrid algorithm dubbed the Soldier Ant Algorithm (SAA) a blend of the Diffie-Hellman algorithm and Delta Encoding technique (Newton Forward and Backward Interpolation). The motivation obtained is the integration of algorithms for cryptography purposes. The integration makes the proposed SAA algorithm symmetric and also makes the Diffie-Hellman algorithm withstand security threats such as man-in-the-middle attacks. The proposed algorithm establishes a secured connection between the cloud client and the cloud service provider and at the same time secures the data sent to the cloud. A comparative analysis was performed against RSA and ElGamal and indicated that the proposed algorithms' encryption and decryption time were lower even though it is linear (O(n)).**

***Keywords*—Diffie-Hellman, delta encoding, data in transmission, dictionary attack, soldier ant algorithm**

I. Introduction

Cloud computing is a model for enabling helpful, an on-request system that enables organizations to access a common pool of configurable resources that can be quickly made available with insignificant managerial effort through cloud-provider interaction [1]. Cloud computing is a rising and progressive approach to computing and it is becoming more risk-inclined [2]. It is a transformative approach to utilizing resources and services on request and according to the needs of the client [3]. Cloud computing is given a platform on the Internet for use of Information Technology services and adaptable infrastructure by clients and businesses [4]. The control and management of organizations' resources and assets in terms of data are at the mercy of a third party. In a cloud computing environment, there are a lot of associated benefits but security should be provided for both identified and unforeseen interferences that can corrupt and delete essential data by employing appropriate cryptographic schemes [5−7]. A proposed model of security concerns in the cloud is shown in Fig. 1 [8].

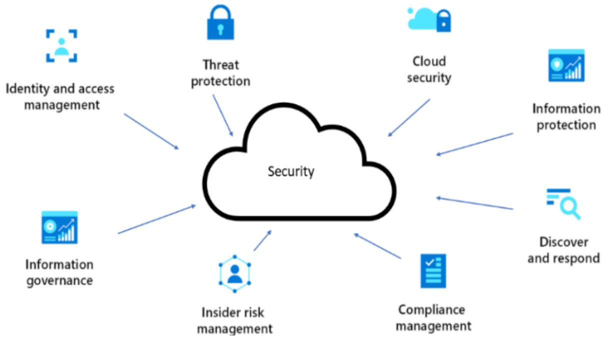


Figure 1. Proposed model of security concerns in the cloud.

The principle of the Diffie-Hellman algorithm is to secure channels for communication excluding the data under transmission or storage making it prone to attacks [9]. This implies that the data does not undergo encryption or decryption processes. Diffie-Hellman is appropriate for use in information communication, however, it is less regularly utilized for information storage [10]. Diffie-Hellman algorithm is a mathematical algorithm that permits two computers to create an indistinguishable shared secret on both systems, even though those systems may never have communicated with each other before. That mutual secret key can then be utilized to safely exchange a cryptographic encryption key. The generated key then encodes the data in traffic between the two systems [11]. This does not make the data to be transmitted to achieve the security advantage because the Diffie-Hellman algorithm only secures the communication channel instead of the data.

The will of many organizations to move onto the cloud has increased due to the advantages associated with it. Some of these include scalability, reduced capital expenditure, and pay-as-you-use services. Security is the slagging problem limiting an organization's full adoption of the cloud. There has been a series of research to propose variants of the Diffie-Hellman algorithm to ensure data security on the cloud. These variants have high execution

Manuscript received June 14, 2022; revised August 12, 20232; accepted October 11, 2022; published February 23, 2023.

doi: 10.12720/jait.14.1.130-137

time with linear time complexity. This, therefore, raises the need to provide a more robust security scheme to achieve the security of cloud data with low execution but linear time complexity. This paper proposes an algorithm that has lower execution time and linear time complexity. The proposed algorithm ensures the security of data in the cloud by integrating Diffie-Hellman algorithm and Delta encoding scheme.

II. Related Work

A considerable number of researchers have proposed varied cryptographic algorithms generated toward guaranteeing cloud information secrecy and protection. Amongst them is the work of Digra and Sharma [12] that proposed an enhanced Diffie-Hellman algorithm to secure the internet for the safe transmission of data. The present best procedure for attacking Diffie-Hellman depends on compromising one of the private keys by computing the discrete log of the related public value of the Diffie-Hellman scheme. As said over, the parameters to generate it are beforehand shared in clear as part of the Finite Field Cryptography (FFC) domain parameters. Be that as it may, an adversary who performs a large pre-calculation for a prime p can then quickly compute arbitrary discrete logs in that group, amortizing the cost overall targets that share this parameter. This proposed scheme still had a high execution rate with linear time complexity.

Diffie and Hellman [13] identified that Diffie-Hellman’s key agreement protocol usage has been attacked by genuine security defects. The attack can be exceptionally subtle and, as a rule, has not been considered by protocol designers. Byun and Lee [14] proposed two password-based Diffie-Hellman key exchange protocols which are claimed to be provably secure based on Diffie-Hellman problems. For simplicity of description, they referred to the two protocols as the EKE-U and EKE-M protocols, following the notation used in [14]. Byun and Lee claimed that the protocols are secure against dictionary attacks, especially insider dictionary attacks. However, it is noted that the EKE-U protocol suffers from offline dictionary attacks, and the EKE-M protocol suffers from undetectable online dictionary attacks which can be mounted by any malicious participant.

To contribute to addressing the challenges of data encryption and the establishment of secured connections, Naik *et al.* [15] proposed an algorithm that integrates the Elliptic curve and the Diffie-Hellman algorithm. Although their results indicated high efficiency concerning data encryption, the introduction of the Elliptic curve increased the computational cost in a situation involving a high amount of data.

Dabhad [16] proposed an algorithm for Data Security in the Cloud using the Aggregate Key and Diffie-Hellman Algorithm. They employed an efficient, simple, and publicly verifiable scheme which helps to ensure the security of data and also time-sharing between multiple users. However, their proposed scheme only works with text but could not support images since images do not deal with the constant size of Ciphertext.

To contribute to addressing the man-in-the-middle attack, Khaldi [17] proposed a symmetric algorithm to generate a 128-bit key without a digital signature. The proposed scheme on the other hand was not able to work on other image formats other than bitmap images [17]. According to Alam, the work of Nanli failed to eliminate man-in-the-middle attacks as such proposed a third-party authentication mechanism. This can secure the communication channel against any hacker. Their scheme even though is strong to secure communication channels could not secure data in storage or transit [18].

Khari *et al.* [19] proposed the use of layered architecture to ensure data security through identification and verification. Their approach employed SHA and ERSA to secure the upload of data and 3DES to ensure data security. The security scheme supported all forms of cloud models. Despite the security strength, they proposed the use of other hybrid algorithms to ensure data security.

Aikins-Bekoe and Ben [20] proposed the integration of the Diffie-Hellman algorithm and the Elliptic curve to get the Elliptic Curve Diffie-Hellman algorithm (ECDH). This is an advanced algorithm of the Traditional Diffie-Hellman algorithm. The ECDH functions based on the security of the key used but used large storage sizes and had linear time complexity.

To summarize the methodologies reviewed, they all pointed to ensuring data security of the cloud. The execution times of the reviewed methodologies had higher execution times with linear time complexity of O(n). There is, therefore, the need for a more robust security scheme with higher security, lower execution time, and linear time complexity. A proposed algorithm dubbed Soldier Ant Algorithm (SAA) is proposed in this study. The proposed algorithm integrates the Diffie-Hellman algorithm and Delta Encoding techniques.

III. Methodology

The methodology employed in this work is the integration of the Diffie-Hellman algorithm [10] and the Delta Encoding technique. The Delta encoding technique is a term used for the combination of Newton’s Forward and Backward Interpolation. The methodology separates an application into subsystems and each subsystem leads to a bigger system. In the proposed algorithm, the initial stage is the generation of the ASCII values for the various alphabets in the plaintext to be transmitted. The Newton Forward Interpolation is then applied to the ASCII values of the plaintext to encrypt it. The Diffie-Hellman algorithm is now employed to secure the communication channel between the cloud client and the cloud service provider. The Ciphertext is transmitted to the cloud service provider via the secured channel. To request outsourced data from the cloud service provider to the cloud client, the Diffie-Hellman algorithm is applied to secure the communication channel after which Newton Backward Interpolation [21, 22] is applied to the encrypted data to obtain the plaintext as shown in Fig. 2.

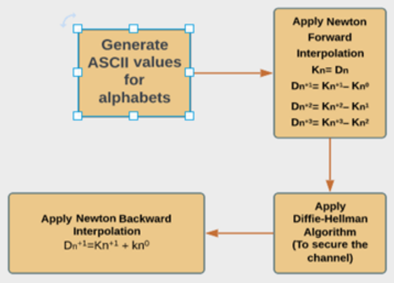

Generate ASCII values for alphabets
Apply Newton Forward Interpolation
Kn= Dn
Dn+1= Kn+1– Kn0
Dn+2= Kn+2– Kn1
Dn+3= Kn+3– Kn2
Apply Diffie-Hellman Algorithm (To secure the channel)
Apply Newton Backward Interpolation
Dn+1=Kn+1 + kn0

Figure 2. Workflow diagram of proposed algorithm (SAA).

A. Generate ASCII Value of Alphabets

Generate the ASCII values for the alphabet in plaintext to be secured and transmitted to the cloud. For example, A is 65, and B is 66.

B. Newton's Forward Interpolation Formula (Delta Encoding Scheme)

Using the changes of $y_1 - y_0, y_2 - y_1, y_3 - y_2$. The difference between y_1 and y_0 is Newton's First Forward Interpolation [15]. The initial ASCII value is maintained as the first Ciphertext for the encryption process as shown in Eq. (1).

$$K_n = D_n \tag{1}$$

The subsequent ASCII values are deducted from the initial by applying the formula in Eq. (2). The formula in Eqs. (2−4) are used to compute the first and second-level Ciphertext.

$$D_n^{+1} = K_n^{+1} - K_n^{0} \tag{2}$$

$$D_n^{+2} = K_n^{+2} - K_n^{1} \tag{3}$$

$$D_n^{+3} = K_n^{+3} - K_n^{2} \tag{4}$$

For example, Table I indicates the ASCII values for the word HELLO.

TABLE I. ASCII VALUES FOR HELLO

H	E	L	L	O
72	69	76	76	79

By applying Newton Forward Interpolation on the plaintext to obtain the Ciphertext for the given plaintext, apply Eqs. (2−4) as indicated in Table II.

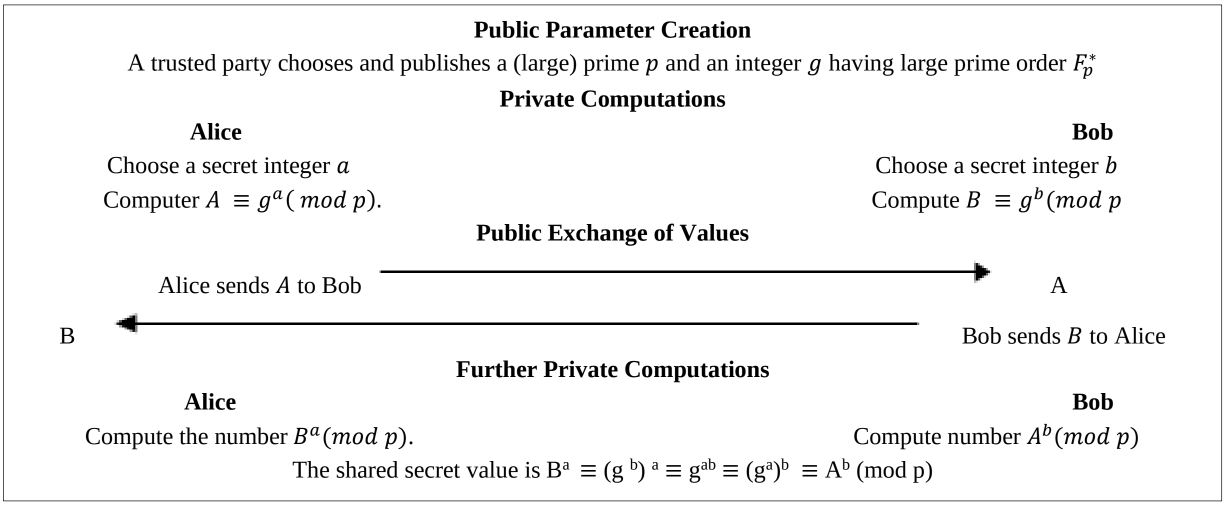
TABLE II. CIPHERTEXT FOR HELLO

72	69	76	76	79
72	−3	7	0	3

C. Apply Diffie-Hellman Algorithm

The Diffie-Hellam algorithm is used to establish a secured connection between two devices to share a secret key over an unsecured channel [23]. The Diffie-Hellman algorithm is used to establish the link between the cloud client and the cloud service provider as depicted in Algorithms 1 and 2.

In a Diffie-Hellman algorithm, if Bob and Alice want to communicate, they do so using a public key system. There is a key exchange between Bob and Alice as shown in Fig. 3.



Public Parameter Creation

A trusted party chooses and publishes a (large) prime p and an integer g having large prime order F_p^*

Private Computations

Alice

Choose a secret integer a

Computer $A \equiv g^a(\,mod\ p)$.

Bob

Choose a secret integer b

Compute $B \equiv g^b(mod\ p$

Public Exchange of Values

Alice sends A to Bob

A

B

Bob sends B to Alice

Further Private Computations

Alice

Compute the number $B^a(mod\ p)$.

Bob

Compute number $A^b(mod\ p)$

The shared secret value is $B^a \equiv (g^{\,b})^{\,a} \equiv g^{ab} \equiv (g^a)^b \equiv A^b$ (mod p)

Figure 3. Diffie-Hellman algorithm computation.

Fig. 3 explains the Diffie-Hellman algorithm indicating how data is exchanged between Alice and Bob over an unsecured channel. The users, Alice and Bod have the same B^a and A^B values indicating a secured connection. As a result, the secret key is shared between the two users which becomes authentic for that session only [19]. The users can communicate again provided a new prime number with its primitive root is required, which gives security in the Diffie-Hellman algorithm.

The Diffie-Hellman algorithm is used in the proposed algorithm to secure the connection between the cloud client and the cloud service provider. This is attained when $B^a(mod\ p)$ and $A^b(mod\ p)$ are the same depicted in Fig. 3. The Ciphertext is then transmitted via the secured channel.

D. Newton's Backward Interpolation Formula (Delta Encoding Scheme)

With the given points $y_0, y_1, y_2, \dots, y_n$ applying the function $\mathcal{F}$ using $[y_0, y_n]$.

Considering the points $y_0, y_1, y_2, \dots, y_n$ which are equidistant to each other. This implies that

$$k = Ya^{+1} - y_1,$$

where $a = 0, 1, 2, 3, \dots, a-1$ [22, 23].

This can be identified in the finite order backward difference using the function $\mathcal{F}$ relating to a as shown in Eq. (5).

$$\nabla\mathcal{F}\,(a) = \mathcal{F}_{a+1} - \mathcal{F}_a \tag{5}$$

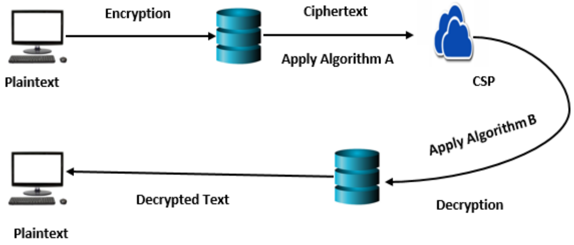


Figure 4. Proposed SAA architecture.

The framework for the proposed algorithm is shown in Fig. 4. The plaintext for the cloud storage is converted to Ciphertext. The ciphertext is then sent through the Internet Service provider's network for cloud storage. On request for the ciphertext from the cloud service provider, the decryption algorithm is applied to the ciphertext to convert it to plaintext for onward forwarding to the recipient.

E. Encryption Process for the Proposed Soldier Ant Algorithm (SAA)

The user after creating an account with the cloud service provider outsources its data to the cloud. Apply Newton Forward Interpolation on the ASCII values.

The algorithm to establish a secured connection and the conversion of plaintext to ciphertext is shown in Algorithm 1.

Algorithm 1: Encryption Approach and Secured Connection Between Client and Cloud Provider

```
Input:    Plaintext (P)
Output: Ciphertext (D)
    Initialization: a = alphabets of plaintext), K =ASCII values of a, D=Cipher text, $x = \{x:\mathbb{Z}\}$, $d = \{d:\mathbb{Z}\}$, $f = \{f:\mathbb{Z}\}$, $n = (1,2,3,\dots)$
    $P = (a_1, a_2, a_2, \dots, a_n)$ //Alphabets of plain text
    $\forall K_n = ASCII(a_n)$ //Find the ASCII value for each $a_n$
    $y = prime(n)$ // Select prime numbers of n
    $v = x^d mod\ y$ //Computer and send to the cloud client
    $j = x^f mod\ y$ //Send to the cloud client
    $m = j^d mod\ y$ //Executes by the cloud service provider
    $s = m^d mod\ y$ //Cloud provider executes
    if $m == s$ {
        $D_1 = K_1$ //Use initial $K_n$ as cipher text
        $D_{n+1} = K_{n+1} - K_n$ //
        Send($D_1, D_{n+1}$)
    }
```

To obtain the plaintext from the ciphertext uploaded to the cloud service provider, Algorithm 2 is applied. A secured connection is established between the cloud client and the cloud service provider using Eq. (6).

$$l = g^d mod\ p \tag{6}$$

where g is the value sent to the cloud client, d is the secret integer selected by the cloud client, and p is the prime number selected. The result obtained from using Eq. (6) is sent to the cloud service provider. The cloud service provider now computes the value q by using Eq. (7).

$$q = v^d mod\ p \tag{7}$$

The value for v is the base value selected, d is the random number selected, and p is the prime number. The result from Eq. (6) and now compared with the results from Eq. (7). When $l = q$ then a secured connection is established after which Newton Backward Interpolation is applied to the ciphertext to obtain the ASCII values for the plaintext as shown in Eqs. (8−11) and depicted in Algorithm 2.

$$K_n = D_n \tag{8}$$

$$D_n^{+1} = K_n^{+1} - K_n^0 \tag{9}$$

$$D_n^{+2} = K_n^{+2} - K_n^1 \tag{10}$$

$$D_n^{+3} = K_n^{+3} - K_n^2 \tag{11}$$

Algorithm 2: Decryption Approach and Secured Connection Between Client and Cloud Provider

```
Input:    Ciphertext (D)
Output: Plaintext (P)
    Initialization: $p = \{p: prime\ numbers\}$, K =ASCII values of a, D=Cipher text, $v = \{v:\mathbb{Z}\}$, $d = \{d:\mathbb{Z}\}$, $f = \{f:\mathbb{Z}\}$, $n = (1,2,3,\dots)$
    $m = v^d mod\ p$ //Send to cloud service provider
    $g = v^f mod\ p$ //Send to the cloud client
    $l = g^d mod\ p$ //Executes by cloud service
    $q = v^d mod\ p$ //Cloud provider executes
    if $l == q$ {
        $K_1 = D_1$ // Newton Backward interpolation for initial $K_1$
        $K_{n+1} = D_{n-1} + D_n$ //Newton Backward interpolation for $K_{n+1}$
        $P = toString(K_n, K_{n+1})$ //
        Send(P)
    }
```

F. Proof of Proposed Algorithm against Man-in-the-Middle Attack

If user A wants to send a message to user B, without being intercepted by the man in the middle.

STAGE 1: APPLY NEWTON FORWARD INTERPOLATION (DELTA ENCODING SCHEME) ON ASCII VALUES.

Generate the ASCII values for the message to be transmitted. For example, if the message to be transmitted is HELLO.

$$ASCII\ VALUES = 7269767679$$

$$CIPHERTEXT = 72 - 3703$$

STAGE 2: APPLY DIFFIE–HELLMAN ALGORITHM TO SECURE CONNECTION BETWEEN CLOUD CLIENT AND CLOUD SERVICE PROVIDER.

Three random numbers are selected by user A. Equation 12 is used to compute the private key to be sent to user B.

$$Q = 19, X_a = 10, P_a = 3 \tag{12a}$$

$$Y_a = 3^{10}\ mod\ 19 \tag{12b}$$

$$Y_a = 16 \tag{12c}$$

$$Y_aA = 16 \tag{12d}$$

The results for equation 12d are intercepted by the attacker and saved.

Stage 2 is composed of five phases which are discussed as follows:

Phase1: Computation by Attacker

A random number X is generated by the ATTACKER. Using equation 13b compute the private key to be sent by the ATTACKER to user B. The final value for Y_{malB} is then sent to user B.

$$X_{malB} = 2 \tag{13a}$$

$$Y_{malB} = 3^2\ mod\ 19 \tag{13b}$$

$$Y_{malB} = 9 \tag{13c}$$

Phase 2: User B Computation

User B selects an integer. Upon receiving the private key from the ATTACKER, user B computes the private key to be sent to user A which is intercepted by the ATTACKER as shown in equation 14. The result from user B is then acknowledged by the ATTACKER as user B to establish a secured connection using equations 15a and 16a. After the computation, the value for X_B is sent to user A.

$$X_B = 11 \tag{14a}$$

$$X_B = 3^{11} mod\ 19 \tag{14b}$$

$$X_B = 10 \tag{14c}$$

$$K_B = 9^{11}\ \mathrm{mod}\ 19 \tag{15a}$$

$$K_B = 5 \tag{15b}$$

Phase 3: Attacker Intercept

$$K_B = \ 10^2\ mod\ 19 \tag{16a}$$

$$K_B = 5 \tag{16b}$$

Phase 4: Computation by Attacker

A secured connection is now established between the ATTACKER and user B. Now a secured connection needs to be established between user A and the ATTACKER. This is computed using equations 17a, 18a, and 19a to achieve the secured connection between user A and the ATTACKER as shown in Fig. 5.

The attacker generates a random integer

$$X_{malA} = \ 3^7 mod\ 19 \tag{17a}$$

$$X_{malA} = 2 \tag{17b}$$

$$K_A = 16^7 mod\ 19 \tag{18a}$$

$$K_A = 17 \tag{18b}$$

Phase 5: Values from Attacker To user A

Y_A Private Key

$$K_A = \ 2^{10} mod\ 19 \tag{19a}$$

$$K_A = 17 \tag{19b}$$

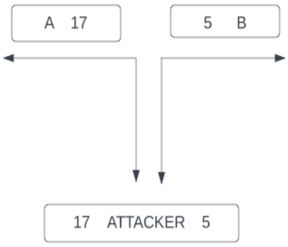


Figure 5. A secured connection is established with the man–in-the middle.

$$C_1 = E(17{,}65) \tag{20}$$

- **65 the ASCII value for character A**

$$C_1 = 1105$$

$$P_1 = D(17, 1105) \tag{21}$$

$$P_1 = 65$$

$$C_2 = E(5{,}65) \tag{22}$$

$$C_2 = 325$$

$$P_1 = (5, 325) \tag{23}$$

$$P = 65$$

The attacker can decrypt the message from user A and forward the modified message to user B based on the private keys computed for users A and B using Eqs. (20−23).

On the other hand, with the integration of Delta Encoding Techniques with the Diffie-Hellman algorithm, the security of the data is increased since the encryption and decryption are not dependent on the private key generated by users A and B intercepted by the attacker.

STAGE 3: APPLY NEWTON BACKWARD INTERPOLATION TO DECRYPT THE MESSAGE

$$CIPHERTEXT: 72 - 3303$$

$$PLAINTEXT: 7269767679$$

IV. Implementation, Results, and Discussion

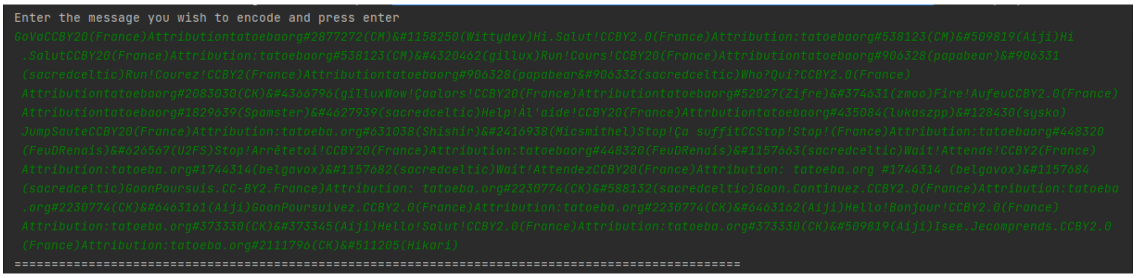
Implementation/Environmental Setup for the Experiment

The implementation of the proposed Soldier Ant Algorithm (SAA) is presented in this section. The SAA is an integration of the Diffie-Hellman algorithm and Delta Encoding scheme. This section details the environmental setup for the proposed algorithm, the validation of the method used in the proposed algorithm, the results of the proposed algorithm, and the comparison with existing algorithms.

To validate the experiment, a simulation was conducted on an i7 Lenovo computer, a 2.10GHz CPU, and implemented using the C# language. The ASCII values for the plaintext were generated and a secured connection is established between the cloud client and the cloud service provider. The encrypted text which is not based on the private values generated by the cloud client and the cloud service providers is then transmitted to the cloud service provider as depicted in Figs. 6−10. A predesigned dataset from the Kaggle database [24] was used for the experimental work and a comparative analysis was run against RSA [25] using data size applied in the works of Ali *et al.* [25].

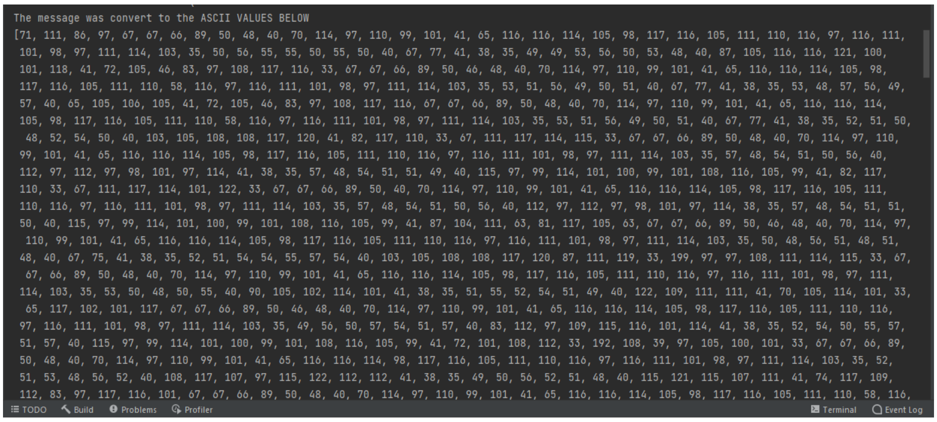
V. Results of the Proposed Algorithm

There was a comparison of the performance metrics of three cryptographic schemes on their encryption and decryption time. The algorithms considered were ElGamal [26], RSA [25], and the proposed algorithm (Soldier Ant Algorithm) using data sizes indicated in Table III as used in the works of Sann *et al.* [27]. The results of the simulations are depicted in Tables III and IV.



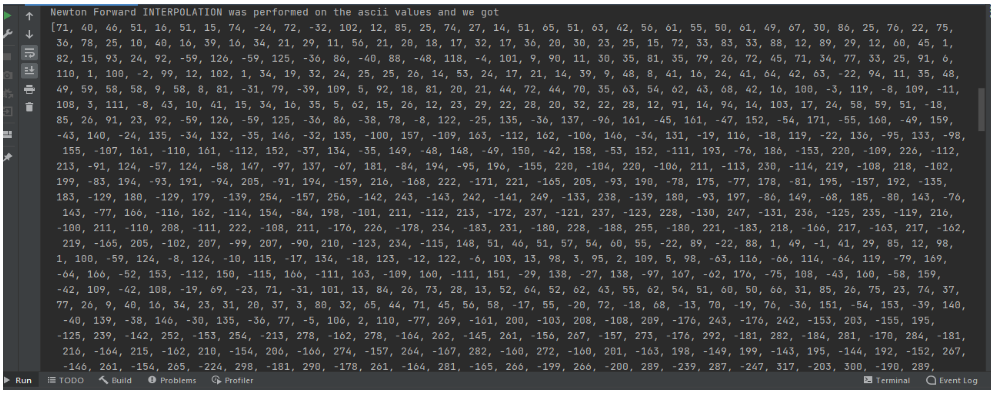
```
Enter the message you wish to encode and press enter
GoVaCCBY20(France)Attributiontatoebaorg#2877272(CM)&#1158250(Wittydev)Hi.Salut!CCBY2.0(France)Attribution:tatoebaorg#538123(CM)&#509819(Aiji)Hi
 .SalutCCBY20(France)Attribution:tatoebaorg#538123(CM)&#4320462(gillux)Run!Cours!CCBY20(France)Attributiontatoebaorg#906328(papabear)&#906331
 (sacredceltic)Run!Courez!CCBY2(France)Attributiontatoebaorg#906328(papabear&#906332(sacredceltic)Who?Qui?CCBY2.0(France)
 Attributiontatoebaorg#2083030(CK)&#4366796(gilluxWow!Çaalors!CCBY20(France)Attributiontatoebaorg#52027(Zifre)&#374631(zmoo)Fire!AufeuCCBY2.0(France)
 Attributiontatoebaorg#1829639(Spamster)&#4627939(sacredceltic)Help!Àl'aide!CCBY20(France)Attrbutiontatoebaorg#435084(lukaszpp)&#128430(sysko)
 JumpSauteCCBY20(France)Attribution:tatoeba.org#631038(Shishir)&#2416938(Micsmithel)Stop!Ça suffitCCStop!Stop!(France)Attribution:tatoebaorg#448320
 (FeuDRenais)&#626567(U2FS)Stop!Arrêtetoi!CCBY20(France)Attribution:tatoebaorg#448320(FeuDRenais)&#1157663(sacredceltic)Wait!Attends!CCBY2(France)
 Attribution:tatoeba.org#1744314(belgavox)&#1157682(sacredceltic)Wait!AttendezCCBY20(France)Attribution: tatoeba.org #1744314 (belgavox)&#1157684
 (sacredceltic)GoonPoursuis.CC-BY2.France)Attribution: tatoeba.org#2230774(CK)&#588132(sacredceltic)Goon.Continuez.CCBY2.0(France)Attribution:tatoeba
 .org#2230774(CK)&#6463161(Aiji)GoonPoursuivez.CCBY2.0(France)Attribution:tatoeba.org#2230774(CK)&#6463162(Aiji)Hello!Bonjour!CCBY2.0(France)
 Attribution:tatoeba.org#373330(CK)&#373345(Aiji)Hello!Salut!CCBY2.0(France)Attribution:tatoeba.org#373330(CK)&#509819(Aiji)Isee.Jecomprends.CCBY2.0
 (France)Attribution:tatoeba.org#2111796(CK)&#511205(Hikari)
==========================================================================================
```

Figure 6. Plaintext to be encrypted.



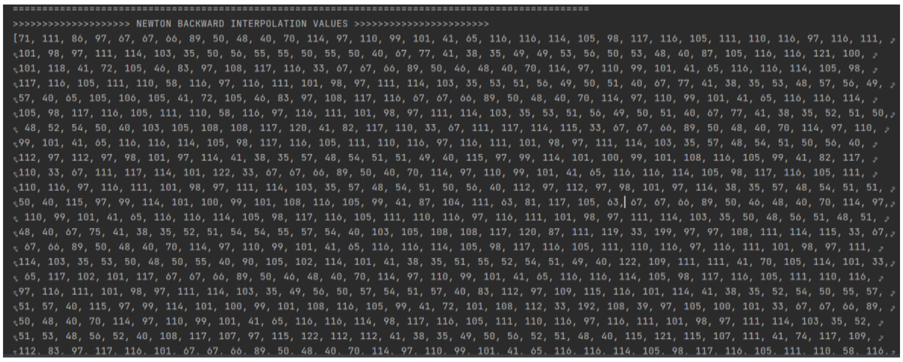
```
The message was convert to the ASCII VALUES BELOW
[71, 111, 86, 97, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111,
 101, 98, 97, 111, 114, 103, 35, 50, 56, 55, 55, 50, 55, 50, 40, 67, 77, 41, 38, 35, 49, 49, 53, 56, 50, 53, 48, 40, 87, 105, 116, 116, 121, 100,
 101, 118, 41, 72, 105, 46, 83, 97, 108, 117, 116, 33, 67, 67, 66, 89, 50, 46, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98,
 117, 116, 105, 111, 110, 58, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 53, 51, 56, 49, 50, 51, 40, 67, 77, 41, 38, 35, 53, 48, 57, 56, 49,
 57, 40, 65, 105, 106, 105, 41, 72, 105, 46, 83, 97, 108, 117, 116, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114,
 105, 98, 117, 116, 105, 111, 110, 58, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 53, 51, 56, 49, 50, 51, 40, 67, 77, 41, 38, 35, 52, 51, 50,
  48, 52, 54, 50, 40, 103, 105, 108, 108, 117, 120, 41, 82, 117, 110, 33, 67, 111, 117, 114, 115, 33, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110,
 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 57, 48, 54, 51, 50, 56, 40,
 112, 97, 112, 97, 98, 101, 97, 114, 41, 38, 35, 57, 48, 54, 51, 51, 49, 40, 115, 97, 99, 114, 101, 100, 99, 101, 108, 116, 105, 99, 41, 82, 117,
 110, 33, 67, 111, 117, 114, 101, 122, 33, 67, 67, 66, 89, 50, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111,
 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 57, 48, 54, 51, 50, 56, 40, 112, 97, 112, 97, 98, 101, 97, 114, 38, 35, 57, 48, 54, 51, 51,
 50, 40, 115, 97, 99, 114, 101, 100, 99, 101, 108, 116, 105, 99, 41, 87, 104, 111, 63, 81, 117, 105, 63, 67, 67, 66, 89, 50, 46, 48, 40, 70, 114, 97,
  110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 50, 48, 56, 51, 48, 51,
 48, 40, 67, 75, 41, 38, 35, 52, 51, 54, 54, 55, 57, 54, 40, 103, 105, 108, 108, 117, 120, 87, 111, 119, 33, 199, 97, 97, 108, 111, 114, 115, 33, 67,
  67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111,
 114, 103, 35, 53, 50, 48, 50, 55, 40, 90, 105, 102, 114, 101, 41, 38, 35, 51, 55, 52, 54, 51, 49, 40, 122, 109, 111, 111, 41, 70, 105, 114, 101, 33,
  65, 117, 102, 101, 117, 67, 67, 66, 89, 50, 46, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116,
 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 49, 56, 50, 57, 54, 51, 57, 40, 83, 112, 97, 109, 115, 116, 101, 114, 41, 38, 35, 52, 54, 50, 55, 57,
 51, 57, 40, 115, 97, 99, 114, 101, 100, 99, 101, 108, 116, 105, 99, 41, 72, 101, 108, 112, 33, 192, 108, 39, 97, 105, 100, 101, 33, 67, 67, 66, 89,
 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 52,
 51, 53, 48, 56, 52, 40, 108, 117, 107, 97, 115, 122, 112, 112, 41, 38, 35, 49, 50, 56, 52, 51, 48, 40, 115, 121, 115, 107, 111, 41, 74, 117, 109,
 112, 83, 97, 117, 116, 101, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 58, 116,
```

Figure 7. ASCII values of plaintext.



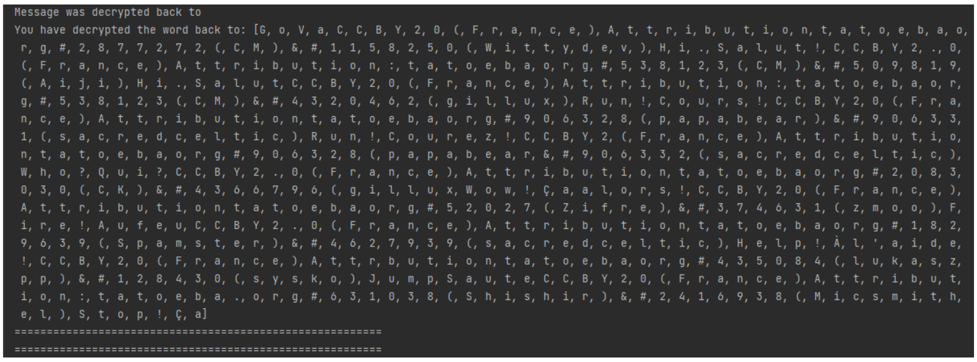
```
Newton Forward INTERPOLATION was performed on the ascii values and we got
[71, 40, 46, 51, 16, 51, 15, 74, -24, 72, -32, 102, 12, 85, 25, 74, 27, 14, 51, 65, 51, 63, 42, 56, 61, 55, 50, 61, 49, 67, 30, 86, 25, 76, 22, 75,
 36, 78, 25, 10, 40, 16, 39, 16, 34, 21, 29, 11, 56, 21, 20, 18, 17, 32, 17, 36, 20, 30, 23, 25, 15, 72, 33, 83, 33, 88, 12, 89, 29, 12, 60, 45, 1,
 82, 15, 93, 24, 92, -59, 126, -59, 125, -36, 86, -40, 88, -48, 118, -4, 101, 9, 90, 11, 30, 35, 81, 35, 79, 26, 72, 45, 71, 34, 77, 33, 25, 91, 6,
 110, 1, 100, -2, 99, 12, 102, 1, 34, 19, 32, 24, 25, 25, 26, 14, 53, 24, 17, 21, 14, 39, 9, 48, 8, 41, 16, 24, 41, 64, 42, 63, -22, 94, 11, 35, 48,
 49, 59, 58, 58, 9, 58, 8, 81, -31, 79, -39, 109, 5, 92, 18, 81, 20, 21, 44, 72, 44, 70, 35, 63, 54, 62, 43, 68, 42, 16, 100, -3, 119, -8, 109, -11,
 108, 3, 111, -8, 43, 10, 41, 15, 34, 16, 35, 5, 62, 15, 26, 12, 23, 29, 22, 28, 20, 32, 22, 28, 12, 91, 14, 94, 14, 103, 17, 24, 58, 59, 51, -18,
 85, 26, 91, 23, 92, -59, 126, -59, 125, -36, 86, -38, 78, -8, 122, -25, 135, -36, 137, -96, 161, -45, 161, -47, 152, -54, 171, -55, 160, -49, 159,
 -43, 140, -24, 135, -34, 132, -35, 146, -32, 135, -100, 157, -109, 163, -112, 162, -106, 146, -34, 131, -19, 116, -18, 119, -22, 136, -95, 133, -98,
  155, -107, 161, -110, 161, -112, 152, -37, 134, -35, 149, -48, 148, -49, 150, -42, 158, -53, 152, -111, 193, -76, 186, -153, 220, -109, 226, -112,
 213, -91, 124, -57, 124, -58, 147, -97, 137, -67, 181, -84, 194, -95, 196, -155, 220, -104, 220, -106, 211, -113, 230, -114, 219, -108, 218, -102,
 199, -83, 194, -93, 191, -94, 205, -91, 194, -159, 216, -168, 222, -171, 221, -165, 205, -93, 190, -78, 175, -77, 178, -81, 195, -157, 192, -135,
 183, -129, 180, -129, 179, -139, 254, -157, 256, -142, 243, -143, 242, -141, 249, -133, 238, -139, 180, -93, 197, -86, 149, -68, 185, -80, 143, -76,
  143, -77, 166, -116, 162, -114, 154, -84, 198, -101, 211, -112, 213, -172, 237, -121, 237, -123, 228, -130, 247, -131, 236, -125, 235, -119, 216,
 -100, 211, -110, 208, -111, 222, -108, 211, -176, 226, -178, 234, -183, 231, -180, 228, -188, 255, -180, 221, -183, 218, -166, 217, -163, 217, -162,
  219, -165, 205, -102, 207, -99, 207, -90, 210, -123, 234, -115, 148, 51, 46, 51, 57, 54, 60, 55, -22, 89, -22, 88, 1, 49, -1, 41, 29, 85, 12, 98,
 1, 100, -59, 124, -8, 124, -10, 115, -17, 134, -18, 123, -12, 122, -6, 103, 13, 98, 3, 95, 2, 109, 5, 98, -63, 116, -66, 114, -64, 119, -79, 169,
 -64, 166, -52, 153, -112, 150, -115, 166, -111, 163, -109, 160, -111, 151, -29, 138, -27, 138, -97, 167, -62, 176, -75, 108, -43, 160, -58, 159,
 -42, 109, -42, 108, -19, 69, -23, 71, -31, 101, 13, 84, 26, 73, 28, 13, 52, 64, 52, 62, 43, 55, 62, 54, 51, 60, 50, 66, 31, 85, 26, 75, 23, 74, 37,
 77, 26, 9, 40, 16, 34, 23, 31, 20, 37, 3, 80, 32, 65, 44, 71, 45, 56, 58, -17, 55, -20, 72, -18, 68, -13, 70, -19, 76, -36, 151, -54, 153, -39, 140,
  -40, 139, -38, 146, -30, 135, -36, 77, -5, 106, 2, 110, -77, 269, -161, 200, -103, 208, -108, 209, -176, 243, -176, 242, -153, 203, -155, 195,
 -125, 239, -142, 252, -153, 254, -213, 278, -162, 278, -164, 262, -145, 261, -156, 267, -157, 273, -176, 292, -181, 282, -184, 281, -170, 284, -181,
  216, -164, 215, -162, 210, -154, 206, -166, 274, -157, 264, -167, 282, -160, 272, -160, 201, -163, 198, -149, 199, -143, 195, -144, 192, -152, 267,
  -146, 261, -154, 265, -224, 298, -181, 290, -178, 261, -164, 281, -165, 266, -199, 266, -200, 289, -239, 287, -247, 317, -203, 300, -190, 289,
```

Figure 8. Apply Newton forward interpolation.



```
==========================================================================================
>>>>>>>>>>>>>>>>>>>> NEWTON BACKWARD INTERPOLATION VALUES >>>>>>>>>>>>>>>>>>>>>>>
[71, 111, 86, 97, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111,
101, 98, 97, 111, 114, 103, 35, 50, 56, 55, 55, 50, 55, 50, 40, 67, 77, 41, 38, 35, 49, 49, 53, 56, 50, 53, 48, 40, 87, 105, 116, 116, 121, 100,
101, 118, 41, 72, 105, 46, 83, 97, 108, 117, 116, 33, 67, 67, 66, 89, 50, 46, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98,
117, 116, 105, 111, 110, 58, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 53, 51, 56, 49, 50, 51, 40, 67, 77, 41, 38, 35, 53, 48, 57, 56, 49,
57, 40, 65, 105, 106, 105, 41, 72, 105, 46, 83, 97, 108, 117, 116, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114,
105, 98, 117, 116, 105, 111, 110, 58, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 53, 51, 56, 49, 50, 51, 40, 67, 77, 41, 38, 35, 52, 51, 50,
 48, 52, 54, 50, 40, 103, 105, 108, 108, 117, 120, 41, 82, 117, 110, 33, 67, 111, 117, 114, 115, 33, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110,
99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 57, 48, 54, 51, 50, 56, 40,
112, 97, 112, 97, 98, 101, 97, 114, 41, 38, 35, 57, 48, 54, 51, 51, 49, 40, 115, 97, 99, 114, 101, 100, 99, 101, 108, 116, 105, 99, 41, 82, 117,
110, 33, 67, 111, 117, 114, 101, 122, 33, 67, 67, 66, 89, 50, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111,
110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 57, 48, 54, 51, 50, 56, 40, 112, 97, 112, 97, 98, 101, 97, 114, 38, 35, 57, 48, 54, 51, 51,
50, 40, 115, 97, 99, 114, 101, 100, 99, 101, 108, 116, 105, 99, 41, 87, 104, 111, 63, 81, 117, 105, 63, 67, 67, 66, 89, 50, 46, 48, 40, 70, 114, 97,
 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 50, 48, 56, 51, 48, 51,
48, 40, 67, 75, 41, 38, 35, 52, 51, 54, 54, 55, 57, 54, 40, 103, 105, 108, 108, 117, 120, 87, 111, 119, 33, 199, 97, 97, 108, 111, 114, 115, 33, 67,
 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111,
114, 103, 35, 53, 50, 48, 50, 55, 40, 90, 105, 102, 114, 101, 41, 38, 35, 51, 55, 52, 54, 51, 49, 40, 122, 109, 111, 111, 41, 70, 105, 114, 101, 33,
 65, 117, 102, 101, 117, 67, 67, 66, 89, 50, 46, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 116,
97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 49, 56, 50, 57, 54, 51, 57, 40, 83, 112, 97, 109, 115, 116, 101, 114, 41, 38, 35, 52, 54, 50, 55, 57,
51, 57, 40, 115, 97, 99, 114, 101, 100, 99, 101, 108, 116, 105, 99, 41, 72, 101, 108, 112, 33, 192, 108, 39, 97, 105, 100, 101, 33, 67, 67, 66, 89,
50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 98, 117, 116, 105, 111, 110, 116, 97, 116, 111, 101, 98, 97, 111, 114, 103, 35, 52,
51, 53, 48, 56, 52, 40, 108, 117, 107, 97, 115, 122, 112, 112, 41, 38, 35, 49, 50, 56, 52, 51, 48, 40, 115, 121, 115, 107, 111, 41, 74, 117, 109,
112, 83, 97, 117, 116, 101, 67, 67, 66, 89, 50, 48, 40, 70, 114, 97, 110, 99, 101, 41, 65, 116, 116, 114, 105, 98, 117, 116, 105, 111, 110, 58, 116,
```

Figure 9. Apply Newton backward interpolation on encrypted text.



```
Message was decrypted back to
You have decrypted the word back to: [G, o, V, a, C, C, B, Y, 2, 0, (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t, i, o, n, t, a, t, o, e, b, a, o,
 r, g, #, 2, 8, 7, 7, 2, 7, 2, (, C, M, ), &, #, 1, 1, 5, 8, 2, 5, 0, (, W, i, t, t, y, d, e, v, ), H, i, ., S, a, l, u, t, !, C, C, B, Y, 2, ., 0,
 (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t, i, o, n, :, t, a, t, o, e, b, a, o, r, g, #, 5, 3, 8, 1, 2, 3, (, C, M, ), &, #, 5, 0, 9, 8, 1, 9,
 (, A, i, j, i, ), H, i, ., S, a, l, u, t, C, C, B, Y, 2, 0, (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t, i, o, n, :, t, a, t, o, e, b, a, o, r,
 g, #, 5, 3, 8, 1, 2, 3, (, C, M, ), &, #, 4, 3, 2, 0, 4, 6, 2, (, g, i, l, l, u, x, ), R, u, n, !, C, o, u, r, s, !, C, C, B, Y, 2, 0, (, F, r, a,
 n, c, e, ), A, t, t, r, i, b, u, t, i, o, n, t, a, t, o, e, b, a, o, r, g, #, 9, 0, 6, 3, 2, 8, (, p, a, p, a, b, e, a, r, ), &, #, 9, 0, 6, 3, 3,
 1, (, s, a, c, r, e, d, c, e, l, t, i, c, ), R, u, n, !, C, o, u, r, e, z, !, C, C, B, Y, 2, (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t, i, o,
 n, t, a, t, o, e, b, a, o, r, g, #, 9, 0, 6, 3, 2, 8, (, p, a, p, a, b, e, a, r, &, #, 9, 0, 6, 3, 3, 2, (, s, a, c, r, e, d, c, e, l, t, i, c, ),
 W, h, o, ?, Q, u, i, ?, C, C, B, Y, 2, ., 0, (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t, i, o, n, t, a, t, o, e, b, a, o, r, g, #, 2, 0, 8, 3,
 0, 3, 0, (, C, K, ), &, #, 4, 3, 6, 6, 7, 9, 6, (, g, i, l, l, u, x, W, o, w, !, Ç, a, a, l, o, r, s, !, C, C, B, Y, 2, 0, (, F, r, a, n, c, e, ),
 A, t, t, r, i, b, u, t, i, o, n, t, a, t, o, e, b, a, o, r, g, #, 5, 2, 0, 2, 7, (, Z, i, f, r, e, ), &, #, 3, 7, 4, 6, 3, 1, (, z, m, o, o, ), F,
 i, r, e, !, A, u, f, e, u, C, C, B, Y, 2, ., 0, (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t, i, o, n, t, a, t, o, e, b, a, o, r, g, #, 1, 8, 2,
 9, 6, 3, 9, (, S, p, a, m, s, t, e, r, ), &, #, 4, 6, 2, 7, 9, 3, 9, (, s, a, c, r, e, d, c, e, l, t, i, c, ), H, e, l, p, !, À, l, ', a, i, d, e,
 !, C, C, B, Y, 2, 0, (, F, r, a, n, c, e, ), A, t, t, r, b, u, t, i, o, n, t, a, t, o, e, b, a, o, r, g, #, 4, 3, 5, 0, 8, 4, (, l, u, k, a, s, z,
 p, p, ), &, #, 1, 2, 8, 4, 3, 0, (, s, y, s, k, o, ), J, u, m, p, S, a, u, t, e, C, C, B, Y, 2, 0, (, F, r, a, n, c, e, ), A, t, t, r, i, b, u, t,
 i, o, n, :, t, a, t, o, e, b, a, ., o, r, g, #, 6, 3, 1, 0, 3, 8, (, S, h, i, s, h, i, r, ), &, #, 2, 4, 1, 6, 9, 3, 8, (, M, i, c, s, m, i, t, h,
 e, l, ), S, t, o, p, !, Ç, a]
========================================================
========================================================
```

Figure 10. Decrypted text.

TABLE III. COMPARING THE ENCRYPTION TIMES (SECONDS) OF RSA, ELGAMAL, AND SAA

File Sizes (KB)	RSA	ElGamal	SAA
64	54	5142	43
96	81	7713	75
150	126	12051	118
208	175	16711	143
298	250	23941	185

TABLE IV. COMPARING THE DECRYPTION TIMES (SECONDS) OF RSA, ELGAMAL, AND SAA

File Sizes (KB)	RSA	ElGamal	SAA
64	205	179	193
96	307	268	230
150	408	419	374
208	666	580	532
298	954	831	864

From Table III, it can be deduced that the encryption time for RSA is lower compared with the ElGamal algorithm but higher as compared with Soldier Ant Algorithm (SAA). The performance of SAA is lower when the data size was eased to 298 KB indicating the strength of SAA against RSA and ElGamal even though encryption times are proportional to the data sizes.

On the part of decryption, Table IV indicates that the decryption time is proportional to the data sizes. From Table IV, with a data size of 64KB, the decryption time for RSA is 205 seconds, ElGamal 179 seconds, and SAA 193 seconds. From, this it can be said that SAA's decryption performance is better even though its execution performance is linear. This is confirmed by the works of Masram *et al.* [26] and Ali *et al.* [25] that data sizes are proportional to execution times.

VI. CONCLUSION

In this work a hybrid algorithm is proposed which is strong and can resist security attacks of Diffie Hellman algorithm, being it dictionary attack, mathematical attack, timing attack, and Man in the Middle Attack. This paper presents the implementation of the Diffie-Hellman and Delta Encoding scheme using encryption and decryption procedures, which are readily available for commercial use. This algorithm is secure due to the encryption and decryption approaches which are generated secretly by the sender and receiver keys. Three-level of security are adopted which has made the proposed algorithm symmetric. This approach enhanced security for PaaS, SaaS, and IaaS on the level of transmission of messages, handling of information, and management of keys. The execution metrics of the proposed non-mathematical linear (O(n)) but its performance is lower which indicates the efficiency of a good cryptographic scheme.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

John Kwao Dawson conducted the research work; Philemon Baah and Agyemang Sylvester performed the simulation and comparison work. Mathematical works were performed by Ben Beklisi Kwame and proofreading by Samual Akyeramfo-Sam. All authors had approved the final version.

REFERENCES

[1] K. Palmgren, "Diffie-Hellman key exchange: A non-mathematical explanation," *The Global Voice of Information Security*, pp. 30−33, 2006.

[2] G. A. A. Aljarah, "Efficiency of using the Diffie-Hellman key in cryptography for internet security," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 6, pp. 2039−2044, Apr. 2021.

[3] M. Khari, M. Kumar, and Vaishali, "Comprehensive study of cloud computing and related security issues," *Advances in Intelligent Systems and Computing*, pp. 699−707, Oct. 2017.

[4] J. M. Vidal, A. L. S. Orozco, and L. J. G. Villalba, "Adaptive artificial immune networks for mitigating DoS flooding attacks," *Swarm and Evolutionary Computation*, vol. 38, pp. 94−108, Feb. 2018.

[5] A. Panwar, V. Bhatnagar, M. Khari, *et al.*, "A blockchain framework to secure Personal Health Record (PHR) in IBM cloud-based data lake," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1−19, Apr. 2022.

[6] R. Abid, C. Iwendi, A. R. Javed, *et al.*, "An optimized homomorphic CRT-RSA algorithm for secure and efficient communication," *Personal and Ubiquitous Computing*, Sep. 2021.

[7] N. Tirthani and R. Ganesan, "Data security in cloud architecture based on Diffie Hellman and elliptical curve cryptography," in *Proc. IACR Cryptol. ePrint Arch.*, 2014, p. 49.

[8] N. Ripa, "Analysis of Newton's forward interpolation formula," *International Journal of Computer Science and Emerging Technologies*, vol. 1, no. 4, pp. 2044−6004, 2010.

[9] A. Khaldi, "Diffie-Hellman key exchange through steganographied images," *Law, State and Telecommunications Review*, vol. 10, no. 1, pp. 147−160, May 2018.

[10] B. Alam, "Diffie-Hellman key exchange protocol with entities authentication," *International Journal of Engineering and Computer Science*, vol. 6, no. 4, Apr. 2017.

[11] Z. Trifunov, L. Zenku, and T. Jusufi-Zenku, "Application of Newton's backward interpolation using Wolfram Mathematica," *International Journal of Mathematics Trends and Technology*, vol. 67, no. 2, pp. 53−56, Feb. 2021.

[12] N. Digra and S. Sharma, "A novel approach of enhancing security in cloud using Diffie Hellman algorithm," *International Journal of Scientific Research and Management*, vol. 5, no. 7, Jul. 2017.

[13] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644−654, November 1976.

[14] J. W. Byun and D. H. Lee, "N-party encrypted Diffie-Hellman key exchange using different passwords," *Applied Cryptography and Network Security*, pp. 75−90, 2005.

[15] R. M. Naik, S. Sathyanarayana, and T. Sowmya, "Key management using elliptic curve diffie hellman curve 25519," in *Proc. Third International Conference on Multimedia Processing, Communication & Information Technology (MPCIT)*, 2020, pp. 33−39.

[16] M. S. K. Dabhade and M. K. Kshirsagar, "Data security in cloud using aggregate key and Diffie-Hellman algorithm," *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 4, pp. 906−923, 2015.

[17] B. Alam, "Diffie-Hellman key exchange protocol with entities authentication," *International Journal of Engineering and Computer Science*, vol. 6, no. 4, Apr. 2017.

[18] P. Priyadarshinee, R. D. Raut, M. K. Jha, *et al.*, "Understanding and predicting the determinants of cloud computing adoption: A two-staged hybrid SEM − Neural networks approach," *Computers in Human Behavior*, vol. 76, pp. 341−362, Nov. 2017.

[19] M. Khari, M. Kumar, and Vaishali, "Secure data transference architecture for cloud computing using cryptography algorithms," in *Proc. 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 2141−2146

[20] S. Aikins-Bekoe and J. Ben, "Elliptic Curve Diffie-Hellman (ECDH) analogy for secured wireless sensor networks," *International Journal of Computer Applications*, vol. 176, no. 10, pp. 1−8, Apr. 2020.

[21] F. Sun, S. He, X. Zhang, *et al.*, "A fully authenticated Diffie-Hellman protocol and its application in WSNs," in *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1986−1999, 2022. doi: 10.1109/TIFS.2022.3173536

[22] A. P. D. Camargo, "Backward and forward stability analysis of Neville's algorithm for interpolation and a pyramid algorithm for the computation of Lebesgue functions," *Numerical Algorithms*, vol. 89, no. 4, pp. 1521−1531, Jul. 2021.

[23] B. Das and D. Chakrabarty, "Newton's divided difference interpolation formula: Representation of numerical data by a polynomial curve," *International Journal of Mathematics Trends and Technology*, vol. 35, no. 3, pp. 197−203, Jul. 2016.

[24] English to French translations. [Online]. Available: https://www.kaggle.com/datasets/digvijayyadav/frenchenglish/metadata

[25] K. Ali, F. Akhtar, S. A. Memon, *et al.*, "Performance of cryptographic algorithms based on time complexity," in *Proc. 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2020, pp. 1−5.

[26] R. Masram, V. Shahare, J. Abraham, *et al.*, "Analysis and comparison of symmetric key cryptographic algorithms based on various file features," *International Journal of Network Security & Its Applications*, vol. 6, no. 4, pp. 43−52, Jul. 2014.

[27] Z. Sann, T. T. Soe, K. W. M. Knin, *et al.*, "Performance comparison of asymmetric cryptography (case study-mail message)," *APTIKOM Journal on Computer Science and Information Technologies*, vol. 4, no. 3, pp. 105−111, Jan. 2020.

John Kwao Dawson is a Ph.D. candidate in computer science at the Kwame Nkrumah University of Science and Technology. He holds a master of philosophy in information technology and a bachelor's in information technology from the Kwame Nkrumah University of Science and Technology and the University of Education Winneba, respectively. His area of research is cloud computing, algorithm design, machine learning, artificial intelligence, and network security.

Ben Beklisi Kwame Ayawli received the B.Ed. degree in information technology from the University of Education, Winneba, Ghana, in 2008, and the M.Sc. degree in information technology from Sikkim Manipal University, India, in 2011. He completed a Ph.D. in power engineering automation from Nanjing Tech University, Nanjing, China in 2019. He is currently a Senior Lecturer and the Head of Department of the Computer Science Department, at Sunyani Technical University, Ghana. He was the ICT Director of Sunyani Technical University, from 2013 to 2016. He is also a Web Application Developer. His research interests include data security, robot path planning and navigation, the internet of things, deep learning, data mining, and web applications.

Agyemang Sylvester is currently a lecturer in the Computer Science Department, at Sunyani Technical University in the Bono region of Ghana. Currently a Ph.D. candidate in Information Technology (IT) at Kwame Nkrumah University of Science and Technology (KNUST) in Ghana. He received his master's degree in information technology (IT) at the University of Cape Coast (UCC) in June 2014 and Bachelor of Education from the University of Education Winneba (UEU) in October 2012 and a Diploma from Berekum College of Education, Berekum Bono Region of Ghana. His areas of expertise are Cloud Computing, Distributed Network Systems, Network Configuration and Security, and Database Management Systems. He has teaching experience of more than ten (10) years with Ghana Education Service (GES) from basic level to tertiary institution.

Philemon Baah is a Ph.D. student in Mathematical Statistics at the Kwame Nkrumah University of Science and Technology. He holds a master of philosophy in applied mathematics and a Bachelor of Science degree in mathematics from the Kwame Nkrumah University of Science and Technology. His area of research is statistical learning.

Samuel Akyeramfo-Sam is a Ph.D. candidate in information technology at Kwame Nkrumah University of Science and Technology. He holds a master of education in information technology from the University of Cape Coast. A former head of the Computer Science Department at Sunyani Technical University. His area of research is artificial intelligence, data mining, and computer security.