Journal of Theoretical and Applied Information Technology
29th February 2024. Vol.102. No 4

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

CONSENSUS SCHMIDT-SAMOA JACCARD EXTREME LEARNED BLOCKCHAIN FOR ENHANCED SECURITY WITH SIDE CHANNEL ATTACK DETECTION IN CLOUD

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

Side Channel Analysis (SCA) is among the newly emerged threats to affect confidential information by examining non-functional computation aspects of program implementation such as elapsed time, amount of memory allocation, or network packet size. The ability to automatically find out the amount of information that a malicious user expands through side-channel observations allows evaluation of the security of an application. During the access the confidential information from the cloud, some amount of information is discharged through side channels attacks. Therefore, it is important to specify the amount of discharging in order to detect vulnerabilities in recovering confidential information. In order to improve the side channel attack detection accuracy, a new Cryptographic Consensus Schmidt-Samoa Jaccard Extreme Learned Blockchain (CCSJELB) is introduced. The proposed CCSJELB technique performs two different processing steps such as block generation and validation. In the first step, the cloud server request to user about the registration. Then the server generates the blocks for each user with the help of the HAVAL cryptographic hash function to minimize communication overhead. Then the Wilcoxon signed-rank test consensus mechanism is applied for active or inactive user-generated blocks. After that, an improved Schmidt-Samoa Jaccard Extreme Learning Machine is employed for block validation. The Extreme Learning Machine consists of many layers for side-channel attack detection and secure data transmission. The number of generated blocks for each cloud user is given to the input layer. For each generated block, pair of keys is generated by applying improved schmidt-Samoa cryptography. After that, encryption is performed with a public key and stored on the cloud server. Whenever the user accesses the block, the validation is performed based on the Jaccard coefficient. Then the sigmoid activation function is applied to provide the attack classification results. Finally, the decryption is performed to obtain original data. The attack classification results, and secured communication results are obtained at the output layer with higher accuracy. An experimental assessment of the proposed CCSJELB is carried out with respect to communication overhead, throughput, attack detection accuracy, false positive rate, attack detection time, and confidentiality rate with a different number of traces. The analyzed performance results designate that the CCSJELB increases data communication security with a higher channel attack detection accuracy, throughput, confidentiality rate, and minimum overhead and time than the conventional methods.

Keywords: *Cloud Computing, Side Channel Attacks, HAVAL Cryptographic Hash, Wilcoxon Signed-Rank Test Consensus Mechanism, Improved Schmidt-Samoa Cryptography, Extreme Learning Machine, Extreme Learning Machine, Extreme Learning Machine*.

1. INTRODUCTION

Information security is a design constraint in roughly all domains of computing. Computing resources provided as a service through the internet is referred as cloud computing. It tracks the on-demand property and payment model. With the property of resource sharing in remote locations, cloud suffers security issues such as attacker or malicious user access, data modification, outsources auditing and so on. Among the many types of security threats, Side-Channel Analysis (SCA) is a dominant attack that assistances any leakage of a system to recover secret information. Side-channel attacks are sorted as profiled and non-profiled attacks based on an attacker's environment. A

profiled attack is an example of a side-channel attack performed through a profiling device, which is the same as the attack target device with a fixed secret key. On the other hand, a non-profiled attack is part of a side-channel attack in an environment without profiling devices. Therefore, cryptography mechanism is used to detect the attack in the cloud network to increase the security of user data. Cloud cryptography is a group of methods applied to secure data stored in cloud.

Modern cryptographic algorithms are employed to preserve information at the software level. Parallel Sponge-Based AE with Side-Channel Protection and Adversary-Invisible Nonces (PSASPIN) was introduced in [1] to detect lower-order differential power attacks. But the multiple attack detection was not resolved. A multi-input deep-learning model was developed in [2] with multiple outflow intervals collaboratively to perform the attack detection. The designed model increases the classification accuracy, but the other efficient cryptographic algorithms were not implemented for finding similar attacks.

Different deep learning techniques were developed in [3] for non-profiled attacks on the AES-128 encryption implementation. However, the technique failed to reduce the data complexity of attack detection. Information theory-based automatic neuro evolution method was introduced in [4] that successfully perform the side channel attack traces with different classes. But the performance of overhead was not minimized. A machine learning-based side-channel attack (SCA) detection approach called WHISPER was developed in [5] with higher accuracy and minimum overhead. However, the specialized machine learning models were not trained to identify anonymous attacks.

A twin support vector with a deep kernel model was designed in [6] with the implementation of AES-128 for accurately predicting the side-channel leakages. However, the performance of the throughput was not improved. An Entropy-based Cost Function (EBCF) and Illegal Sequence Detection (ISD) were developed in [7] for side-channel attacks. But the evaluation of efficient countermeasures and an extension of the attack detection framework were not considered.

A chosen-ciphertext clustering attack on CRYSTALS-KYBER was performed in [8] by means of responsive variable restrictive outflow of Barrett attenuation. But it did not guarantee security against timing attacks. The side-channel attack over the key assertion in the quantum key distribution system was developed [9] to receive a shared secret key among two users. But **THE HIGHER** confidentiality level was not improved. The secure Block-Level Double Encryption (SBLDE) model was designed in [10] for user signature authentication to recognize irregular cache data and channel behaviors during data transformation. But the designed model failed to enhance the security with efficient tracking of data for communication purposes.

1.1 Contributions of the work

The major contributions of CCSJELB technique are listed below.

- A novel method CCSJELB technique is introduced to improve the side channel attack detection with higher accuracy and minimum time.
- The HAVAL cryptographic technique is designed in CCSJELB to generate the blocks for each user to minimize communication overhead. Then the Wilcoxon signed-rank test consensus mechanism is developed for active or inactive user generated block to enhance the throughput of data transmission.
- An improved schmidt-samoa Jaccard Extreme Learning Machine is employed in CCSJELB technique for block validation. The improved schmidt-samoa cryptography is applied in Extreme Learning Machine for encryption and decryption. This helps to improve data confidentiality level. The Jaccard coefficient is applied for secret word matching to classify different types of attacks with higher accuracy and minimum time based on sigmoid activation function.
- Finally, extensive experiments are conducted to evaluate the performance of our CCSJELB technique along with the conventional scheme based on various performance metrics.

1.2 Organization

The paper is organized into various sections as follows. Section 2 provides research problem Section 3 discusses the related works. Section 4 implements the proposed CCSJELB technique. The detailed experimental setup and dataset description are presented in section 5. Performance outcomes are discussed in Section 6 and justification of the study is given in section 7. Finally, Section 8 provides the conclusion of proposed work.

2. RESEARCH PROBLEM

Cloud dominates all types of computing in the network owing to the service-oriented and on-demand nature of the model. The user utilizes a cloud model to achieve the resource needs with minimum cost. Cloud security plays a major role as it is securing the user data in a more reliable manner. Diverse stages of security problems imposed through the cloud provider are still taken as a problem in the resource level. In cloud, side channel attack leads to severe concerns as the attacker carry outing access of the resources that causes the reliability problem. Through the malicious user access, the confidentiality, integrity and security of data is degraded. Therefore, there is a need for authenticated cryptography schemes for secure communications to protect the confidentiality and integrity of cloud data. In this regard, this paper establishes a novel and efficient cryptography techniques to address the side channel attack in cloud.

3. RELATED WORKS

A Convolutional Neural Network architecture was designed in [11] for detecting side-channel attacks. But the cryptography technique was not implemented to improve confidential data transmission. An improved lightweight convolutional neural network was introduced in [12] for side channel analysis for comparative experiments. But it failed to investigate the application of deep learning techniques for side-channel analysis.

A new computational security approach was designed in [13] and decentralized blockchain network to detect the different types of side-channel IoT attacks. However, the computational burden was improved using the designed blockchain. Non-profiled side-channel attacks based on deep learning were developed in [14] to improve the success rate of attack detection. But the time complexity of the deep learning was not enhanced. A new threat model was developed in [15] against side-channel attacks to minimize communication costs. But the accuracy was not improved.

Side-channel cryptanalysis with multi-thread mixed leakage was investigated in [16]. But the deep learning technique was not applied to enhance the performance of side-channel attack analysis. An effective Convolutional Neural Network (CNN) based method was developed in [17] for side-channel analysis. However, it failed to train much quicker with better performance by means of consuming fewer resources.

A secure and lightweight method was designed in [18] to ensure data confidentiality. However, the performance of attack detection time was not minimized. New side-channel-based password cracking system was deled in [19]. But the false positive rate was not minimized. A free-space Quantum key distribution system was designed in [20] for side-channel attack detection.

4. METHODOLOGY

In side-channel attacks, the adversary corrupts sensitive information from the victim through shared resources. The sensitive information is typically related to cryptographic operations for secret-dependent control flows. A conventional cryptographic algorithm is needed to protect information at the software level. These algorithms are requiring massive computing power to break with side-channel attacks. As side-channel attacks access a physical device, they require efficient cryptographic algorithms embedded in the cloud for security. A side-channel attack is a physical attack that collides with a security system from cryptographic devices, such as operation time, resource consumption, cache access, and so on. In order to detect the attacks in the cloud environment, a novel technique called CCSJELB is introduced. This technique uses the Blockchain which is a most eminent rising technology to provide security, integrity, as well as confidentiality for any sensitive information transaction by detecting side-channel attacks in a cloud environment.

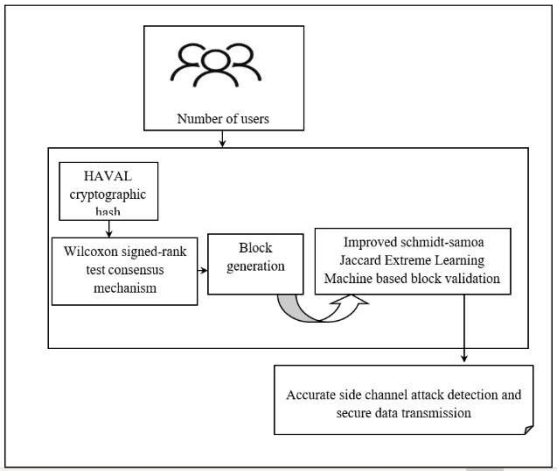

Figure 1: architecture of proposed CCSJELB model

Figure 1 portrays the structural design of the proposed CCSJELB technique to provide security of

data transmission by detecting the different types of side channel attacks in the cloud environment. The CCSJELB technique constructs the blockchain technology for side channel attack detection by including a block generation and block validation. This different process of CCSJELB technique is described in the following subsections.

4.1 HAVAL cryptographic hash block generation

The first process of the proposed CCSJELB technique is to perform the block generation using HAVAL cryptographic hash. A blockchain is a distributed decentralized technology that comprises a chain of blocks and each block encloses a set of data. The blocks are linked together using cryptographic methods and form a sequential chain of information. Blockchain technology is used to protect and secure sensitive data in online data transactions. Contrary to conventional cryptographic techniques, there is no third-party interference is required. The main advantage of the blockchain is to increase the speed of the transaction in the cloud. Therefore, the proposed technique is used to enhance blockchain technology to speed up the transmission with higher security.

Before the data transmission, the user registers their details to the cloud server. Block is generated for each registered user. Every blockchain consists of numerous blocks and each block contains the data of the cloud user.

Let us consider the cloud user 'CU' transmits a request to the Cloud Service Provider 'CSP'. After receiving the request, CSP' generates a block.

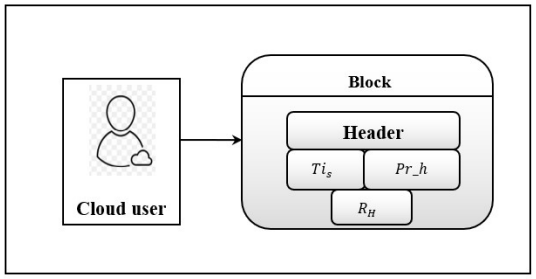

Figure 2: Block Generation

Figure 2 illustrates the block generation for each registered user. Every block includes a header, cryptographic hash of the previous block 'Pr_h', a timestamp ' ⟦Ti⟧ _s', and root hash 'R_H'. Each transaction comprises the user data. The root hash (R_H) is generated using hash function cryptographic hash function to enhance the data integrity. As shown in the block generation, the data block has a user data. Then the user information's is protected by applying the HAVAL cryptographic hash function.

First, the input user data is divided into number of message blocks

$$D\_i=b\_1,b\_2,b\_3,\ldots.b\_k \quad (1)$$

Where,D_i indicates a user data divided into message blocks b_1,b_2,b_3,….b_k with a uniform size and the hash is computed from the initial value (b_0)

$$h\_0 \leftarrow b\_0 \quad (2)$$

$$h\_(i+1)= ⟦[Q]⟧ \_(b\_i) (h\_i) \oplus h\_i] \text{ for } 0 \leq i < n \quad (3)$$

Where, h_(n,) denotes a final hash output of the 128bit variant, the data block (b_i) as the key to a block cipher 'Q' and provide to the previous hash value (h_i). The output ciphertext is then also '⊕' Xored with the previous hash value and the message block (b_i) as the key to a block cipher 'Q. Initially, the previous hash value it set to a pre-specified initial value (h_0). As a result, generated hash of one data block is not the related to another block i.e h_1≠h_2. The final resultant hash is obtained as the output of the last data block.

4.1.2 Wilcoxon signed-rank test consensus mechanism

Once the block is generated, active and inactive cloud user's block is determined to enhance the trust among the blocks. An active and inactive cloud user's blocks are identified through the Wilcoxon signed-rank consensus mechanism. The consensus mechanism is applied to attain trust and security across a decentralized cloud network.

Wilcoxon signed-rank test is a non-parametric statistical test used to find the active user block based on trusted behavior. The trusted behavior of each user block is identified by sending the feedback information from other blocks about the particular block in the consensus process.

First, collect all the feedback information from other blocks. Then assign the sign for all the feedback.

$$Q=\sum\_(i=1)\^n ⟦S\_(n)f\_i⟧ \quad (4)$$

Where, Q denotes signed-rank sum test, S_(n) denotes sign i.e positive (+ve) or negative (-

ve), f_i indicates a feedback information. Based on the positive sign, the blocks are sorted in an ascending order. Then the positive ranked feedback information observation is one, the rank of the next smallest is two, and so on. In this way, the ranking is applied. Finally, high ranked blocks are selected as an active user block and other blocks are removed.

$$Z=R\_high\ (\ [\![Bl]\!]\ \_i\ ) \quad (5)$$

Where, Z denotes an output of Wilcoxon signed-rank test, R_high denotes higher ranking block ([[Bl]] _i). The selected active cloud user's block are connected to chain and other blocks are removed. The pseudo code representation of block generation process is given below.

Algorithm 1: 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: **Foreach** user CU

2. Register the details to cloud server 'CS'

3. CSP' generate block 'Bl'

4. **For each user** block 'Bl'

5. Generate root hash value 'h_i' using (3)

6. **End for**

7. End for

8. user send requestto join the chain to the service provider 'CSP'

9.CSP distributes the request to other blocks 'Bl_i'

10. For each other block 'Bl_i' in chain

11. Sends the feedback 'f_i' **to** CSP

12.CSP assign the sign to find positive or negative feedback

13. CSP assign the rank based on sign

14. $\boldsymbol{if}\ (R_{high}(Bl_i))$**then**

15. cloud userblock 'Bl_i' is active andconnected to chain

16. else

17. cloud userblock 'Bl_i' is inactive

18. End if

19. Return (generated blocks 'Bl')

20. **End for**

21. **End for**

End

Algorithm 1 provides different processing steps for user block generation. For each user, register their details to cloud service provider. The HAVAL cryptographic function is used to generate the hash for each user data and generates the blocks. After the block generation, active and inactive user blocks are identified through the Wilcoxon signed-rank test consensus mechanism based on the ranking procedure. As a result, active user blocks are connected to the chain. Other than the inactive cloud user block is removed, and it did not connect to the chain. Thus, the active cloud user blocks are generated for the validation process.

4.2 Improved schmidt-samoa Jaccard Extreme Learning Machine

After the block generation, the block validation is carried out before the data transaction to ensure the security of data transmission by detecting the different types of side-channel attacks. Th valid blocks are distributed on the network to participate transaction process. The proposed technique uses the Improved schmidt-samoa Jaccard Extreme Learning Machine in order to identify different types of side-channel attacks. The proposed classifier considered the following attacks cache-based attack, timing attack, data remanence attack, differential fault analysis, allocation-based side channels attacks and template attacks. A template attacks recovers cryptographic keys by using a matching "template" tool.

Extreme Learning Machine is an easy learning approach to offer better classification performance of the above said six different types of attacks and fast learning speed. It is a feed-forward

neural network, and it did not use any back-propagation algorithm for tuning the parameters. As a result, iterative learning process is not needed.

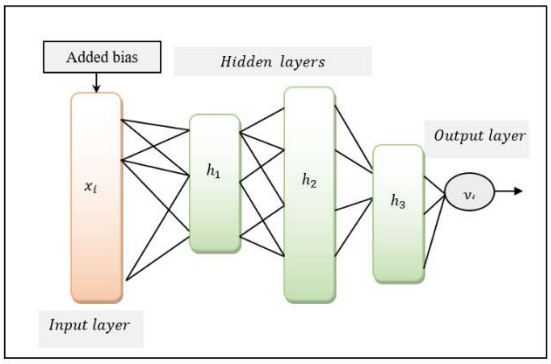

Figure 3: construction of extreme learning machine

Figure 3 portrays the construction of extreme learning machines with a feed-forward neural network. The structure includes input layer, multiple hidden layers, and output layer. Compared with the conventional deep neural network, extreme learning machines comprises two characteristics such as hidden layer parameters such as weights and the biases are arbitrarily initialized.

As shown in the above figure, let us consider that the input as a number of generated blocks 〖Bl〗_1, 〖Bl〗_2, 〖Bl〗_3… 〖Bl〗_n . First, the random weights matrix and bias are created for the input layer.

$$w_{ij} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \dots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} (6)$$

Where, w_ij denotes the weight matrix between input and hidden layer, and bias is generated as follows,

$$x_i = \sum_{i=1}^{n} [Bl_i * w_{ij}] + F \quad (7)$$

From (7), the activity of neurons at the input layer ‘w_ij’ denotes that the weight between input and hidden layer, bias function ’ F’, Then the input is fed into the hidden layers. In that layer, data encryption is performed using improved schmidt-samoa public key cryptosystem.

The proposed cryptosystem focuses on the issues of side-channel attack detection. O the contrary to Schmidt-Samoa public key cryptosystem, the improved version considers the four different prime numbers for the public key and private key generation. Schmidt-Samoa public key cryptosystem is an asymmetric key cryptosystem where the public key is used for data encryption and the private key is used for data decryption. In order to increase the possibility of side-channel attacks in the cloud, a proposed cryptosystem is proposed with four large prime numbers rather than two large prime numbers.

Let us consider four large prime numbers as c,d,u,v.

Calculate least common multiplier (LCM) between c-1 and d-1,

J=LCM (c-1) (d-1) (8)

Calculate least common multiplier (LCM) between u-1 and v-1,

K=LCM (u-1) (v-1) (9)

Then, multiply these two prime numbers

M=c*d (10)

N=u*v (11)

Compute inverse function,

M_n=M^(-1)modJ (12)

N_n=N^(-1) mod K (13)

Finally perform multiplication between M_n and N_n

G=M_n*N_n (14)

Based on above said values, the public key ‘H’ is generated as given below,

H=G mod J (15)

The private key is generated as given below,

d=H^(-1) mod J (16)

Let us consider the plain text ‘〖Bl〗_i’

β_c= 〖〖Bl〗_i〗^H mod H (17)

Where β_c denotes a cipher text, 〖Bl〗_i denotes plain text, H denotes a public key. For each user, dynamic secret word ‘DSW’ is generated for authorization.

Then the encrypted output is stored, and it sends to the receiver. On the receiver end, the decryption is performed with their private key. During the decryption, the blockchain server performs an authentication process, and the smart contracts are established to create communication between the users with certain rules without believing outer third parties. The rule of smart contracts defines only authorized users to access the data from the server and avoid access from the unauthorized.

When the users access the data, the server requests to enter the dynamic secret words ‘DSW’. Then it is verified using Jaccard coefficient with the entered DSW’ and already stored word DSW’.

$$\rho = \frac{DSW_S \cap DSW_E}{|DSW_S| + |DSW_E| - |DSW_S \cap DSW_E|} \quad (18)$$

Where, ρ denotes a Jaccard coefficient, 〖DSW〗_S indicates dynamic secret words at server, 〖DSW〗_E indicates a dynamic secret word entered, the intersection symbol ‘⋂’ designates mutual dependence between the secret words are statistically dependent, |〖DSW〗_S| denotes a cardinality of the set indicates number of words in the stored secret word, |〖DSW〗_E| denotes a cardinality of the set indicates number of words in the entered secret word. The coefficient returns the output values as 0 or 1.

The sigmoid activation function is applied to analyze the coefficient results,

δ= 〖(1+exp 〖(-ρ)〗)〗^(-1) (19)

Where δ indicates sigmoid activation function, ρ indicates coefficient results.

$$\delta = \begin{cases} 1 & , authorized \\ 0 & , unauthorized \end{cases} \quad (20)$$

The activation function ‘δ’ returns the output ‘1’ indicates that the user is classified as authorized, ‘0’ denotes unauthorized or attacks (i.e., cache attack, timing attack, data remanence attack, differential fault analysis attacks, allocation-based side channels attacks). In this accurate classification performed to identify the attacks and minimize the false positive rate.

Finally, authorized user decrypts the data and obtains the original data.

〖Bl〗_i= 〖β_c〗^d mod J (21)

Where, 〖Bl〗_i plain text, β_c cipher text, d private key, J denotes a prime number. This helps to improve the accuracy of attack detection and improve the secure transaction with higher confidentiality rate. The algorithmic steps for block validation are given below.

// Algorithm 2: block validation

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

Output: Improve attack detection accuracy

Begin

1. **Number of g**enerated blocks $Bl_1, Bl_2, Bl_3 \dots Bl_n$**in the input layer**
2. **For each g**enerated blocksBl_i
3. Assign the random weight ‘w_{ij}’ and bias F’
4. **End for**
5. **For each**blocksBl_i–**[hidden layer 1]**
6. Generate the pair of keys using (15) (16)
7. Perform encryption using (17)
8. Store cipher text into server
9. **If** user access the data then **[hidden layer 2]**
10. **Perform** dynamic secret words ‘*DSW*’ matching using (18)
11. Apply activation function ‘δ’ **[hidden layer 3]**
12. **If** ($\delta = 1$) **then**
13. User is classified as authorized
14. **else**
15. User is classified as attacks
16. **End if**
17. **End if**

18. For authorized user
19. Perform decryption using (21) to get original data **at the output layer**
20. Return (different types of attacks)
21. End for

End

The above algorithm explains the different procedures of attack detection in the cloud. Extreme Learning Machine consists of many layers to analyze the given input. The number of generated blocks for each user is considered the input layer. The weights are randomly allocated to each input as well as the bias function. Then the input is sent to the first hidden layer. For each generated user block, pair of keys is generated for secure data transmission. Then the encryption is carried out on the sender side to generate the ciphertext. After that, the validation is said to be performed based on the Jaccard coefficient in the second hidden layer. Then the sigmoid activation function is applied to provide the attack classification results. Finally, the authorized user decrypts the data to improve data confidentiality at the output layer.

5. EXPERIMENTAL SETUP

In this section, experimental assessment of the CCSJELB and existing PSASPIN [1] [2] are implemented in JAVA high-level class-based object-oriented programming language through CloudSim simulator. In order to conduct the experiment, a SCAAML dataset is used, and it taken from the https://github.com/google/scaaml/tree/master/scaaml_intro. The SCAAML dataset includes a more than 8000 samples instances are considered as input sample data, and it used for simulation. For the experimental consideration, the number of samples instances taken in the ranges from 800 to 8000.

6. PERFORMANCE RESULTS ANALYSIS

The relative result analysis of proposed CCSJELB and existing PSASPIN [1] Multi-input deep-learning model [2] are discussed in this section with different parameters such as communication overhead, throughput, attack detection accuracy and attack detection time, confidentiality rate with a different number of traces or input sample data.

6.1. communication overhead

It is evaluated as amount of memory consumed for block generation. This mathematically formulated as given below.

$$CO = \sum_{i=1}^{n} T_i * Mem[BG\ ] \qquad (22)$$

Where, the communication overhead 'CO' is measured based on the network traces 'T_i' involved in the simulation process and the memory consumed in performing block generation 'Mem[BG]' respectively. It is measured in terms of kilobytes (KB).

Table 1: comparison of communication overhead

Number of Traces	Communication overhead (KB)		
	CCSJELB	PSASPIN	Multi-input deep-learning model
800	192	280	240
1600	216	315	264
2400	264	350	300
3200	291.2	395	336
4000	328	435	364
4800	360	450	388.8
5600	392	480	448
6400	416	525	480
7200	446.4	585	511.2
8000	464	625	544

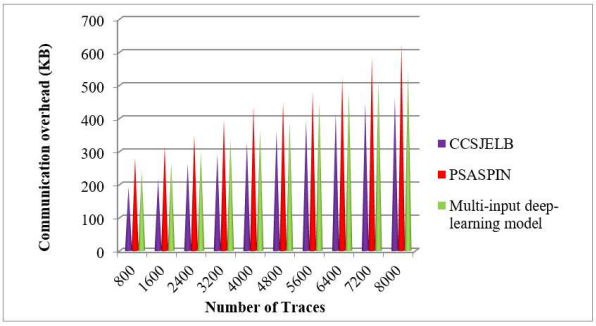

Figure 4: Performance analysis of communication overhead versus number of Traces.

Figure 4 depicts the performance analysis of communication overhead for varying numbers of traces taken in the range from 800 to 8000. As exposed in figure 7, the numbers of traces in the horizontal axis whereas the computation overhead observed in vertical axis. Among three methods, CCSJELB technique is comparatively provides better performance. Let us consider 800 traces. By applying CCSJELB technique, communication overhead was observed by 192 KB whereas the communication overhead of existing PSASPIN [1] Multi-input deep-learning model [2]were observed by 280 KB and 240KB respectively. Likewise, the nine various runs are performed for each method. As a result, different ten results are detected with respect to a different number of traces. Finally, the performance of CCSJELB technique is compared to the existing methods. As a result, the overall performance results indicate that the average outcome of communication overhead gets minimized by 25% compared to [1] and 13% when compared to [2]. This is because of applying a HAVAL cryptographic hash and Wilcoxon signed-rank test consensus mechanism. Then the server generates the blocks for each registered user by applying HAVAL cryptographic hash to generate the root hash of the user data. Then the Wilcoxon signed-rank test consensus mechanism is applied to identify the active or inactive user generated block. This helps to minimize the communication overhead in block generation.

6.2 Throughput

It refers to the maximum amount of data received at the receiver end. It is calculated using given mathematical formula,

$$Throughput = \sum \frac{T_{received}}{T_{sent}} * 100 \qquad (23)$$

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

Table 2: comparison of Throughput

Number of Traces	Throughput (%)		
	CCSJELB	PSASPIN	Multi-input deep-learning model
800	96.25	79	89.37
1600	96.12	79.5	89.68
2400	96.66	78	86.04
3200	94.06	77	85.78
4000	95.25	79.6	88.12
4800	96.04	81	89.16
5600	95.03	79.4	88.83
6400	94.14	77	87.89
7200	93.75	76.3	86.18
8000	93.12	75	84.2

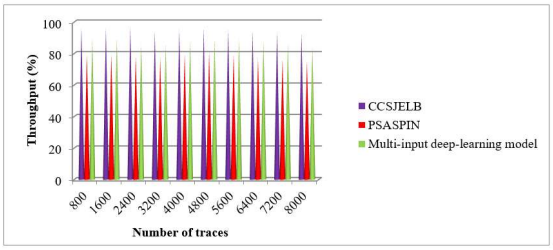

Figure 5:Performance analysis of throughput versus number of traces

Figure 5 portrays the performance outcomes of throughput of data transmission versus number of the traces. The obtained graphical outcomes demonstrate that the throughputs of all three methods are shown in figure 5. The graph indicates that the overall performance of throughput using CCSJELB technique is found to be improved than the existing methods. Let us consider the number of traces 800. By applying a proposed CCSJELB technique, throughput was found to be 96.25% whereas the throughput by applying two existing methods [1], [2] were found to be 79% and 89.37% respectively. Accordingly, the overall performance of throughput is identified by taking the average of ten comparison results. The observed results indicate that the throughput is considerably improved by 22% when compared to [1] and 9% when compared to [2] respectively. This is because of applying the Wilcoxon signed-rank test consensus mechanism for identifying the active or inactive user block. These

active blocks are connected to the chain to perform the data transaction which results increases the throughput.

6.3 Attack detection accuracy

It is measured as the ratio of number of traces that has identified as different types of side channel attack to the total number of traces. It is mathematically formulated as given below.

$$SCAD_{acc} = \sum_{i=1}^{n} \frac{T_{ad}}{T_i} * 100 \quad (24)$$

Where, 〖SCAD_acc denotes a side-channel attack detection accuracy, T_ad denotes a network traces that has detected the type of attack accurately, T_i denotes a total number of network traces. It is measured in terms of percentage (%).

Table 3: comparison of attack detection accuracy

Number of Traces	Attack detection accuracy (%)		
	CCSJELB	PSASPIN	Multi-input deep-learning model
800	96.25	80	88.75
1600	95.31	78.45	86.87
2400	94.37	77	85.5
3200	93.12	76	84.68
4000	94	78.55	87.8
4800	95.41	80	88.02
5600	94.28	78.35	87.87
6400	93.12	76	86.9
7200	92.91	75.25	85.06
8000	92.65	74	84.81

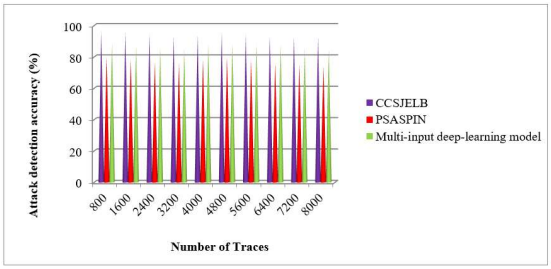

Figure 6: Performance analysis of attack detection accuracy versus number of traces

Figure 6 portrays the performance outcomes of attack detection accuracy with respect to number of traces. Accuracy is measured using proposed CCSJELB and existing PSASPIN [1] Multi-input deep-learning model [2]. In the above figure, horizontal axis represents the number of traces, and the vertical axis point outs the attack detection accuracy. The observed result indicates that the proposed CCSJELB technique outperforms well than the other methods, [1] and [2]. However, with simulations conducted with 800 traces, the accuracy was observed to be 96.25% using proposed CCSJELB technique and 80% and 88.75% using existing two methods [1], and [2] respectively. The overall performance of ten results designates that the of attack detection accuracy using proposed CCSJELB technique is considerably improved by 22% when compared to [1] and 9% when compared to [2] respectively. With this measure, the accuracy is improved using CCSJELB technique due to the application of extreme learning machine classifier. The improved schmidt-samoa cryptography performs encryption and decryption. The Jaccard coefficient is applied for matching the dynamic secret word. The sigmoid activation function is used to identify the authorized or attackers with higher accuracy.

6.4. False positive rate:

It is defined as the ratio of number of traces that has detected attack incorrectly to the total number of traces. It is mathematically formulated as given below.

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

Where, FPr denotes a false positive rate, T_ad (incorrectly) denotes a network traces that has detected the type of attack incorrectly, T_i denotes a total number of network traces. It is measured in terms of percentage (%).

Table 4: comparison of false positive rate

Number of Traces	False positive rate (%)		
	CCSJELB	PSASPIN	Multi-input deep-learning model
800	8.75	15	11.25
1600	9.68	16.31	13.12
2400	9.79	17.29	14.5
3200	8.28	16.40	15.31
4000	10.37	17	12.2
4800	10.70	15.10	12.18
5600	11.16	15.78	13.92
6400	10.96	14.45	12.89
7200	8.88	15.62	13.12
8000	9.37	14.37	11.56

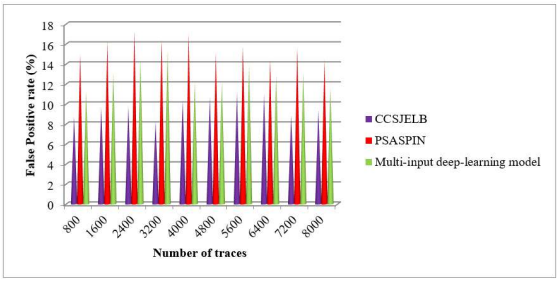

Figure 7 Performance analysis of false positive rate versus number of traces

Figure 7 illustrates performance analysis of false positive rate with respect to three different methods namely CCSJELB, existing PSASPIN [1] Multi-input deep-learning model [2]. As shown in figure 7, the performance of false positive rate is smaller using CCSJELB technique than the existing methods. This is due to the reason that all the instances accurately classified as a different types of side channel attacks. However, with experiment is conducted with 800 traces, the false positive rate was observed to be 8.75% using CCSJELB technique and 15% and 11.25% using [1], and [2] respectively. This is because of applying an extreme learning machine classifier for accurately detects the attacks and minimizes the incorrect classification with the help of Jaccard coefficient and sigmoid activation function. The comparison results indicate that the false positive rate is considerably minimized by 37% and 24% when compared to [1], and [2].

6.5. Side-channel attack detection time:

It refers to amount of time consumed for detecting different types of side-channel attacks in cloud data transaction. The attack detection time is measured as given below.

$$SCAD_{time} = \sum_{i=1}^{n} T_i * Time\ [AD] \quad (26)$$

Where, 〖SCAD〗_time denotes a side-channel attack detection time, T_i denotes a total number of traces, Time [AD] denotes a time consumed in detecting the side-channel attacks. It is measured in terms of milliseconds (ms).

Table 5: comparison of attack detection time

Number of Traces	Attack detection time (ms)		
	CCSJELB	PSASPIN	Multi-input deep-learning model
800	528	880	680
1600	688	920	832
2400	792	980	864
3200	848	1085	960
4000	980	1245	1120
4800	1017.6	1355	1152
5600	1181.6	1405	1288
6400	1222.4	1485	1344
7200	1231.2	1535	1368
8000	1240	1585	1440

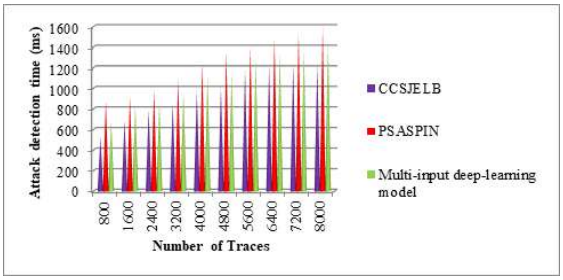

Figure 8 Performance analysis of attack detection time versus number of traces

Figure 8 given above demonstrates the graphical illustration of attack detection time with respect to a number of traces. As shown in figure 8, 'x 'axis represents the number of traces, and 'y' axis represents the attack detection time. The graphical demonstrates that the s attack detection time of

different methods gets increased while increasing the number of number of traces. But comparatively, the attack detection time of the CCSJELB technique is found to be decreased when compared to existing methods. The reason behinds the minimum amount of time owing to the application of dynamic secret word matching through the Jaccard similarity.. This helps to minimize the attack detection time. From the validation result, the attack detection time using CCSJELB technique is said to be comparatively reduced by 23% and 13% than [1] and [2].

6.6 Confidentiality rate:

It refers to number of data accessed by authorized user. The confidentiality rate is measured as given below.

$$CR = \sum_{i=1}^{n} \frac{Traces\ accessed\ by\ AU}{T_i} \quad (26)$$

Where, CR denotes a confidentiality rate, T_i denotes a total number of traces, Traces accessed by AU denotes a traces accessed by authorized user. It is measured in terms of percentage (%).

Table 6: comparison of confidentiality rate

Number of Traces	Confidentiality rate (%)		
	CCSJELB	PSASPIN	Multi-input deep-learning model
800	95.62	78.12	87.5
1600	95.93	78.43	88.12
2400	95.83	77.29	85.20
3200	93.75	76.71	84.37
4000	94.12	77.5	87.5
4800	95.83	80.33	88.54
5600	94.64	78.78	86.07
6400	93.75	76.75	86.95
7200	92.77	74.52	85.48
8000	92.81	74.43	83.2

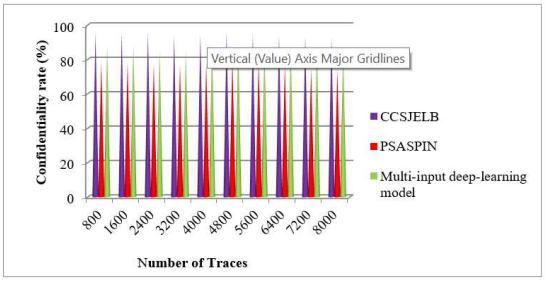

Figure 9 Performance analysis of attack detection time versus number of traces

Figure 9 demonstrates the evaluation results of confidentiality rate using three different methods, CCSJELB, existing PSASPIN [1] Multi-input deep-learning model [2] respectively. The confidentiality rate is computed with respect to a distinct number of traces ranging between 800 and 8000. Compared to other existing methods, the proposed CCSJELB technique provides enhanced performance when compared to other related methods. This is because of the CCSJELB technique uses improved schmidt-samoa Jaccard extreme learning machine-based block validation. As a result of block validation, authorized user access and perform decryption and avoid the attacks. Different performance results are observed for each method. With this, the overall confidentiality rate using the CCSJELB technique is said to be improved by 22% compared to [1] and 10% compared to [2].

7. JUSTIFICATION OF THE STUDY

The CCSJELB methodology is deemed appropriate for this study owing to its innovative security enhancement focus, designed to resolve a distinct problem and side channel attack in cloud. This investigation focused on the design of an integrated security framework incorporating the Blockchain technology in Extreme learning machine model. This investigation strives to contribute to this domain by formulating a framework to increase the security of cloud user data, hence, the CCSJELB methodology was chosen. On the contrary to existing works, the proposed CCSJELB methodology uses the Wilcoxon signed-rank test consensus mechanism to separately produce active and inactive user generated blocks. This aids to improve the throughput of data transmission. With the help of Schmidt-samoa Jaccard Extreme Learning Machine, block validation for each cloud user was made through the encryption and

decryption process to increase the data confidentiality.

8. CONCLUSION

In this paper, a novel CCSJELB technique is developed with the goal of achieving side-channel attack detection with higher accuracy for increasing the security of cloud data access. The above said objective is attained by the contribution of block generation and validation process. In the block generation step, the HAVAL cryptography technique is contributed to generate the hash value for each user. Then the Wilcoxon signed-rank test consensus mechanism is designed for active or inactive user block identification for efficient data transmission to achieve higher throughput and minimize the attack detection time. This also reduces the communication overhead. Subsequently, an improved Schmidt-Samoa Jaccard Extreme Learning Machine is contributed for block validation to detect the different types of attacks and provide higher security during the data transmission. Also, Jaccard coefficient and sigmoid activation function are applied to accurately classify the attack in cloud for side channel attack detection. With this, data confidentiality rate, integrity and reliability of data are enhanced. An inclusive experimental measurement is carried out with respect to a number of trace instances and compares the results of the proposed traces with two conventional algorithms. The observed numerical results confirmed that the proposed CCSJELB technique outperforms well by achieving higher attack prediction accuracy, throughput, confidentiality rate, and minimizing the time as well as overhead than the other existing methods. Though the technique finds the side channel attacks, the overfitting problem is not solved. Future work can design countermeasures to against deep-learning based side channel attacks for addressing overfitting concern. Also, future work plan further to investigate the multi-leakage phenomena by training new models on other attack points.

REFERENCES

[1] Mohamud Ahmed Jimale, Muhammad Reza Zaba, Miss Laiha Binti Mat Kiah, Mohd Yamani Idna Idris, Norziana Jamil, MoesfaSoeheila Mohamad, Mohd SaufyRohmad, “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] Fanliang Hu, HuanyuWang,Junnian Wang, “Multi-Leak Deep-Learning Side-Channel Analysis”, IEEE Access, Volume 10, 2022, Pages 22610 – 22621. DOI: 10.1109/ACCESS.2022.3152831

[3] 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, Volume 17, Issue 3, 2022, Pages 377-393. https://doi.org/10.1049/ise2.12102

[4] Rabin Y. Acharya, Fatemeh Ganji and Domenic Forte, “Information Theory-based Evolution of Neural Networks for Side-channel Analysis”, IACR Transactions on Cryptographic Hardware and Embedded Systems, Volume 2023, Issue 1, Pages 401–437. DOI:10.46586/tches.v2023.i1.401-437

[5] Maria Mushtaq, Jeremy Bricq, Muhammad Khurram Bhatti, Ayaz Akram, Vianney Lapotre , Guy Gogniat, and Pascal Benoit, “WHISPER: A Tool for Run-Time Detection of Side-Channel Attacks”, IEEE Access, Volume 8, 2020, Pages 83871 – 83900. DOI: 10.1109/ACCESS.2020.2988370

[6] SoroorGhandali, SamanehGhandali, 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

[7] Alexander Kulow, Thomas Schamberger, Lars Tebelmann, Georg Sigl, “Finding the Needle in the Haystack: Metrics for Best Trace Selection in Unsupervised Side-Channel Attacks on Blinded RSA”, IEEE Transactions on Information Forensics and Security, Volume16, 2021, Pages 3254 – 3268. DOI: 10.1109/TIFS.2021.3074884

[8] Bo-Yeon Sim, Aesun Park, Dong-Guk Han, “Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER Using the Side-Channel Leakage of Barrett Reduction”, IEEE Internet of Things Journal , Volume 9, Issue 21, 2022, Pages 21382 – 21397. DOI: 10.1109/JIOT.2022.3179683

[9] Dongjun Parka , GyuSang Kima , DonghoeHeoa , Suhri Kima , HeeSeokKimb, 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, 2021,

Pages 36-40. https://doi.org/10.1016/j.icte.2021.01.013

[10] S. Gnanavel , K. E. Narayana, K. Jayashree, P. Nancy, and Dawit Mamiru Teressa , "Implementation of Block-Level Double Encryption Based on Machine Learning Techniques for Attack Detection and Prevention", Wireless Communications and Mobile Computing, Hindawi, Volume 2022, July 2022, Pages 1-9. https://doi.org/10.1155/2022/4255220

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

[12] Feng Ni, Junnian Wang, Jialin Tang, Wenjun Yu, Ruihan Xu, "Side channel analysis based on feature fusion network", PLoS ONE, Volume 17, Issue 10, 2022, Pages 1-20. https://doi.org/10.1371/journal.pone.0274616

[13] Rashidah Funke Olanrewaju, Burhan Ul Islam Khan, Miss Laiha Mat Kiah, Nor Aniza Abdullah and Khang Wen Goh, "Decentralized Blockchain Network for Resisting Side-Channel Attacks in Mobility-Based IoT", Electronics, Volume 11, Issue 23, 2022, Pages 1-22. https://doi.org/10.3390/electronics11233982

[14] Donggeun Kwon, Seokhie Hong, Heeseok Kim, "Optimizing Implementations of Non-Profiled Deep Learning-Based Side-Channel Attacks", IEEE Access, Volume 10, 2022, Pages 5957 – 5967. DOI: 10.1109/ACCESS.2022.3140446

[15] Guanxiong Ha, Hang Chen, ChunfuJia,Mingyue Li, "Threat Model and Defense Scheme for Side-Channel Attacks in Client-Side Deduplication", Tsinghua Science and Technology ,Volume 28, Issue 1, 2023, Pages 1 – 12. DOI: 10.26599/TST.2021.9010071

[16] Yiwen Gao and Yongbin Zhou, "Side-Channel Attacks With Multi-Thread Mixed Leakage", IEEE Transactions on Information Forensics and Security, Volume 16, 2020, Pages 770 – 785. DOI: 10.1109/TIFS.2020.3023278

[17] Naila Mukhtar, Apostolos P. Fournaris, Tariq M. Khan, Charis Dimopoulos and Yinan Kong, "Improved Hybrid Approach for Side-Channel Analysis Using Efficient Convolutional Neural Network and Dimensionality Reduction", IEEE Access , Volume 8, 2020, Pages 184298 – 184311. DOI: 10.1109/ACCESS.2020.3029206

[18] Ezinam Bertrand Talaki , Olivier Savry, Mathieu Bouvier Des Noes and David Hely, "A Memory Hierarchy Protected against Side-Channel Attacks", Cryptography, Volume 6, Issue 2, 2022, Pages 1-17. https://doi.org/10.3390/cryptography6020019

[19] Dajiang Chen, Zihao Zhao, Xue Qin, Yaohua Luo, Mingsheng Cao, Hua Xu, and Anfeng Liu, "MAGLeak: A Learning-based Side-channel Attack for Password Recognition with Multiple Sensors in IIoT Environment", IEEE Transactions on Industrial Informatics , Volume: 18, Issue 1, 2022, Pages 467 – 476. DOI: 10.1109/TII.2020.3045161

[20] 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