Journal of Theoretical and Applied Information Technology

31st December 2023. Vol.101. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GRØSTL STOCHASTIC GRADIENT DEEP MULTILAYER PERCEPTIVE BLOCKCHAIN BASED SIDE CHANNEL ATTACK DETECTION FOR ENHANCED SECURITY IN CLOUD COMPUTING

RAMAKRISHNA SUBBAREDDY[1] Dr.P. TAMIL SELVAN[2]

[1]Research Scholar, Department Computer Science, Karpagam Academy of Higher education, Echanari, Coimbatore, INDIA

[2]Associate Professor, Department Computer Science, Karpagam Academy of Higher education, Echanari, Coimbatore, INDIA

E-mail: [1] ramki.blr@gmail.com , [2]tamilselvancs@kahedu.edu.in

ABSTRACT

Data security has a vital role in Cloud Computing. Cache side channel attacks are a type of cryptanalysis in the cloud for acutely threatens the security of the cryptosystem. Therefore, a novel cryptography model named Grøstl stochastic gradient deep multilayer perceptive Blockchain (GSGDMPB) model is designed for detecting side channel attacks in the cloud computing environment. At first, the Grøstl cryptography function generates hash value for every user data. Then Borda positional voting consensus algorithm is also applied for identifying the active blocks based on the majority votes. Secondly, Lai-Massey stochastic gradient deep multilayer perceptive learning is employed to perform encryption and decryption. After that, the generated cloud user block validation is performed based on the simplex matching coefficient. Then the max-out activation function is employed to provide final attack classification outcomes through enhanced accuracy. Experimental assessment of proposed model is performed by dissimilar metrics by a different number of traces. The results of GSGDMPB model improves data communication security through enhanced channel attack detection accuracy, throughput, and minimum overhead and time than the conventional methods.

Keywords: *Cloud Computing, Security, Side Channel Attacks, Grøstl Cryptography, Borda Positional Voting Consensus Algorithm, Lai-Massey Stochastic Gradient Deep Multilayer Perceptive Learning, Simplex Matching Coefficient, Maxout Activation Function.*

1. INTRODUCTION

CC allows additional users to store as well as distribute their applications and data. Network security has been a key concern in cloud-based environments to protect data from malicious attacks. In the cloud, a side-channel attack is a security violence that aims to leak information from a physical cryptosystem. Several methods have been developed for side-channel attack detection to enhance the security level in loud.

In [1] Parallel Sponge-Based AE with Side-Channel Protection and Adversary-Invisible Nonces (PSASPIN) was developed using parallel fresh rekeying in addition to the sponge construction for identifying the lower-order differential power attacks. The PSASPIN uses a key generation PRF on the basis of proposed Galois Field multiplication. A deep learning models-based side-channel analysis (DL-SCA) with a modular network was developed in [2] to detect the profiled side-channel attacks. But the modular network was not improved since it failed to apply other types of deep learning architecture by tuning the hyperparameters (i.e., weight).

A new secured cloud infrastructure technique was developed in [3] to detect cache-based side-channel attacks. But the other types of attacks were not detected. A secure data deduplication system was developed in [4] to resist the two typical side-channel attacks and this method was greatly save the storage overhead of cloud by remove duplicated data and retaining one copy. But the overhead of the attack detection was not minimized. Nemesis Guard mechanism was designed in [5] to automatically mitigate side-channel attacks with minimum execution time. But it failed to improve the accuracy of attack mitigation.

An RSA cryptosystem was introduced in [6] for high-resolution cache side-channel attack detection.

But performance of attack detection was not enhanced. For executing side channel analysis based on FPGA crypto implementations, Long Short-Term Memories (LSTM) was developed in [7]. But it has more running time for side-channel analysis.

Different deep learning techniques were developed in [8] for detecting the non-profiled side-channel attacks based on the AES-128 encryption algorithm. But the performance of neural networks for non-profiled attack detection was not improved. Neural networks underlying architecture were designed in [9] for detecting the side channel attack analysis. However, the cryptographic technique was not implemented to improve the security level. A quantum key distribution system was introduced in [10] for side-channel attacks on the key reconciliation side-channel attacks. But the higher throughput was not achieved.

1.1 Major Contributions of The Paper

- To enhance the security in cloud, a novel GSGDMPB model is introduced based on the block generation and validation.
- To minimize the communication overhead and increase the throughput, GSGDMPB model uses the Grøstl cryptographic hashed positional voting consensus algorithm. The Grøstl cryptography function generates the hash value for each cloud user data. This helps to minimize the communication overhead. After that the Borda positional voting consensus algorithm is applied to detect the active blocks based on the majority votes principles. These blocks are used to form a chain for improving the data transmission with higher throughput.
- To improv attack detection accuracy with minimum time, Lai-Massey stochastic gradient deep multilayer perceptive learning is developed. The Lai-Massey cryptographic technique with symmetric key used for encryption and decryption of cloud user block. After that, the simplex matching coefficient is used for block validation. Then the maxout activation function provides the different side channel attack classification results based on coefficient results.
- Finally, comprehensive experiment evaluations are carried out to estimate the performance of the GSGDMPB model along with the various metrics.

1.2 Structure Of Paper

Remainder of manuscript is organized as below. Section 2 reviews literature review of security in cloud. Section 3 gives detailed description of GSGDMPB model. Section 4 explains experimentation through dataset description. In section 5, performance outcomes of GSGDMPBmodel and conventional methods are examined by various metrics. Finally, Section 6 concludes manuscript.

2. LITERATURE REVIEW

Quantum key distribution **(**QKD**)** system was developed in [11] for side-channel attack detection**.** But the conditional security with limited technological power was not improved. The security of data transmission was performed in [12] by introducing a blockchain data transfer based on homomorphic encryption. But the overhead was not minimized.

For preventing private data loss during data communication Linear Elliptical Curve Digital Signature (LECDS) with Hyperledger blockchain was designed [13]. But the data integrity was the most important concern. An Artificial Neural Network (ANN) was developed in [14] to perform a secure communication system for a cloud environment. But the designed system provides less security.

An optimized deep learning security analytics method was developed in [15] for detecting the attacks of cloud computing. But it failed to perform the multiclass classification of different attacks. A new Twin support vector by deep kernel scheme was introduced in [16] by using AES-128 for profiled power side-channel attacks. However, failed to address the time utilization of attack detection.

Deep neural networks were applied in [17] to side-channel analysis. But the designed method considered a less number of traces for attack detection. A dynamic compiler approach was introduced in [18] for mitigating the side-channel attacks with less overhead. A multi-input deep-learning model was developed in [19] for side-channel attacks. But the other cryptographic algorithms were not implemented to detect the side-channel attacks for enhancing security. An effective Convolutional Neural Network (CNN) based approach was developed in [20] for side-channel attack detection by consuming fewer resources. But the error rate of attack detection was not reduced.

1.3 Problem Statement

The performance of the modular network was failed to improve because other types of deep learning architecture was not applied to tuning the hyper parameters (i.e., weight). A secure data deduplication system was performed to resist the typical side-channel attacks and this method significantly save the storage overhead of cloud by remove duplicated data and retaining one copy. But it failed to reduce the overhead of attack detection. Different deep learning techniques designed to detect the non-profiled side-channel attacks based on AES-128 encryption algorithm. But the performance of neural networks for non-profiled attack detection was not improved. A Neural network underlying architecture were designed for detecting the side channel attack analysis. Nevertheless, the cryptographic technique was not improving the security level. A quantum key distribution system was introduced to side-channel attacks on the key reconciliation side-channel attacks. But the higher throughput was not achieved. To overcome this Grøstl stochastic gradient deep multilayer perceptive Blockchain (GSGDMPB) model is designed for detecting side channel attacks in CC environment.

3. PROPOSED METHODOLOGY

Cloud computing is a type of technology that offers remote services on the internet to handle, access, and store data. Due to the increasing trend of users in the cloud environment, hackers have more and more chances to attack them effectively. Detecting side-channel attacks in the cloud is important to protect cloud infrastructures from attacks. Attack detection is difficult in cloud infrastructures due to the complex and distributed infrastructures which increase the complexity of attack detection. Proficient method named GSGDMPB is designed for the detection of side-channel attacks in cloud computing using an enhanced blockchain technique. Major features of cloud security component which executes recommended method are the capability to identify different types of side channel-attacks attacks as well as usage of blockchain methods.

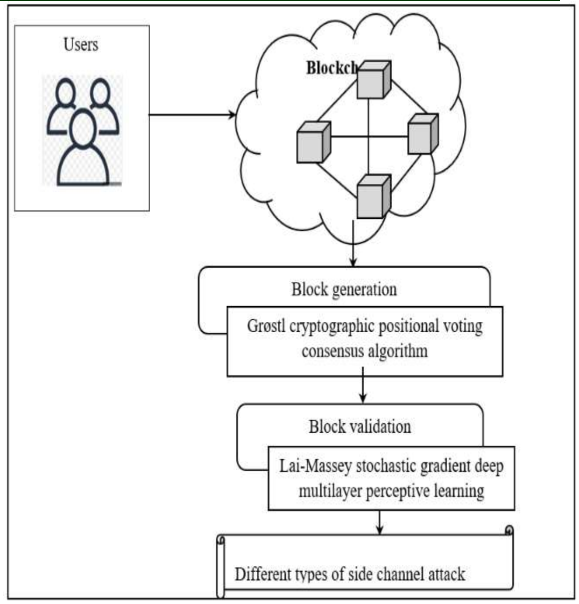

Figure 1: architecture of proposed GSGDMPB model

Figure 1 depicts architecture diagram of GSGDMPB to give security in the cloud computing environment. Our scheme contains cloud user, and cloud server. Cloud server is a centralized service that gives enduring data storage as well as retrieval services. A user who generates the data either uploads or downloads data items to or from cloud. Once user uploaded data to cloud, he becomes owner of which data item. During data transmission, security is the most important task. The proposed GSGDMPB model uses improved blockchain technology to enhance security by detecting side-channel attacks in cloud environment. The blockchain includes a set of distributed ledgers with increasing lists of blocks linked together by means of cryptographic hashes. The blockchain employs decentralized architecture as well as store cloud user data by various kinds of cryptographic techniques in databases.

The proposed methods contain block generation and validation, between user as well as cloud server. Initially, the cloud service provider generates blocks for each registered user using the Grøstl cryptographic hashed positional voting consensus algorithm. The hash value is created with the Grøstl cryptography function for each user data. Followed by the active blocks are determined by Borda positional voting consensus algorithm via majority votes. With the generated blocks, validation-based side channel attack detection is carried out by applying a Lai-Massey stochastic gradient deep multilayer perceptive learning. A brief description of

Journal of Theoretical and Applied Information Technology
31st December 2023. Vol.101. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

these two processes of the GSGDMPB model is explained in the following subsections.

3.1 Grøstl Cryptographic Positional Voting Consensus Algorithm for Block Generate On

Blockchain is also called distributed ledger technology and it makes record of any digital transaction unalterable as well as visible during decentralized network and cryptographic hashing method. The data recorded on the blockchain cannot be altered once written and it enhances the level of security. Every chain consists of numerous blocks and each block contains the data of the cloud user.

Registered user data is stored to block and initiates transaction in block format. Initial, block is generated for every user to build chain. First, the cloud user 'CU' transmits a request to Cloud Service Provider 'CSP'. After receiving request, CSP' assigns the resources with block.

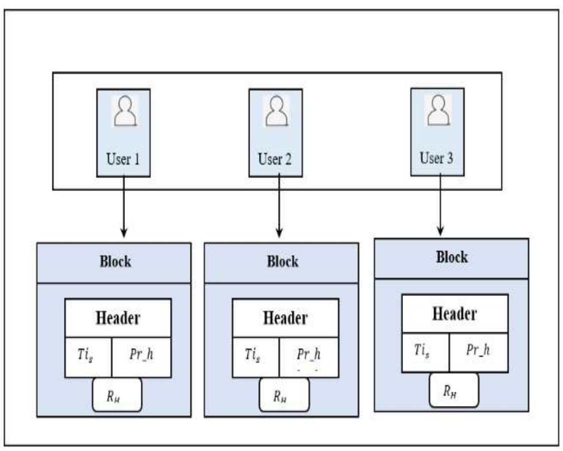

Figure 2 : construction of Block

Figure 2 illustrates generation of block for every cloud user. Every block contains block header, cryptographic hash of the previous block 'Pr_h', a timestamp 'Ti_s', and root hash 'R_H'. each transaction comprises the user data. Timestamp proves which transaction data exist when block was produced. Since every block consist of information about previous block, they efficiently construct chain. Root hash (R_H) value is generated by Grøstl cryptographic hash function to improve data integrity. As exposed in block generation, data block has user data. After that user information's is protected by applying the Grøstl cryptographic hash function. Grøstl cryptographic hash function transforms two fixed-length inputs and provides a fixed-length output.

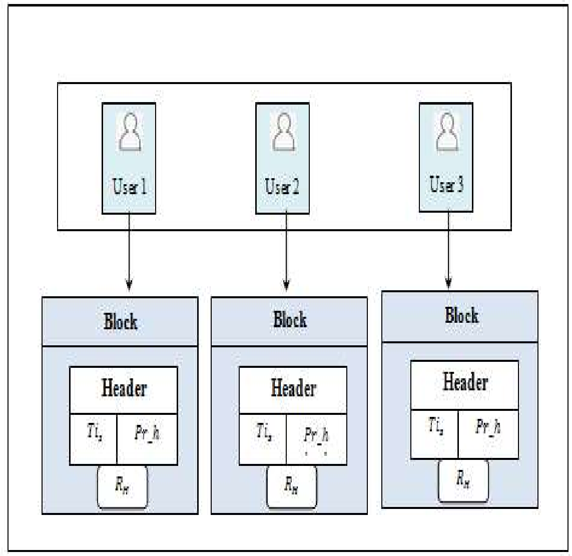

Figure 3: block diagram of hash generation using Grøstl cryptographic hash function.

Figure 3 depict block diagram of hash generation for every user data using Grøstl cryptographic hash function. Then input data is given to Grøstl cryptographic function. The Grøstl cryptographic function receives input data and it separated to number of message blocks or chunks (c_i)

$$D_i = c_1, c_2, c_3, \dots . c_k \quad (1)$$

Where, D_i indicates a user data, $c_1, c_2, c_3, \dots . c_k$ denotes a message blocks or chunks with fixed size. Each message block is provided to compression function $f_{C1}, f_{C2}, f_{C3}, f_{C4}, \dots f_{Ck}$.

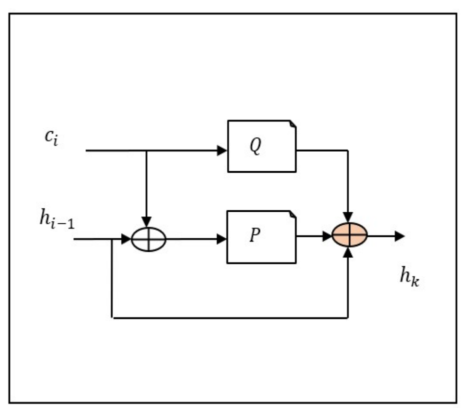

Figure 4: process diagram of Grøstl compression function

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 4 portray process of Grøstl compression function that receives input message block 'c_i' and previous hash value 'h_{i-1}'. In first round, the algorithm initializes constant pre-specified initial value (h_0). Output of the Grøstl compression function is obtained as below,

$$h_i = fc(h_{i-1}, c_i) \quad (2)$$

The compression function 'fc' generates the output hash based on 256- or 512-bit permutation functions P and Q.

$$fc(h_{i-1}, c_i) = P(h_{i-1} \oplus c_i) \oplus Q(c_i) \oplus h_{i-1} \quad (3)$$

Where, $fc(h_{i-1}, c_i)$ indicates a compression function, P, Q denotes a pair of 256- or 512-bit permutation functions, '$\oplus$' indicates an XOR operation. The final hash is obtained at the output of the last compression function 'f_{Ck}'. The obtained hash value 'h_m' is given to the input of the truncation function as given below,

$$h_i = T\left(P(h_k) \oplus h_k\right) \quad (4)$$

Where, T denotes a truncation function, h_k denotes an output hash of the final message block, P denotes a permutation. During the truncation process, the right half of $(P(h_k) \oplus h_k)$ last 'n' bits are obtained, and all other bits are truncated. In this way, root hash values are generated for each data with the objective of ensuring its integrity.

Whenever, the cloud user sends the request to access data from the block in the blockchain. The cloud service provider distributes the request message to other blocks in the distributed network.

$$CSP \xrightarrow{RE_q} \sum_{i=1}^{n} Bl_i \quad (5)$$

Where, CSP denotes a cloud service provider, RE_q denotes a request, Bl_i blocks in blockchain. For each block, sends the response the feedback information about the new block in the consensus process. The feedback information includes the behavior information of that block.

By applying a Borda positional voting consensus algorithm, active and inactive cloud user's block is identified to improve trust between blocks. But conventional consensus method suffers from minimum throughput, and enhanced vulnerability to dissimilar kinds of attacks in network. Therefore, the proposed technique uses the Borda positional voting consensus algorithm for block behavior information identification.

Let us consider $D(Bl_i, V)$ be the number of cloud user's blocks Bl_i and the votes 'V'. The active cloud user's block is identified by a majority vote of every other user's blocks.

For lowest ranked (R_{low}) cloud user's blocks Bl_i gets 0 votes,

$$R_{low}(Bl_i) = 0 \quad (6)$$

For next- lowest ranked (R_{n-low}) cloud user's blocks Bl_i gets 1 vote,

$$R_{n-low}(Bl_i) = 1 \quad (7)$$

For the highest-ranked $(R_{high}(Bl_i))$ cloud user's blocks Bl_i gets the votes as give below,

$$R_{high}(Bl_i) = n - 1 \quad (8)$$

Where n the number of cloud user's blocks in blockchain. Once all votes have been counted, the blocks with the higher votes are the said to be an active cloud user's block. Otherwise, the blocks are said to be inactive. Then the active cloud user's block is linked to chain. Otherwise, the block is not linked to the blockchain. In this way, the cloud user's blocks are generated.

The pseudo code representation of Grøstl cryptographic hashed positional voting consensus algorithm for block generation is given below.

Algorithm 1: Grøstl cryptographic hashed positional voting consensus algorithm for block generation
Input: Dataset 'DS', Cloud User '$CU = CU_1, CU_2, \dots, CU_m$', Block '$Bl = Bl_1, Bl_2, \dots, Bl_n$', Cloud Service Provider 'CSP'
Output: Block generation
Begin 1: **For** each registered cloud user CU 2. Generate data in the form of block 'Bl' 3. **For each data block** 4. Generate root hash value 'h_i' using (4) 5. **End for** 6. Cu' initiate the block 'B' sends its request 'RE_q' to the service provider 'CSP' 7. CSP distributes the request to other blocks 'Bl_i' using (5) 8. **For each** other block in network 9. Sends the feedback information **to** CSP interms of voting 10. CSP counts the vote ''

```
        if (V(B_i) > th) then
            cloud userblock is active
            linked to chain
    else
            cloud userblock is inactive
            not linked to chain
    End if
    Return blocks generated 'Bl'
End for
End for
End
```

Algorithm 1 illustrates block generation by Grøstl cryptographic hashed positional voting consensus algorithm. For every user data in, generate hash value using Grøstl cryptographic technique to guarantee integrity. Next, generated user block is active or inactive by borda positional voting consensus algorithm. Through this, active cloud user block is linked to chain. But the inactive cloud user block is not linked to chain in dispersed network. In this way, active cloud user blocks are generated.

3.2Lai-Massey Stochastic Gradient Multilayer Perceptive Deep Learning-Based Block Validation

Once block is generated, block validation is carried out. Validation is important process previous to data transaction. This guarantees which only valid generated blocks are distributed on network. Proposed technique uses the Lai-Massey stochastic gradient multilayer perceptive deep learning to perform the block validation for identifying dissimilar kinds of side-channel attacks as well as guarantee the security of data transmission.

The proposed technique uses Side Channel Attacks Assisted with Machine Learning (SCAAML) dataset that included an input with more than 8000 sample instances collected for simulation and each trace is visualized in the form of blocks. In network security, a side-channel attack is any attack based on additional information gathered due to a basic way. Among different types or classes of side-channel attacks, our work considered the following attacks.

- **Cache-based attack**: It is a type of side-channel attacks where shared cache is utilized to gain data through analyzing the hit/miss ratio as well as time needed to entrance assured cache lines.
- **Timing Attack**: It is another type of side-channel attacks along with the time different computations take to perform.
- **Data Remanence attack:** It is another type of attacks in which sensitive generated block sensitive data are read subsequent to evidently removed.
- **Differential Fault Analysis: It** is a type of side-channel attack wherein the generated block is obtained by introducing faults in computation.
- **Allocation-based side channels attacks**: the attacker monitors the number of resources that have been allocated to a cloud user block generation and validation.

A Multilayer Perceptive Classifier is type of feed-forward deep neural network. It represents which the data is transferred from input layer to the output layer in the forward manner. The structural diagram of the Block Validation is shown in figure 2.

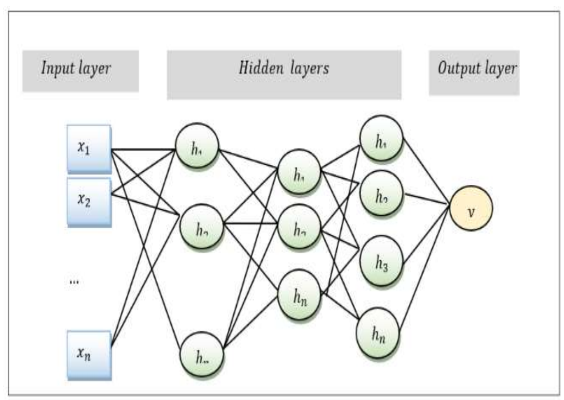

Figure 5: structure of Deep Multilayer Perceptive neural learning

Figure 5 illustrates structure of Deep Multilayer Perceptive neural learning with multiple layers such as input layer, three hidden layers, as well as output layer. Each layer comprises small identical units named artificial neurons or perceptrons. Every neuron in one layer is completely linked to other neurons in subsequently successive layer.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

As depicted in figure 5, first layer is input layer, and its neurons receive the values of input i.e., generated user blocks. The last layer is the output layer, and it has four units to provide the network outputs in terms of four different types of attacks. The two or more arbitrary number of hidden layers is positioned between input as well as output layer. Hidden layers are true computational position of deep multilayer perceptive classifier.

The input layer receives the number of generated user blocks $Bl_i \in Bl_1, Bl_2, Bl_3, \dots . Bl_n$. Each generated user blocks are associated with a weight '$a_1, a_2, \dots, a_n$' and summed with bias 'k'. Therefore, the activity of neuron is given below,

$$x(t) = [\sum_{i=1}^{n} Bl_i * a_i] + H \quad (9)$$

Where, '$x(t)$' denotes a activity of neuron at input layer and obtained based on the number of generated user blocks $Bl_i \in Bl_1, Bl_2, Bl_3, \dots . Bl_n$ multiplied with the weights 'a_i' and summed with the bias 'H' that stored integer value is '1'.

After that input is forwarded to first hidden layer where encryption and decryption are performed between the sender and receiver using Lai–Massey scheme. Lai–Massey scheme is deterministic cryptographic algorithm operating on fixed-length collections of bits, named blocks. Major advantage of this cryptographic algorithm is widely used to encrypt large volumes of data. The Lai–Massey scheme is a symmetric cryptographic algorithm that uses a similar key for both encryption and decryption.

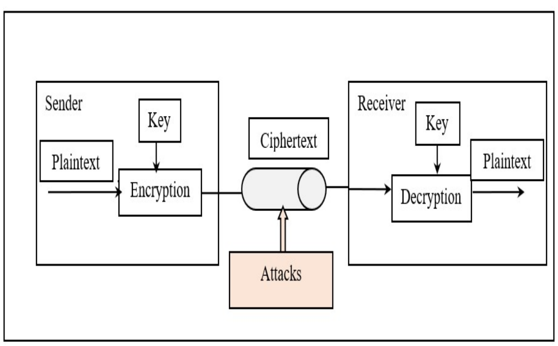

Figure 6: encryption and decryption using Lai–Massey cryptographic scheme.

Figure 6 illustrates the encryption and decryption using Lai–Massey cryptographic scheme. The input of the plaintext is a generated block.

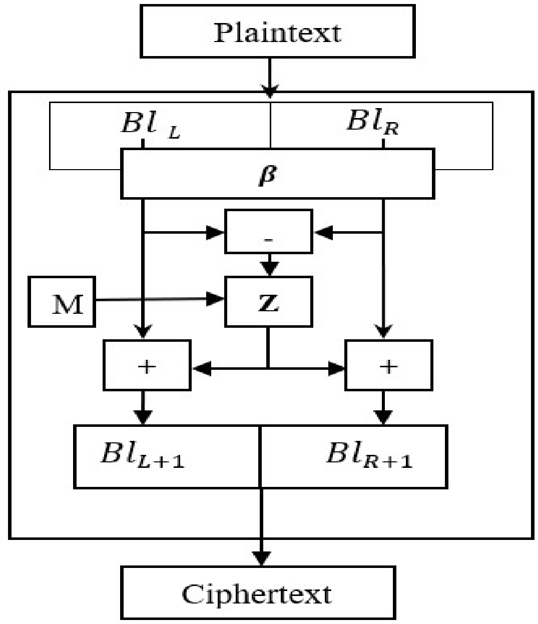

Figure 7: flow process of encryption

Let us consider the plain text 'Bl_i' also splits into two equal parts 'Bl_L' and Bl_R.

$$'Bl_i = (Bl_L, Bl_R) \quad (10)$$

For each round, compute

$$(Bl'_{L+1}, Bl'_{R+1}) = \beta\,(Bl_L + G,\ Bl_R + G) \quad (11)$$

Where, β denotes a half-round function,

$$G = Z\,(Bl_L - Bl_R, M_i) \quad (12)$$

Where, Z denotes a round function, M_i denotes a sub keys for each round $i = 1,2,3, \dots w$

$$(Bl_{L0}', Bl_{R0}') = \beta\,(Bl_{L0}, Bl_{R0}) \quad (13)$$

The ciphertext is obtained as follows,

$$(Bl_{L+1}, Bl_{R+1}) = \beta\,(Bl_{L+1}', Bl_{R+1}') \quad (14)$$

From equation (14), the encrypted output is stored in the blockchain server.

Next, On the receiver end, the decryption is performed with the symmetric sub keys key. The

Journal of Theoretical and Applied Information Technology
31st December 2023. Vol.101. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

decryption process is an inverse process of the plain text generation. The flow process of decryption is shown in Figure 8.

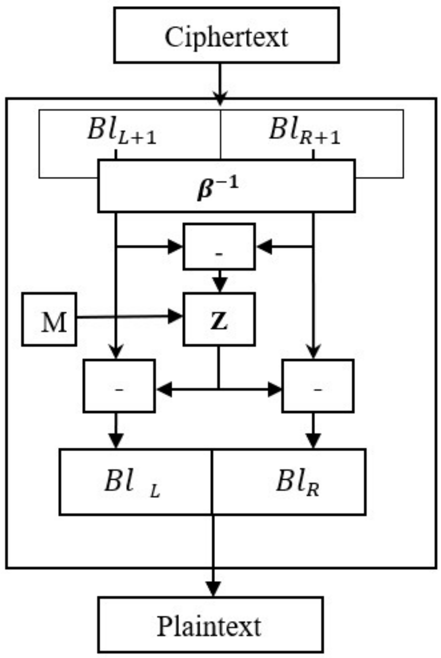

Figure 8: flow process of decryption

The decryption performed by the receiver as given below,

For all rounds, calculate.

(〖Bl'〗_L, 〖Bl'〗_R)=β^(-1) (〖Bl〗_(L+1)^'-G, 〖Bl'〗_(R+1)-G) (15)

From equation (15), the half-round function represented as β,

G=Z (〖Bl'〗_(L+1)-φ_R,M_i) (16)

In equation (16), (Bl〖_L〗,〖Bl〗_R) denotes a plain text of the generated block, M_i denotes a symmetric sub key for each round i=1,2,3,…w,

(〖Bl'〗_(L+1), 〖Bl'〗_(R+1))=β^(-1) (〖Bl〗_(L+1), 〖Bl〗_(R+1)) (17)

The plain text is obtained as follows,

(Bl〖_L〗,〖Bl〗_R)=(Bl'〖_L〗,〖Bl'〗_R) (18)

The decrypted output is stored in the blockchain server.

The ciphertext is given to second hidden layer where authentication process is performed using smart contract. Smart contract is self-implementing contract in blockchain which resides among two parties (i.e., sender and receiver) based on agreement or assured rules which are encoded and stored in the blockchain. The smart contracts create communication among users during transactions. These contracts stored in blockchain mechanically carry out legally relevant events along with terms of definite rules without considering external third parties. Rule defines that only authorized users access data from server and avoid access from the unauthorized user through the simple matching coefficient.

The simple matching coefficient is statistical technique to measure relationship among plain text obtained by receiver and already stored in the cache.

$$SMC = \left[\frac{MC_{Patterns}}{Total\ N_{Patterns}}\right] \quad (19)$$

Where, SMC denotes a simple matching coefficient, $MC_{Patterns}$ indicates number of matched patterns among newly generated plaintext as well as already stored in cache at time of encryption, $Total\ N_{Patterns}$ represents total number of patterns in already stored plaintext. Coefficient gives value from 0 to 1. Output of pattern matching coefficient is provided to third hidden layer where maxout activation function is used for user authentication.

$$A = \begin{cases} if \ \arg\max \ SMC, & 1 \\ otherwise, & 0 \end{cases} \quad (20)$$

The maxout activation function 'A' returns output '1' denotes which two plaintexts are matched perfectly, '0' denotes two plaintexts are not matched. Depend on activation function outcomes, user is classed to authorized or unauthorized. If two plaintexts are matched exactly, after that user is classified to authorized user. If plaintexts are not matched exactly, then user is categorized to five types of attacks (i.e., cache attack, timing attack, data remanence attack, differential fault analysis attacks, allocation-based side channels attacks).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

After the classification, the error rate is computed as give below,

$$RE = \frac{1}{2}(y\,(tar) - y\,(pre))^2 \qquad (21)$$

Where, RE indicates an error rate, $y\,(tar)$ denotes an actual target result '$y\,(pre)$' denotes an output produced by the perceptron. In order to minimize the error, the initial weight of the input sample gets updated by applying the stochastic gradient descent. This helps to update the weights by subtracting the current weight by a learning rate of its gradient.

$$a_{new} = a_{old} - \vartheta\left[\frac{\partial RE}{\partial a_{old}}\right] \quad (22)$$

Where, a_{new} denotes updated weight, a_{old} indicates current weight, ϑ represent learning rate($\vartheta < 1$). A large learning rate permits the classifier to learn faster than the smaller value, '$\frac{\partial RE}{\partial a_{old}}$' denotes partial derivative of error 'RE' with current weight 'RE'. . This process is iterated until it reaches minimum error. Lastly, classified results are transferred into the output layer of the multilayer perceptive classifier. With the classified, secure data broadcast is performed from sender to receiver by detecting the attacks. This in turn helps to enhance accuracy of attack detection as well as reduce false positive rate.

The algorithmic steps for Lai-Massey stochastic gradient multilayer perceptive deep learning-based block validation are given below.

// Algorithm 2: Lai-Massey stochastic gradient multilayer perceptive deep learning-based block validation

Input: Generated blocks $Bl_1, Bl_2, Bl_3 \dots Bl_n$

Output: Improve attack detection accuracy

Begin

1. **Number of** generated blocks $Bl_1, Bl_2, Bl_3 \dots Bl_n$ **in the input layer**
2. **For each** generated blocks Bl_i
3. Assign the set of weight 'a_i' and bias 'H'
4. Determine the neuron activity at the input layer '$x(t)$'
5. **end for**
6. **For** each generated blocks Bl_i–**[hidden layer 1]**
7. Convert plain text into ciphertext using (14)
8. **Return (ciphertext)**
9. End for
10. **For ciphertext**
11. Convert ciphertext into plaintext using (18)
12. **Return (plaintext)**
13. **End for**
14. Apply simple matching coefficient 'SMC' –**[hidden layer 2]**
15. Apply maxout activation function 'A'–**[hidden layer 3]**
16. **If** ($arg\,max\,SMC$) **then**
17. A returns '1'
18. User is classified as authorized
19. **else**
20. A returns '0'
21. User is classified as attacks
22. **End if**
23. **For each results 'Y'**
24. Measure the error rate
25. Update the initial weight using (22)
26. Find minimum error
27. Obtain the final classification results with minimum error **at the output layer**
28. **Return** (different types of attacks)
29. **End for**

End

Algorithm 2 explains process of attack detection in cloud during data transmission. The Multilayer deep Perceptive classifier contain numerous layers to analyze the given input. Number of generated blocks for each user is provided to input layer. Weights are assigned to each input and add the bias function. After that input is sent to neuron of first hidden layer. For each generated block, encryption is performed on the sender side to generate the ciphertext. Then the receiver side decrypts the data. After that, the validation is said to be performed based on the simplex matching coefficient in second hidden layer. Then the activation function is applied to a third hidden layer to provide final attack classification results. Finally, weight gets updated to minimize error rate. This process is incessantly

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

iterated until algorithm reaches lesser error. Lastly, attack classification outcomes are obtained at output layer.

4. EXPERIMENTAL SETUP

Experimental evaluation of the GSGDMPB and existing PSASPIN [1] [2] are implemented in JAVA programming language with CloudSim simulator. To perform experiment, SCAAML dataset is employed, and it taken from the https://github.com/google/scaaml/tree/master/scaaml_intro. The SCAAML dataset includes a more than 8000 samples instances and it used for simulation. For experimental deliberation, number of samples instances taken in ranges from 800 to 8000.

5. PERFORMANCERESULTS AND DISCUSSION

Comparative outcome analysis of GSGDMPB and conventional PSASPIN [1] [2] are explained with communication complexity, throughput, attack detection accuracy and attack detection time with a different number of traces.

Communication Overhead: In computer science, overhead is some combination of excess or indirect computation time and memory that is needed to perform an exact task. In the cloud, memory is important to the security of data for the users. Blockchains (such as Bitcoin and Ethereum) have been heavily criticized owing to their high overhead. Because blockchains are digital records, it is how much data are stored on a cloud. In our work, it is measured as the amount of memory consumed for three different processes such as prepare, execute, and validate. We handle the problem of efficiently managing the computation overhead (i.e., memory) incurred with conventional blockchains and current studies on how blockchains use data to validate transactions and blocks. More transactions in blockchain, more memory it employs. Cryptocurrency "miners" authenticate novel transactions as well as search for unique hashes to allocate to them, encrypting and decrypting each entry, keeping chain secure as well as authentic. This is referred to as communication complexity and is mathematically formulated as given below.

This is referred to as the communication complexity and is mathematically formulated as given below.

$$CO = \sum_{i=1}^{n} T_i * Mem[BG\ ] \quad (23)$$

From equation (23), the communication overhead 'CO' is measured depend on network traces 'T_i' involved in the simulation process and the memory consumed in performing block generation '$Mem[BG\ ]$' respectively. It is measured in kilobytes (KB).

Throughput: It refers to the maximum amount of data being transmitted in the receiver end. It is mathematically calculated as given below,

$$Throughput = \sum \frac{T_{received}}{T_{sent}} * 100 \quad (24)$$

Where, $T_{received}$ denotes a network traces received 'T_{sent}' denotes a traces sent. It is measured in percentage (%).

Attack Detection Accuracy: It is defined as ratio of number of network traces which has detected the different types of attack accurately to total number of traces. It is formulated as below.

$$SCAD_{acc} = \sum_{i=1}^{n} \frac{T_{ad}}{T_i} * 100 \quad (25)$$

Where, $SCAD_{acc}$ denotes a side-channel attack detection accuracy, T_{ad} denotes a network trace that has detected the type of attack accurately, T_i denotes a total number of network traces. It is measured in percentage (%).

False Positive Rate: It is defined as ratio of number of network traces that has detected the different types of attack incorrectly to total number of traces. It is formulated as below.

$$FPr = \sum_{i=1}^{n} \frac{T_{ad(icorrectly)}}{T_i} * 100 \quad (26)$$

Where, FPr denotes a false positive rate, $T_{ad}(icorrectly)$ denotes a network trace that has detected the type of attack incorrectly, T_i indicates total number of network traces. It is measured in percentage (%).

Side-Channel Attack Detection Time: It refers to amount of time consumed through algorithm for identifying different types of side-channel attacks. It is measured as below.

$$SCAD_{time} = \sum_{i=1}^{n} T_i * Time\ [AD] \quad (27)$$

Where, $SCAD_{time}$ denotes a side-channel attack detection time, T_i indicates total number of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

traces, $Time\,[AD]$ denotes a time consumed in detecting the side-channel attacks. It is measured in milliseconds (ms).

Table 1: Tabulations for communication overhead

Number of Traces	Communication overhead (KB)		
	GSGDMPB	PSASPIN	
800	216	280	248
1600	256	315	288
2400	288	350	324
3200	313.6	395	352
4000	340	435	380
4800	374.4	450	408
5600	414.4	480	459.2
6400	428.8	525	499.2
7200	460.8	585	540
8000	480	625	560

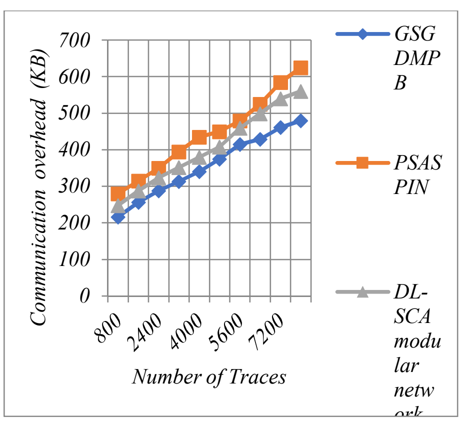

Figure 9: performance results of communication overhead

Figure 9 illustrates performance results of communication overhead with number of traces. As shown in figure, the communication complexity of the GSGDMPB model is found to be reduced than the existing PSASPIN [1] and DL-SCA modular network [2]. By increasing the number of traces from 800 to 8000, the communication complexity of every three methods gets enhanced. However, GSGDMPB model is better compared to [1] [2]. Reason for minimum computational complexity is to apply the Grøstl cryptographic hashed positional voting consensus algorithm. For each user data, Grøstl cryptographic method is to generate hash value to minimize the memory consumption of the block generation. After that, the generated user block is active or inactive before the join into the chain with the help of borda positional voting consensus algorithm. The proposed consensus algorithm minimizes the memory consumption for user data block generation. Overall performance of ten different outcomes indicates that communication complexity of GSGDMPB model is found to be minimized by 19% and 12% when compared to [1] [2] respectively.

Table 2: Compression of throughput

Number of Traces	Throughput (%)		
	GSGDMPB	**PSASPIN**	**DL-SCA modular network**
800	96	79	88.12
1600	95.75	79.5	87.5
2400	95.83	78	85.41
3200	93.75	77	84.37
4000	94.5	79.6	87.5
4800	95.41	81	88.54
5600	94.64	79.4	87.5
6400	93.98	77	86.32
7200	92.36	76.3	84.44
8000	91.25	75	83.12

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

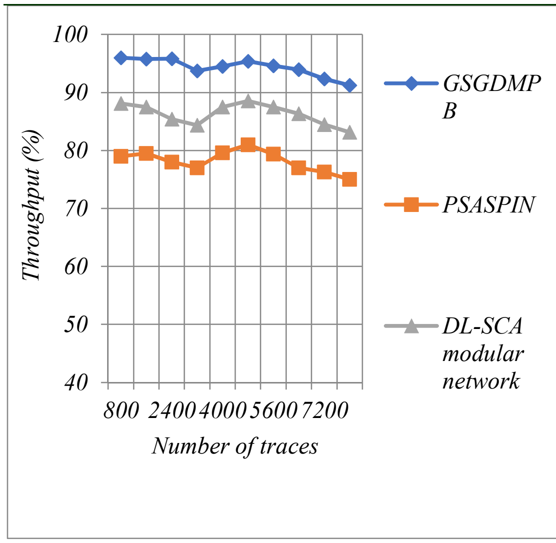

Figure 10: performance results of throughput

Table 2 and figure 10 depicts performance analysis of throughput for secure data communication in cloud. Throughput is estimated based on the data communication between sender and receiver using three different methods GSGDMPB model, PSASPIN [1] and DL-SCA modular network [2] as shown in figure 10. The performance analysis of the throughput for secure data communication is increased than the existing [1][2]. Let us consider the 800 traces for calculating the throughput in the first iteration. throughput was examined '96%' by GSGDMPB model. But throughput of conventional [1], and [2] was 79%and88.12%. Examined outcomes denotes GSGDMPB model increases throughput. After attaining ten results, the overall performance of GSGDMPB model is compared to results of existing methods. Average of ten outcomes denotes that GSGDMPB model increase throughput by 21%, and 9% than the [1] and [2]. This is because of applying the *Grøstl cryptographic hashed positional voting consensus algorithm*. First, the active blocks are identified through the Borda count *positional voting consensus algorithm for accurate data transaction. These active blocks are identified through the number of votes generates by the other blocks in the chain. This active block increases the throughput of transmission.*

Table 3: Compression of Attack detection accuracy

Number of Traces	Attack detection accuracy (%)		
	GSGDMPB	PSASPIN	DL-SCA modular network
800	95	80	86.87
1600	94.68	78.45	85.93
2400	93.75	77	84.62
3200	92.18	76	83.59
4000	93	78.55	85.62
4800	94.16	80	87.81
5600	93.39	78.35	86.71
6400	92.5	76	85.96
7200	91.94	75.25	83.13
8000	90.62	74	82.36

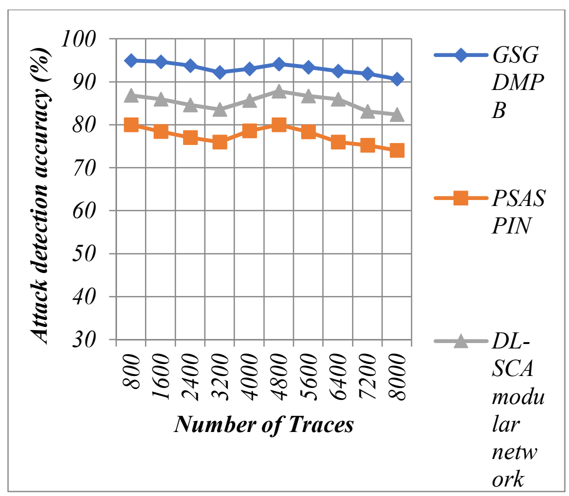

Figure 11: performance results of attack detection accuracy

Figure 11 denotes performance comparison of attack detection accuracy with number of user data. As shown in graph, number of traces is taken in horizontal axis and performance outcomes of attack detection accuracy were examined in vertical axis. Among three various techniques, GSGDMPB model enhances performance of attack detection accuracy than the conventional methods. This is proved during statistical assessment. Let us assume 800 traces considered as input in first iteration for estimating attack detection accuracy. By using GSGDMPB model, the data attack detection accuracy was examined to 95% whereas attack detection accuracy of existing [1] and [2] was found to be 80% and 86.87% respectively. Similarly, different performance results attack detection accuracy was examined for every method. Finally, overall outcomes of GSGDMPB model are compared to conventional methods. Average of ten outcomes indicates which performance of the attack detection accuracy of proposed GSGDMPB model gets enhanced by 20%, and 9% than the [1] and [2]. This is owing to proposed GSGDMPB model uses the Multilayer deep Perceptive classifier for attack detection. The number of generated blocks for each cloud user is encrypted and it transferred into receiver using Lai-Massey cryptographic technique with symmetric key. After that, obtained plaintext is verified through the matching coefficient. Based on the coefficient results, the activation function identifies the five types of the side channel attacks with higher accuracy.

Table 4: Compression of False Positive rate

Number of traces	False Positive rate (%)		
	GSGDMPB	PSASPIN	DL-SCA modular network
800	10	15	12.5
1600	11.25	16.31	13.12
2400	10.62	17.29	12.91
3200	11.40	16.40	12.65
4000	12.05	17	13.85
4800	11.77	15.10	12.75
5600	12.17	15.78	13.92
6400	11.09	14.45	12.68
7200	9.51	15.62	13.12
8000	10.97	14.37	12.81

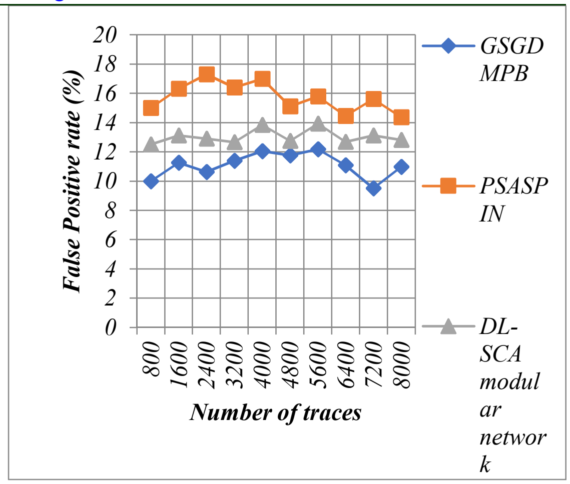

Figure 12: performance results of false positive rate

Table 4 and figure 12 depict performance of false positive rate for 8000 different number of traces. To estimate false positive rate, numbers of number of traces are taken in ranges from 800 to 8000. From above graphical outcomes, GSGDMPB model outperforms well to attain minimum false positive rate when compared to conventional outcomes. Let us assume experiment with 800 numbers of traces, performance of false positive rate was 10% by GSGDMPBmodel. By using [1] [2], performance of false positive rate was 15% and 12.5%. Similarly, various performance outcomes are examined for every method. Overall examined outcomes of GSGDMPBmodel are compared to conventional [1] [2]. Comparison outcomes clearly shows performance of false positive rateusing GPMELCB model is minimized by 29%, and 50% than the [1] and [2]. Reason behind enhancement was due to application of Multilayer deep Perceptive learning uses the stochastic gradient descent function to discover lesser error of attack detection. This aids to reduce incorrect recognition of side channel attacks.

Table 5: Compression of attack detection time

Number of Traces	Attack detection time (ms)		
	GSGDMPB	PSASPIN	DL-SCA modular network
800	640	880	760
1600	720	920	848
2400	840	980	912

3200	966.4	1085	1008
4000	1020	1245	1100
4800	1104	1355	1200
5600	1209.6	1405	1316
6400	1260.8	1485	1376
7200	1281.6	1535	1440
8000	1312	1585	1480

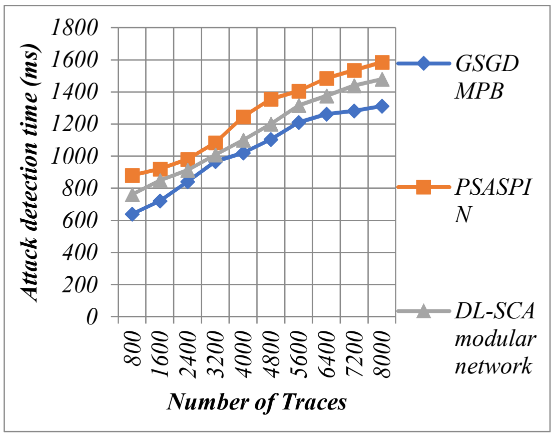

Figure 13: performance results of attack detection time

Table 5 and figure 13 depicts experimental outcomes of attack detection time depend on number of races gathered from dataset. As depicted in examined outcomes, execution time of attack detection is enhanced for every three methods as improving number of traces. However comparatively, GPMELCB model utilizes minimum amount of time to carry outside channel attack detection. This is verified during sample calculation. By considering '800' traces, attack detection time of GPMELCB model was '640ms', likewise '880ms' and 760ms are observed by [1] and [2]. Observed time of GPMELCB model is compared to existing methods. The average of ten results denotes that attack detection time is minimized by 17% and 10% by GPMELCB model than the conventional methods. This enhancement is attained by using blockchain-based Multilayer deep Perceptive learning to securely transmit the data and identifies the differ types of attacks with lesser time.

6. CONCLUSION

A new Grøstl stochastic gradient deep multilayer perceptive Blockchain (GSGDMPB) model is developed in this paper for detecting the different types of side-channel attacks to enhance security in cloud computing. The proposed GSGDMPB model first performs the cloud user block generation using the Grøstl cryptographic hashed positional voting consensus algorithm. This process enhances the throughput by finding the active block and generates the hash value for minimizing the communication overhead. After that, the Lai-Massey cryptography method is applied to perform encryption and decryption. During the data transmission, attacks are interrupted and modify the data. Therefore, block validation is performed through the matching coefficient to detect the different types of attacks. The comprehensive performance analysis of the GSGDMPB model is carried out with different metrics. The analyzed results prove that the proposed GSGDMPB model provides better performance with an improvement of higher attack detection accuracy, and throughput and minimizes the time, overhead as well as false positive rate when compared to the existing works.

REFERENCES

[1] Mohamud Ahmed Jimale, Muhammad Reza Zaba, Miss Laiha Binti Mat Kiah, Mohd Yamani Idna Idris, Norziana Jamil, Moesfa Soeheila Mohamad, Mohd Saufy Rohmad, "Parallel Sponge-Based Authenticated Encryption with Side-Channel Protection and Adversary-Invisible Nonces", IEEE Access, Volume 10, 2022, Pages 50819 – 50838. **DOI:** 10.1109/ACCESS.2022.3171853

[2] Servio Paguada, Lejla Batina, Ileana Buhan, Igor Armendariz, "Playing with Blocks: Toward Re-Usable Deep Learning Models for Side-Channel Profiled Attacks", IEEE Transactions on Information Forensics and Security, Volume 17, 2022, Pages 2835 – 2847. **DOI:** 10.1109/TIFS.2022.3196273

[3] G. Sangeetha & G. Sumathi, "An optimistic technique to detect Cache based Side Channel attacks in Cloud", Peer-to-Peer Networking and Applications, Springer, Volume 14, 2021, Pages 2473–2486.https://doi.org/10.1007/s12083-020-00996-1

[4] Youshui Lu, Yong Qi, Saiyu Qi, Fuyou Zhang, Wei, Xu Yang, Jingning Zhang, and Xinpei Dong, "Secure Deduplication-based Storage Systems with Resistance to Side-Channel Attacks via Fog Computing", IEEE Sensors Journal, Volume 22, Issue 18, 2022, Pages 17529 – 17541. **DOI:** 10.1109/JSEN.2021.3052782

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[5] Majid Salehi, Gilles De Borger, Danny Hughes, Bruno Crispo, "NemesisGuard: Mitigating interrupt latency side channel attacks with static binary rewriting", Computer Networks, Elsevier, Volume 205, 2022.Pages 1-11. https://doi.org/10.1016/j.comnet.2021.108744

[6] M. Asim Mukhtar, Maria Mushtaq, M. Khurram Bhatti, Vianney Lapotre, Guy Gogniat, "FLUSH + PREFETCH: A Countermeasure Against Access-driven Cache-based Side-Channel Attacks", Journal of Systems Architecture, Elsevier, Volume 104, 2020, Pages 1-17. https://doi.org/10.1016/j.sysarc.2019.101698

[7] Zixin Li, Zhibo Wang and Mingxing Ling, "Side-channel Attack Using Word Embedding and Long Short-Term Memories", Journal of Web Engineering, Volume 21, Issue 02, Pages 285–306. https://doi.org/10.13052/jwe1540-9589.2127

[8] Ngoc-Tuan Do, Van-Phuc Hoang, Van Sang Doan, Cong-Kha Pham, "On the performance of non-profiled side channel attacks based on deep learning techniques", IET Information Security, 2022, Pages 1-17. https://doi.org/10.1049/ise2.12102

[9] Hervé Chabanne, Jean-Luc Danger, Linda Guiga, Ulrich Kühne, "Side channel attacks for architecture extraction of neural networks", CAAI Transactions on Intelligence Technology, Volume 6, Issue 1, 2021, Pages 3-16. https://doi.org/10.1049/cit2.12026

[10] Dongjun Park, GyuSang Kim, Donghoe Heo, Suhri Kim, HeeSeok Kim, Seokhie Hong, "Single trace side-channel attack on key reconciliation in quantum key distribution system and its efficient countermeasures", ICT Express, Elsevier, Volume 7, Issue 1, March 2021, Pages 36-40. https://doi.org/10.1016/j.icte.2021.01.013

[11] Pablo Arteaga-Díaz, Daniel Cano, Veronica Fernandez, "Practical Side-Channel Attack on Free-Space QKD Systems with Misaligned Sources and Countermeasures", IEEE Access, Volume 10, 2022, Pages 82697 – 82705. **DOI:** 10.1109/ACCESS.2022.3196677

[12] Sheng Peng, Zhiming Cai, Wenjian Liu, Wennan Wang, Guang Li, Yutin Sun, and Linkai Zhu, "Blockchain Data Secure Transmission Method Based on Homomorphic Encryption", Computational Intelligence and Neuroscience, Hindawi, Volume 2022, April 2022, Pages 1-9. https://doi.org/10.1155/2022/3406228

[13] B. Sowmiya, E. Poovammal, Kadiyala Ramana, Saurabh Singh, Byungun Yoon, "Linear Elliptical Curve Digital Signature (LECDS) With Blockchain Approach for Enhanced Security on Cloud Server", IEEE Access, Volume 9, 2021, Pages 138245 – 138253. **DOI:** 10.1109/ACCESS.2021.3115238

[14] Muhammad Usman Sana, Zhanli Li, Fawad Javaid, Hannan Bin Liaqat, and Muhammad Usman Ali, "Enhanced Security in Cloud Computing Using Neural Network and Encryption", IEEE Access, Volume 9, 2021, Pages 145785 – 145799. **DOI:** 10.1109/ACCESS.2021.3122938

[15] Bhavana Gupta and Nishchol Mishra, "Optimized deep learning-based attack detection framework for secure virtualized infrastructures in cloud", International Journal Numerical Modeling, Wiley, Volume 35, Issue 1, 2022, Pages 1-20. https://doi.org/10.1002/jnm.2945

[16] Soroor Ghandali, Samaneh Ghandali, Sara Tehranipoor, "Deep K-TSVM: A Novel Profiled Power Side-Channel Attack on AES-128", IEEE Access, Volume 9, 2021, Pages 136448 – 136458. **DOI:** 10.1109/ACCESS.2021.3117761

[17] Damien Robissout, Lilian Bossuet, Amaury Habrard, Vincent Grosso, "Improving Deep Learning Networks for Profiled Side-channel Analysis Using Performance Improvement Techniques", ACM Journal on Emerging Technologies in Computing Systems, Volume 17, Issue. 3, 2021, pages 1-30. https://doi.org/10.1145/3453162

[18] Jeroen Van Cleemput, Bjorn De Sutter, Koen De Bosschere, "Adaptive Compiler Strategies for Mitigating Timing Side Channel Attacks", IEEE Transactions on Dependable and Secure Computing, Volume 17, Issue 1, 2020, Pages 35 – 49. **DOI:** 10.1109/TDSC.2017.2729549

[19] Fanliang Hu, Huanyu Wang, Junnian Wang, "Multi-Leak Deep-Learning Side-Channel Analysis", IEEE Access, Volume 10, 2022, Pages 22610 – 22621. DOI: 10.1109/ACCESS.2022.3152831

[20]Naila Mukhtar, Apostolos P. Fournaris, Tariq M. Khan, Charis Dimopoulos, Yinan Kong, "Improved Hybrid Approach for Side-Channel Analysis Using Efficient Convolutional Neural Network and Dimensionality Reduction", IEEE Access, Volume 8, Pages 184298 – 184311. **DOI:** 10.1109/ACCESS.2020.3029206