

Symmetric Encryption using Artificial Neural Network, Binary Tree Traversal, and Interleaving Salting



Atishay Jain, Mihir Gupta, Vibhu Kumar Singh, Aju Dennisan

Abstract: We have proposed a novel way of symmetric encryption utilizing interleaving salting, Binary Tree traversal, and an Artificial Neural Network in a sequential manner. We have presented the use of an Artificial Neural Network as an invertible(reversible) mathematical function so that decryption of the encrypted output may be possible. Knowledge of the encryption pipeline, salt, and neural network weights is required for decryption. As the same set of values is required for encryption and decryption, our proposed approach is a type of symmetric-key algorithm. Each user will have a unique key. Thus if a key attributing to a particular user is compromised, the integrity of the data of the remaining users will still be maintained. Our approach can be utilized to encrypt text data such as messages, documents, and letters. The encryption process consists of interleaving salting, creation of binary tree by considering the input as its level order traversal, and passing the preorder and inorder traversal of the constructed binary tree as input to the Artificial Neural Network. The output of the Artificial Neural Network would be the encrypted data. Decryption would require determining the input of an Artificial Neural Network from the output, hence solving multiple sets of linear equations and constructing a binary tree from its preorder and inorder traversal. We have then analyzed the variation of performance with the change in the input string size. The codebase of our proposed approach is publicly available at <https://github.com/Atishaysjain/Symmetric-Encryption-using-AN-and-Binary-Trees>.

Keywords: Symmetric Encryption, Artificial Neural Network, Binary Tree, Salting.

I. INTRODUCTION

Encryption, in layman's terms, is a way of scrambling data such that only the intended receiver can understand the information. In technical terms, it transforms human-readable text into encoded text, i.e., ciphertext. Decryption, however, converts the encoded text back to its original form, i.e., plain text. This process is used to secure the information being transmitted. However, with the ever-evolving technology,

information security has become a significant security issue. With fast-upgrading technology in hardware acceleration techniques used to crack robust hashing and encryption algorithms, it is essential to be up-to-date to defend against these threats. Encryption is a two-way approach in which plaintext content is sent in as an input, and unintelligible ciphertext is returned. Since encryption is two-way, the data can be decrypted and read yet again. On the other hand, hashing [1] is a one-way process in which the plaintext is jumbled into a unique digest that cannot be decoded. We propose a symmetric cryptographic technique [2] to encrypt data. It is an encryption technique in which the sender and receiver of a message encrypt and decode messages using a single shared key. Each user is assigned a unique key, which they can share with all those to whom they want to give access to their encrypted data. Hence if two users, user A and user B, are communicating with each other using our encryption approach to encode their messages, both the users must have access to each other's key to decode one another's messages. If a key belonging to a given user is leaked, the integrity of the data of the remaining users will remain intact. Thus allocating different keys to each user will enhance data security and privacy.

In cryptography, salting [3] refers to appending or prepending random text to the plain text before processing it through the hash function. While salting decreases vulnerabilities significantly, it is safer not to use salting in a traditional sense. Instead, we use a salting mechanism inspired by jumbling salting [4] to set up an initial layer. The selection of the salt is random and is unique for all users. The idea is to randomize the encrypted text as much as possible without consuming too much memory or time. Numerous encryption algorithms have used data structures like graphs and maps [5]. Keeping this in mind, we have set up a subsequent layer of randomization using a binary tree data structure. Next, we employ an Artificial Neural Network(ANN) [6]. The novelty of our approach lies in the utilization of ANN as an encryption function rather than its traditional usage as a tool for prediction or classification.

II. LITERATURE REVIEW

The Traditional encryption algorithms used to encrypt passwords and sensitive information were asymmetric or public/private encryption [7], [8].

Manuscript received on 15 July 2022 | Revised Manuscript received on 22 July 2022 | Manuscript Accepted on 15 August 2022 | Manuscript published on 30 August 2022.

* Correspondence Author

Atishay Jain*, SCOPE, Vellore Institute of Technology, Vellore (Tamil Nadu), India. Email: atishay.jain2019@vitstudent.ac.in

Mihir Gupta, SCOPE, Vellore Institute of Technology, Vellore (Tamil Nadu), India. Email: mihir.gupta2019a@vitstudent.ac.in

Vibhu Kumar Singh, SCOPE, Vellore Institute of Technology, Vellore (Tamil Nadu), India. Email: vibhukumar.singh2019@vitstudent.ac.in

Aju Dennisan, SCOPE, Vellore Institute of Technology, Vellore (Tamil Nadu), India. Email: daju@vit.ac.in

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license <http://creativecommons.org/licenses/by-nc-nd/4.0/>

Symmetric Encryption using Artificial Neural Network, Binary Tree Traversal, and Interleaving Salting

Asymmetric encryption uses a pair of keys. Data was encrypted with one key and could be decrypted only with the other key. The concept of “signcryption” [9] was introduced to enhance the security of such algorithms. Signcryption was targeted to fulfill the functions of both digital signature and public-key encryption in one step. Symmetric key cryptography [10] is any cryptographic algorithm that depends on a shared key that is utilized to encrypt or decrypt data, as opposed to asymmetric key cryptography, where the encryption and decryption keys are different. Symmetric encryption is usually more efficient than asymmetric encryption and is therefore favored when a significant amount of information has to be traded. Some presently used symmetric key cryptographic techniques are DES [11] strategy, Double DES technique, and Play-fair technique. DES is a block cipher that encrypts information in blocks of 64 bits in length, which means 64 bits of unencrypted information go as the input to DES, producing 64 bits of encrypted information. For both encryption and decryption, the same algorithm and key are used, with minor differences. Similarly, double DES is a symmetric cryptographic approach that encrypts the message first using DES encryption and generates a key. Then this middle text is again encrypted using the DES encryption technique generating the second key. Both of these keys are required at the time of decryption. As a result, the security level is enhanced by adding this additional security layer. AES [10] data encryption is a more mathematically efficient and secure symmetric-key cryptography algorithm. Its key advantage lies in the option for various key lengths. AES cryptographic algorithm supports a 128-bit, 192-bit, or 256-bit key, which makes it exponentially more robust than the 56-bit key of DES. Structurally, DES uses the Feistel network, which divides the block into two halves before the encryption steps. In comparison, AES is a stream cipher that uses the permutation-substitution technique, which involves a series of substitution and permutation steps to create the encrypted block. Binary trees have been a part of many research articles for over half a century. Binary trees have been utilized for encryption for over two decades. Binary tree encryption proposed by [12] uses ASCII manipulation and binary tree traversal techniques. However, the hash produced by this algorithm can be traced back to the plain text. Salting [3] is an idea that is prominently used in password encryption. Salt is a unique set of characters that can be added to the end of the plain text to make an alternate hash output. Hence, salting helps in enhancing security, especially against brute force attacks. BCRYPT [13] is a key derivation function based on blowfish cipher. Moreover, BCRYPT is a time adaptive function: brute-force search attempts can be thwarted even as computing power increases by gradually raising the iteration count to make the algorithm slower. Another approach for building upon the concept of salting was jumbling salting [4]. The jumbling process selects characters from pre-defined character sets and adds them to the plain password using a mathematical modulus function.

III. METHODOLOGY

A. Salting Encryption

Instead of simply appending the salt at the end of the plaintext, salting is done in an interleaving fashion. In other

words, every second character of the resultant plaintext is a character from the salt. The first character in the resulting string is the first character of the plaintext, whereas the second character is the first character of the salt, and so on. If the salt is smaller than the plaintext, it is repeated until it is the same length as the plaintext. Fig. 1 illustrates the above explained process.

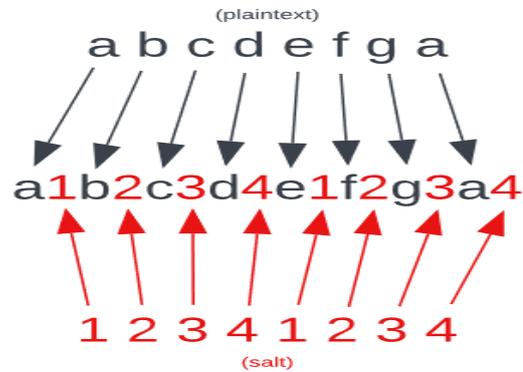


Fig. 1. Interleaving salting encryption on plain text: “abcdefga” using salt: “1234”.

B. Binary Tree Encryption

The next step deals with the duplicates in the resultant string so that it can be processed into the binary tree. First, the frequency of each character is noted. Then, a unique character '*' is added at the end of each repeated character to remove duplicates. That is, "aa" will be converted to "aa*" such that "a" and "a*" are two different characters. The resultant string is then considered as the level-order traversal of a binary tree. Subsequently, this traversal is used to build the binary tree. Fig. 2 illustrates the above explained process.

Finally, the inorder and preorder traversal of the binary tree is calculated and sent for further encryption as input to the ANN.

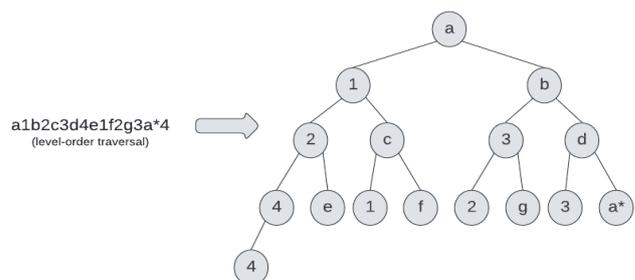


Fig. 2. Creation of binary tree by considering the string as its level-order traversal.

C. Encryption Function using Artificial Neural Network

The Output from the Binary Tree is then passed through an Artificial Neural Network(ANN). Fig. 3 illustrates the architecture of the ANN. We have utilized the neural network as an encryption function. Our inspiration to employ an ANN as an encryption function comes from the design of a Neural Network. ANN is a sequential connection of layered interconnected nodes.

Every connection between two nodes corresponds to a weight, i.e., a numerical value. The value of a node in a particular layer is the weighted sum of the values of the nodes in the previous layer, plus a bias.

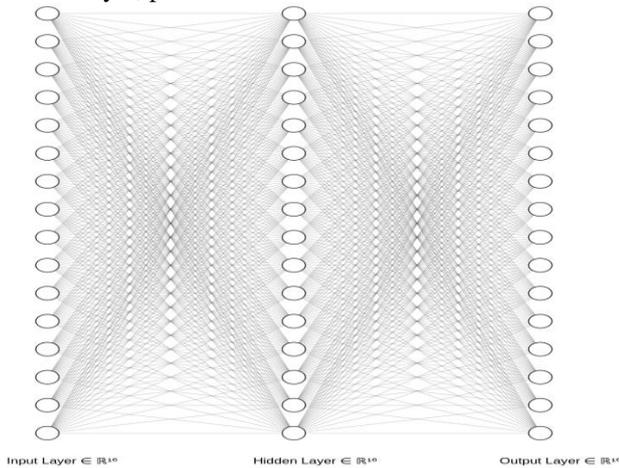


Fig. 3. Architecture of the Artificial Neural Network used.

Therefore the value of a particular node is essentially an output of a mathematical function that takes the values of the nodes present in the previous layer as input. The above statement can be explained with an example of an ANN depicted in fig. 4.

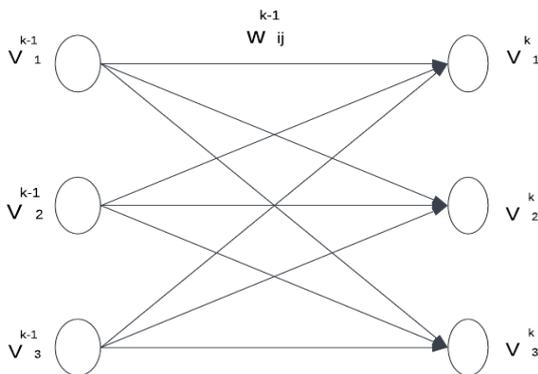


Fig. 4. Architecture of the Artificial Neural Network used.

Equation (1) and (2) explain the mathematical relation between the nodes of the two layers.

$$V_i^k = (W_{i1}^k * V_1^{k-1}) + (W_{i2}^k * V_2^{k-1}) + \dots + (W_{in}^k * V_n^{k-1}) + B^k \quad (1)$$

$$V_i^k = f(V_1^{k-1}, V_2^{k-1}, \dots, V_n^{k-1}) \quad (2)$$

where V_i^k is the i^{th} node of k^{th} layer, W_{ij}^k is the weight corresponding to the connection between the j^{th} node of layer $k-1$ and the i^{th} node of layer k , and B^k is the bias term added for calculating the values of nodes present in the k^{th} layer. Whether the ANN would be an invertible(reversible) or a non-invertible(irreversible) function depends on the architecture of the Neural Network. We have chosen a simple ANN architecture with one hidden layer. The input layer, output layer, and hidden layer each have 16 nodes. We have neglected the addition of the bias term, i.e., considered its value to be 0. This had been done to reduce the number of operations required to calculate the output from the input during encryption and the input from the output during decryption. The ANN functions as a reversible function.

The output from the ANN would be the resultant encrypted value. The inorder and preorder traversal of the binary tree

would be the inputs of the ANN. As ANN only operates upon numeric data, the input would be the ASCII values of the characters of the preorder and inorder strings. As the architecture of the ANN would be constant, the length of the input the ANN would operate upon will be constant. Therefore the preorder and inorder traversals of the binary tree would be divided into arrays with their length being equal to the number of nodes in the first layer of the ANN. The divided arrays would individually be passed into the ANN. The concatenation of the outputs from the ANN of each input array would be the resultant encrypted value. Therefore the final encrypted value would be a 2-dimensional numerical array.

D. Decrypting Artificial Neural Network Output

Decrypting the output from the ANN would involve calculating the input to the ANN using the weights and the output of the ANN. As explained in the previous section, the value of a node of a given layer of the ANN is the weighted summation of the values of all the nodes from the previous layer plus a bias term. As we have neglected the bias term, the value of a node can be considered as a mathematical function of solely the weights and the node values of the previous layer. Calculating the input value to the ANN would involve solving a series of sets of linear equations. A given linear equation set would correspond to two adjacent layers of the ANN. The unknown variables in the set would be the node values of the previous layers, which would be calculated using the node values of the forward layer and the weight connections between the two layers. The matrix representation of a given set of linear equations is shown in fig. 5.

$$\begin{bmatrix} W_{11}^{k-1} & W_{12}^{k-1} & \dots & W_{1n}^{k-1} \\ W_{21}^{k-1} & W_{22}^{k-1} & \dots & W_{2n}^{k-1} \\ \dots & \dots & \dots & \dots \\ W_{1n}^{k-1} & W_{2n}^{k-1} & \dots & W_{mn}^{k-1} \end{bmatrix} * \begin{bmatrix} V_1^{k-1} \\ V_2^{k-1} \\ \vdots \\ V_m^{k-1} \end{bmatrix} = \begin{bmatrix} V_1^k \\ V_2^k \\ \vdots \\ V_n^k \end{bmatrix}$$

Fig. 5. Matrix representation of the connection between nodes of two consecutive layers in a Neural Network. V^k and W are known values, and, V^{k-1} is the unknown variable.

Fig. 5 can be represented by the following equation:
 $W \cdot V^{k-1} = V^k \quad (3)$

Where W is the weights connecting nodes of layer $K-1$ and K , V^{k-1} is the node values of layer $K-1$, and V^k is the node values of layer k . Equation (3) can be simplified as follows:

$$V^{k-1} = W^{-1} \cdot V^k \quad (4)$$

V^{k-1} can be calculated by solving (4). The above process of calculating the values of the nodes of a previous layer from a forward layer will be repeated iteratively across the layers of the ANN from the node values of the output layer till we have calculated the node values of the input layer, i.e., input of the ANN.



Symmetric Encryption using Artificial Neural Network, Binary Tree Traversal, and Interleaving Salting

Both encryption and decryption require the weights of the ANN, i.e., the same set of values. Therefore our proposed approach is a symmetric encryption approach.

E. Binary Tree Decryption

The binary tree can be reconstructed from the inorder and preorder traversals, level-order traversal of which will reproduce the input string used to construct the binary tree.

F. Salting Decryption

Since every other character from the output string obtained by interleaving salting is a character from the salt, extracting the even indices will produce the original plaintext, i.e., the decrypted plaintext. Fig. 6 summarizes the encryption and decryption pipeline.

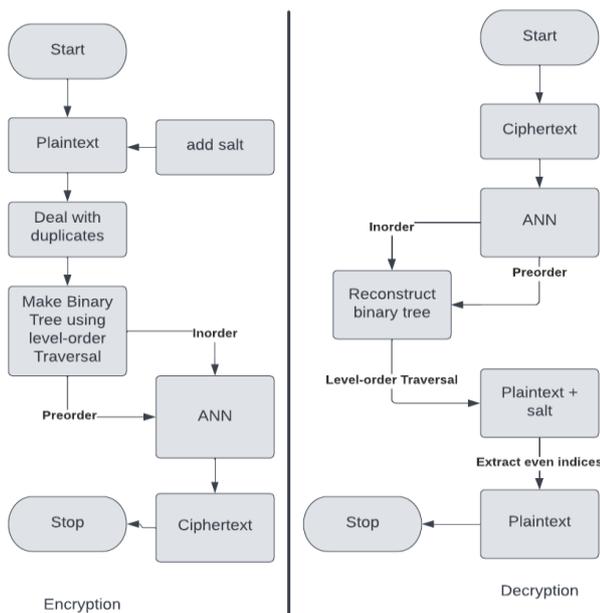


Fig. 6.Flowchart of the proposed encryption and decryption pipeline.

IV. EXPERIMENTATION AND RESULTS

The salt length is nine for the experimentation results shown in the following section. We used a 10th generation i7 core Intel processor for the below-shown results. Fig. 7 illustrates the variation of the combined total time taken by the encryption and decryption pipelines together, individual decryption time, and individual encryption time in seconds with the change in the length of the input string. We can observe that the total time increases almost linearly with the increase in the length of the input string. Fig. 7 shows that the time taken for encryption increases more significantly than for decryption. This can be attributed to the increase in the number of input batches to be passed into the ANN with an increase in the input string size.

Encryption and Decryption time in seconds vs the input string length

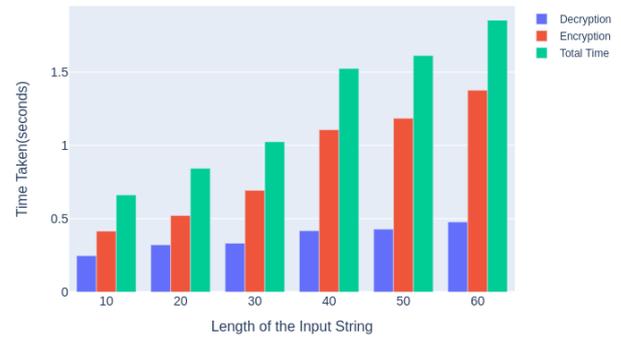


Fig. 7.Variation of the total time taken by encryption and decryption, individual encryption time, and individual decryption time in seconds with a change in input string length.

Fig. 8 shows the variation of the time taken by encryption and decryption using Binary Tree with the change in the length of the input string. The Binary Tree encryption time includes the time taken to get the preorder and inorder traversal of a binary tree such that the interleaved input string and salt is the level order traversal of the binary tree. The Binary Tree decryption time includes the time taken to get the level order traversal of a binary tree from its preorder and inorder.

Time taken in micro seconds for Binary Tree Encryption and Decryption in seconds vs the length of the input string

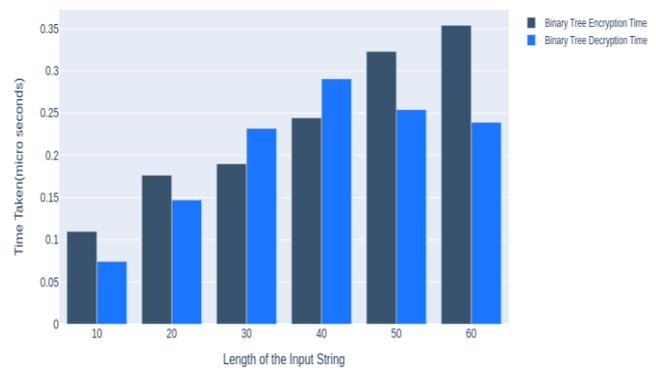


Fig. 8.Variation of the time taken by encryption and decryption using Binary Tree with the change in the length of the input string.

Fig. 9 shows the variation of the time taken by encryption and decryption by the Artificial Neural Network(ANN) with the change in the length of the input string. The ANN encryption time involves the time taken to divide the ASCII values of the preorder and inorder traversals of the binary tree into arrays of length equal to the number of nodes in the first layer of the ANN and then passing each array into the ANN to get the encrypted output. The ANN decryption time involves the time taken to get the input to the ANN from the encrypted output by performing a series of matrix multiplications, as previously explained.

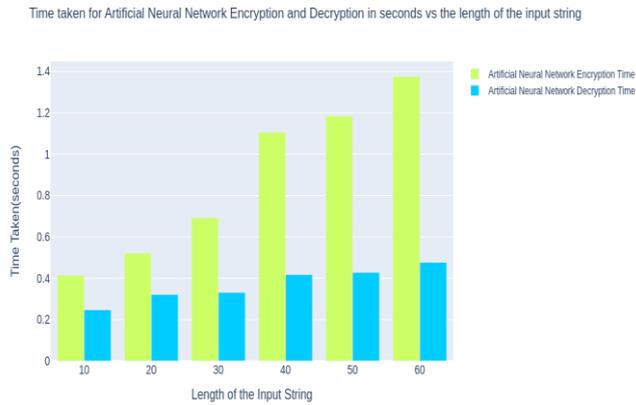


Fig. 9. Variation of the time taken by encryption and decryption using ANN with the change in the length of the input string.

V. CONCLUSION

We have proposed a novel approach to Symmetric Encryption that builds upon previous approaches utilizing binary trees and jumbled salting by employing an Artificial Neural Network that acts as a reversible, i.e., invertible mathematical function. The utilization of ANN has been proposed in a way such that each user is allotted a unique key, hence safeguarding the encrypted data of all remaining users even when the key of a particular user is compromised.

REFERENCES

1. H. Delfs, H. Knebl, and H. Knebl, Introduction to cryptography. Springer, 2002, vol. 2. [CrossRef]
2. S. Chandra, S. Bhattacharyya, S. Paira, and S. S. Alam, "A study and analysis on symmetric cryptography," in 2014 International Conference on Science Engineering and Management Research (ICSEMR). IEEE, 2014, pp. 1–8. [CrossRef]
3. A. V. Zea, J. F. Barrera, and R. Torroba, "Cryptographic salting for security enhancement of double random phase encryption schemes," Journal of Optics, vol. 19, no. 10, p. 105703, 2017. [CrossRef]
4. P. P. Churi, V. Ghate, and K. Ghag, "Jumbling-salting: an improvised approach for password encryption," in 2015 International Conference on Science and Technology (TICST). IEEE, 2015, pp. 236–242. [CrossRef]
5. V. Ustimenko, "Cryptim: Graphs as tools for symmetric encryption," in International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes. Springer, 2001, pp. 278–286. [CrossRef]
6. A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," Computer, vol. 29, no. 3, pp. 31–44, 1996. [CrossRef]
7. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in International conference on the theory and applications of cryptographic techniques. Springer, 2004, pp. 506–522. [CrossRef]
8. A. Singh and R. Gilhotra, "Data security using private key encryption system based on arithmetic coding," International Journal of Network Security & Its Applications (IJNSA), vol. 3, no. 3, pp. 58–67, 2011. [CrossRef]
9. J. Malone-Lee, "Identity-based signcryption," Cryptology ePrint Archive, 2002.
10. J. Thakur and N. Kumar, "Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis," International journal of emerging technology and advanced engineering, vol. 1, no. 2, pp. 6–12, 2011.
11. S. Singh, S. K. Maakar, and S. Kumar, "A performance analysis of des and rsa cryptography," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 2, no. 3, pp. 418–423, 2013.
12. J. Katz, "Binary tree encryption: constructions and applications," in International Conference on Information Security and Cryptology. Springer, 2003, pp. 1–11. [CrossRef]

13. N. Provos and D. Mazieres, "Bcrypt algorithm," in USENIX, 1999.

AUTHORS PROFILE



Atishay Jain is a 4th-year undergraduate student at Vellore Institute of Technology pursuing his undergraduate degree in Computer Science with a Specialization in Data Science. His current research interests include Machine Learning, Deep Learning, and Computer Vision.



Mihir Gupta is a 4th-year undergraduate student at Vellore Institute of Technology pursuing his undergraduate degree in Computer Science. His current research interests include blockchain and cryptography.



Vibhu Kumar Singh is a 4th-year undergraduate student at Vellore Institute of Technology pursuing his undergraduate degree in Computer Science. His current research interests include Cryptographic Security Protocols.



Dr. Aju Dennisan is an experienced Associate Professor with a demonstrated history of working in the education management industry. Skilled in Computer Science, Adobe Creative Suite, Databases, Curriculum Development. Strong education professional with a Doctor of Philosophy (Ph.D.) focused in Medical Image Processing from Vellore Institute of Technology.