WEB BASED DATABASE SECURITY IN INTERNET OF THINGS USING FULLY HOMOMORPHIC ENCRYPTION AND DISCRETE BEE COLONY OPTIMIZATION

R. Joshua Samuel Raj[1], M. Viju Prakash[2],T.Prince[3], K. Shankar[4*],Vijayakumar Varadarajan[5], Fredi Nonyelu[6]

[1]Professor, Department of Information Science and Engineering, CMR Institute of Technology, Bengaluru, India

[2]Assistant Professor, Department of Computer Science, College of Science, Knowledge University, Erbil, Kurdistan Region, Iraq

[3]Lecturer, Faculty of Computing and Software Engineering,Arba Minch Institute of Technology, Arba Minch University, Ethiopia

[4]Department of Computer Applications, Alagappa University, Karaikudi, India

[5]School of Computer Science and Engineering, The University of New South Wales, Australia

[6]Chief Executive, Briteyellow Ltd, United Kingdom

Email: joshua.r@cmrit.ac.in[1], viju.prakash@knowledge.edu.krd[2], prince.thomas@amu.edu.et[3], drkshankar@ieee.org[4*] (corresponding author), v.varadarajan@unsw.edu.au[5] ,fredi.nonyelu@briteyellow.com[6]

DOI: https://doi.org/10.22452/mjcs.sp2020no1.1

ABSTRACT

Web applications are utilized on an extensive scale across the globe and it handles sensitive individual information of users. Structured Query Language (SQL) Data Inference (DI) and injection are procedures that abuse a security defenselessness occurring in the database layer of an application. This research article focuses on website page database security with the help of optimization and encryption methods for Web of Things Environments. Initially, the selected queries in webpage application are injected as per Discrete Bee Colony Optimization (DBCO) procedure. After the Proxy filtering, the injection prevention model is utilized, the injected data with various queries of different special characters are utilized. At long last, the attack gets detected depending on the user query with the assistance of query tree mechanism. Besides, an effective Fully Homomorphic Encryption (FHE) encryption is proposed in the study. From the implementation results, it is to be noted that the proposed method achieved 93.56% security level for the prevented webpage implication-based databases. The effect on the businesses must be comprehended to decrease the risk involved in SQL and DI injection assaults.

Keywords: ***Big data, webpage database, injection, prevention, optimization, attack, security***

1.0 INTRODUCTION

In recent times, web applications are widely utilized in different kinds of organizations, businesses or commerce. This can attributed to its reliable nature and efficient outcome solution to various challenges involved in the communication over the Web of Things Environments [1]. Similar to some evolving structures, it hasalsogained high-interest databases [2] because of the malicious users who wait to exploit their weaknesses and imperfections for their personal goals [3]. The security of databases is important and a developing concern considering the number of occurrences reported constantly [4]. The attack marks at injection focuses contain the examples of Structured Query Language (SQL) tokens and images as SQL-Injection Attacks (SQLIA)-positive while substantial web requests tend to appear as expected information from the application [5]. The SQL injection refers to the process in which the attacker act upon the database by embedding a progression of SQL statements in the query task [6]. Likewise, there are different assets on the web that disclose point-by-point methodology to ordinary users in the best way to attack web applications using various sorts of all-in-all attacks and SQL injection specifically [7].

To overcome the challenges involved in traditional SQL injections, it is easily detected by following a lot of procedures and methodologies [8]. By utilizing different Database Management Systems (DBMS) [9], the developed organizations experience the assurance and security for the cloud storage data [10]. To secure data and prevent the unauthorized access from unprivileged entities, DBMS is engaged with access control [11]. Also, when a malicious user deduces some secured or private data, the inference occurs without directly accessing it [12]. The database security is mainly dependent on its availability, integrity, and confidentiality. To be specific, the term ‘availability’

deals with the avoidance of both hardware and software errors and providing access to the user required information whenever it is needed [13]. Encryption and decoding of the data have turned out to be the ideal approach to attain privacy and integrity of information [14].

With innovations developing every day, the real challenge is the development of novel threats as well as vulnerabilities [15]. In the recent days, the web application is generally utilized in different applications thanks to its reliable and proficient answer for the difficulties. After decryption, the computing operations are applied to the user query data. After applying the tasks, the customer can again encode the outcome and store it on the cloud. These processes such as decoding the information, applying tasks and still encrypting the outcome are overhead system [16]. The information is recovered on the server side and private keys are recouped by accessing the database.

To optimize the security of the model, private keys are occasionally refreshed through mixing procedures such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [17]. Diverse encryption algorithms conclude in selecting the best method based on how it can outperform with the issues of computational cost as well as system security [18]. In this paper, the authors focus on website page database security with the help of optimization and encryption method for Web of Things Environments. Initially, the selected queries in webpage application are injected using Discrete Bee Colony Optimization (DBCO) procedure. After Proxy filtering, the injection prevention model is utilized, the injected data with different queries with various special characters are utilized. At long last, the attack is detected based on user query with the assistance of query tree mechanism to enhance the security level of attack detection and prevention. Besides, an effective Fully Homomorphic Encryption (FHE) encryption is proposed. The outcomes exhibited that the proposed DBCO and FHE model accomplishedthemost extreme precision of 96.22% contrasted with existing algorithms.

The remainder of this paper has been organized as following: Section 2 describes the related works in literature. Section 3 describes the problem statement of the current system while the proposed methodology (query injection & security) is detailed in section 4. The implementation of security results is discussed in section 5 and finally the work is concluded with future scope in section 6.

2.0 LITERATURE REVIEW

It is highly challenging to detect the SQL injection attacks in cloud database security because of its extreme heterogeneity of the attack vectors. One of the novel approaches to detect the SQL injection attack is the graph of tokens whereas centrality measures are trained by Support Vector Machine (SVM), as mentioned by DebabrataKar et al[19]. Though the current web applications are fundamentally focusedaround PHP and MySQL, the methodology can be effortlessly ported to different stages. A Gap-Weighted String Subsequence Kernel technique was actualized to recognize the consequences of shared characters between query strings to yield a similarity metric, by Paul R. McWhirter et al., [20]. At last, SVM algorithm was trained by similarity measure (between known and unknown query strings utilized in classification). By social occasion, for all component information from the query strings, there is no need of extra data from the source application.

An effective SQL injection attack poses a genuine risk to the database, web application, and the whole web server. This attack is normal since it controls the information going fromweb application to the database servers through web servers and it can change or uncover the database substance, as investigated by Asish Kumar Dalai and Sanjay Kumar Jena [21] and Piyush A. Sonewar et al. [22]. With the help of SQL, the injection attackers can take away the private data, as opined by Rajashree A. Katole et al. in 2018 [23]. All SQL injection attack types, systems and devices can recognize or keep these attacks under control using hashing and encryption techniques. To anticipate SQLIA by utilizing encryption in stored database, the Advanced Encryption Standard (AES) Encrypted, and secret phrase were utilized in the previous study conducted by Dinu P S et al. and, DeeviRadha Rani et al. in 2018 [24, 25]to enhance the validation procedure with least overhead. Christine Basta et al., [26] presented a Genetic Algorithm-based fuzzy system for detecting SQLI where the accuracy is a prerequisite one along with the mandate of learning and adaptability of the attained rules. From the reviewed methods, it is understood that many models have been presented earlier and available in the literature. Though various models have been proposed, there is a need to develop an effective security model for web applications.

3.0 PROBLEM STATEMENT OF CURRENT STRATEGY

The attempts of SQL injection and DI attack are unbeaten only when these are infused with the web server database. In existing literary works, many administrations are infused to secure the webpage. The way that drives these new types of SQL injection attacks often do not qualify as completely new though some are already observed attack vectors

which are of minor and imaginative varieties. In the existing filtering model, the removal of special characters, operators and brackets remains the final solution. This is to completely standardize the SQL questions into a text form that is reasonable for applying document so as to calculate the similarity measure. Additionally, for security, different encryption proceduressuch as AES, DES, RSA, andECCwere utilizedin this study [31-37]. The security parameters have not been instated while the attacker can access the sensitive content without checking the personality of the user. In this way, the attacker misuses this defenselessness to infuse SQL injection model.

4.0 INNOVATIVE DATABASE SECURITY MECHANISM

Modern databases turned out to be highly robust with refined structures in the course of evolution of multiple decades. Databases, like all other developing structures, gained attraction in the recent times. A secure database in cloud injectsmulti-attacks in web servers while in the current research work, query attack and DI attack were utilized by the optimization technique. The database server side, being related with a distributed cloud platform, to offer an controlling security system to guarantee the secure implementation of every requested queries with no database hacking. Web server security model generates dummy SQL query to the database server while the proposed DBCO technique performs this generation. After embeddings of the SQL and DI in web data, the function of proxy filtering method is to remove the unwanted server information to secure the web server database. Thepurpose of this filtering technique is to recognize special form like characters, ID, and so on. Finally the admin detects the query from the databaseand identifies the query user with the help of query tree mechanism. To enhancethesecurity,the current study utilized FHE algorithm to encrypt and decrypt the identified query. Its primary function is to comparethe security tool from the original one using similarity measure that filtersthe malicious queries.

4.1 SQL Injection Problem

.

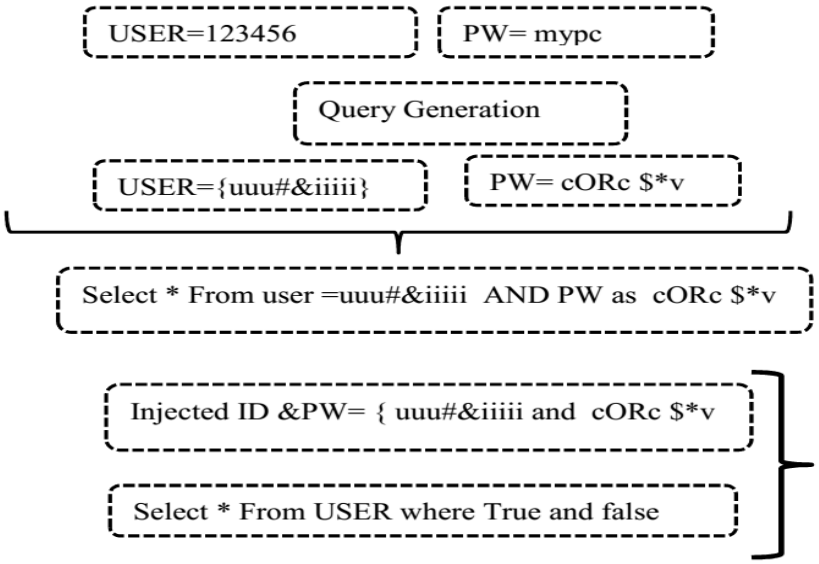

Fig 1: Representation for SQL Injection Model

SQLIA is said to be ancode injection attack system which is generally utilized to attack websites. In this system, an attacker infuses some SQL codes instead of the original codes to access the database. This injection model is shown in the figure 1. An integrated static and dynamic analyzing procedures exploits the advantages of static as well as dynamic analyses methodologies to analyze the SQL injection attacks. It examinaes the website pages and produces the SQL queries for sampling the simulation outcome of this model.

Attack Injection: Any malicious user can enter malevolent information in the fields of username and secret word. The simple command entered by the user can pulverize the entire database or it can prompt the administration about the possible interruption. Numerous web pages take input from users, for example, search terms, feedback content or username and secret word, and utilize such information to fabricate a SQL question which is then passed to the database.

Query: Choose* Customer ID= {2}

SQL Query=Choose*From Customer ID=4121=Valid

Valid Query= Choose attacked customer ID {4121} = 7896

4.2 Database Interferences Problem

It is challenging to sort the database inference to a group of data security attacks. This might be attributed to the way that it leverages the human personality and a logical approach to derive secure data. The database security is dependent on availability, integrity, and confidentiality, whereby the availability deals with the avoidance of hardware and software. Integrity is the information whenever needed with the assurance from unauthorized data access and illegal change; and confidentiality with the insurance of unauthorized data to webpage database security.

4.3 Query preprocessing

In query preprocessing, the front-end restricts the inquiries so as to confine the aggregate results. Each item, being chosen, is checked to guarantee that only a supported aggregate function is used for each. Additionally, a count is added for each selected item to assist in later analysis. From this preprocessing system, the non-valuable queries are neglected to the webpage data security model. Moreover, by decreasing the number of ON operations, the performance of query processing can be greatly enhanced. When a lot of unknown people access the data, the chances of data threats increase. Furthermore, the database attacks are done to make huge money by offering sensitive information in illegal ways.

4.4 Query optimization

Once the preprocessed SQL and DI queries are optimized for security, the query optimization is performed by DBCO. It is frequently a standout amongst the most neglected, yet it possesscritical angles while misusing blind SQL injections. It is not only an optimized SQL injection exclusivelyproduce faster results, it also reduces the network congestion and accordingly provides lower burden tothe server. Without legitimate records, the SQL inquiries can cause table outputs, which causes either performance or locking issues. Besides, the optimization of infused inquiries is intricately clarified in the following sections.

4.4.1 Query optimization process

Generally, the hackers access the web page via login page or the user constrained input. Then they insert a crafted string to increase the unapproved access to the database. An attack model, which exploits access pattern, reveals the boundaries of the encrypted and decrypted ranges of particular injected queries. It is scientifically expressed as follows

$$AQ \Rightarrow \quad \arg \quad \sum \left(\left| S_{s_i}, Q_{q_j} \right| - \left| C_i \cap C_j \right| \right) \quad (1)$$

$$\{ | a_1, q_1 |, ... | a_x, q_x | \}^2$$

This expression explained as $\{ | a_1, q_1 |, ... | a_x, q_x | \}^2$ is a selected range of preprocessed queries along with the proposed output function C. It should be noted that this optimization problem trivially incorporates individual cardinalities of the tuple sets of valuables, i and j.

4.4.2 Bee Colony Optimization (BCO)

Naturally, Bee swarm behaviouris highly characterized via autonomy, distributed functioning and self-organizing. Bees generally hunt for food in the fields nearby their hive. These bees gather and aggregate the food for later use by other different bees which inhabit the hive. Regularly, in the initial step, some scouts inspect the region. The bee cans (a) abandon the food source and turn out at the uncommitted follower. It further keeps on scavenging the food source without enrolling fellow nest mates or dance. In this way, the nestmateswhichpreviously arrived at the food source [28] are selected. The BCO search was performed with numerous iterations until some predefined stopping criterion is attained.

Discrete Process

The discrete model deals with issues in which one need to pick the optimal solution from a limited number of potential outcomes in query optimization process. Optimality is characterized regarding a standard capacity, which is to be limited or amplified, because of target work.

Proposed DBCOmodel

The proposed BCO model was analysed and proved through excellent swarm based optimization process. For the purpose of maximizing the objective function, the discrete function was hybridized with BCO. It isevident that the standard BCOcannot be utilized to solve discrete problems directly, since its positions are real-valued.

(i) Constraints and Objective Function

Let us assume that $|a_1, q_1|$ and $|a_2, q_2|$ are the ranges of preprocessed SQL and DI attack queries (Figure 2). Then the possible outcome value ranges of the selected queries are arranged. One of the ranges is entirely subsumed by the other while the ranges overlap. A pair of queries chose the attacker with the output C1=C2. Then the raised queries are assumed to indicate the same range and hence falls into the first scenario.

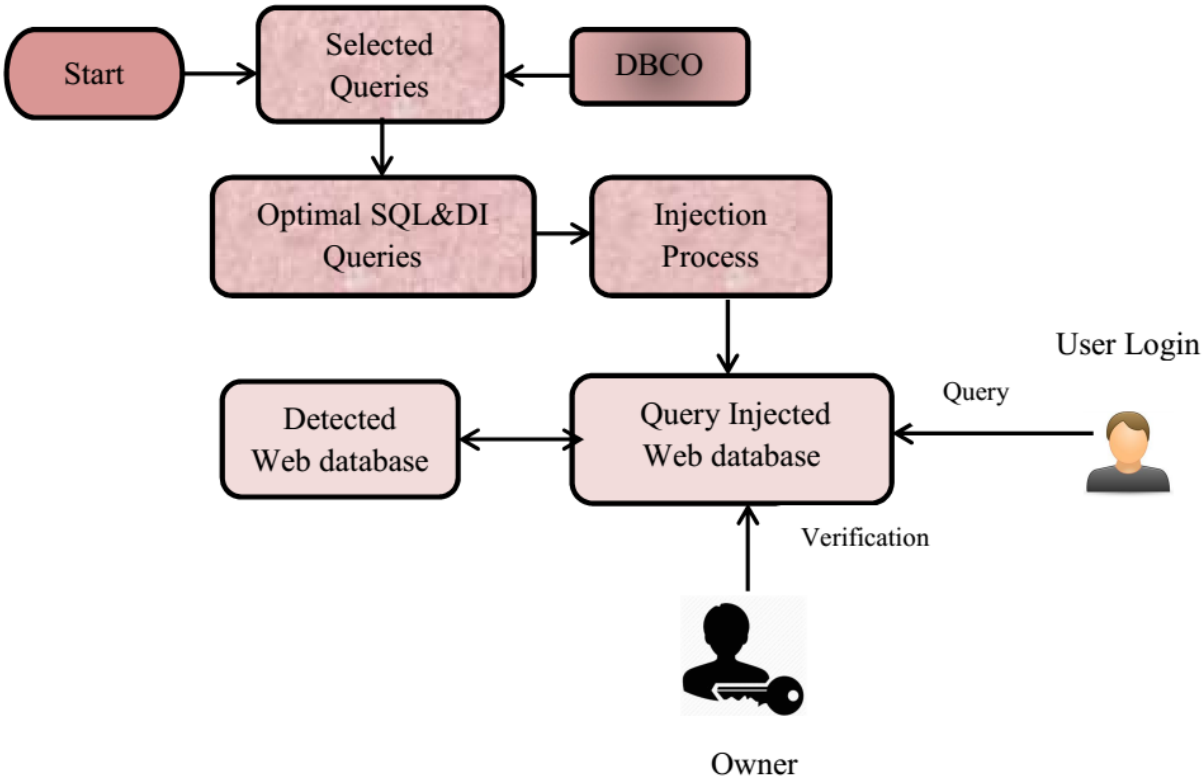

Fig 2: Optimal Query Selection for Security

(ii) Objective

With regards to the fitness evaluation of database security, the chosen preprocessed inquiries are changed over too many buckets. The reason behind bucketization is to set the information to sever and alter over the plain information so that the attacker sees the arrangement of tuples with each chosen queries. The function is characterized as follows

$$Objective = MIN\left(\begin{matrix}\arg & \sum((X_\alpha, Y_\alpha) - C_i)\\ \{(X_{\alpha 1}, Y_{\alpha 1})....(X_{\alpha l}, Y_{\alpha l})\}\end{matrix}\right)^2 \quad (2)$$

Here X_β and Y_β are assigned values for bucket identifier process with proposed output values. As mentioned above, the solution to the problem approximates the bucket boundaries. Therefore, it does not guarantee the revelation of the range queries exactly. The following procedures update these fitness minimization DBCO solutions.

(iii) Updating procedure

In the updating process of bee swarm optimization, the bees are represented by a selected range of queries which collaboratively solve the complex combinatorial optimization problems. This new 'honey bee solution updating' consider two essential procedures such as forward and backward passes. Here every honey bee investigates the pursuit

of space with number of moves. The honey bees go again to the home and begin the second stage i.e., the alleged backward pass. In the backward pass, every single counterfeit honey bee shares the data about their answers.

Forward Pass: Enable B_e to fly to the hive to choose the best bee solution in a particular stage S_n.

Backward Pass: Permit honey bees to trade data about nature of the partial solutions made and to choose whether to desert the made partial solution and turn out to be an uncommitted devotee again; keep on growing a similar partial solution without enlisting the home mates.

Steps

Assign the query range with a discrete process
Evaluate objective Function
I=i+1
All Bess are arrived in hive// Initiate forward process
Find the objective function to sort the bee solution
The objective As Minimum
End
Update the backward process
Choose a recruiter using the roulette wheel, to find the range $lb - rb = p$
New Variable updating Model is $N = b_i + r(0,1)\min(p, lb - b_i)$
If the stopping condition attained
End
Else// i=i+1

From the above procedure,theoptimal inquiries' zone is chosen for the injection model. This procedure considered fitness as the least contention values. Web servers, which provide client administrations, are typically associated with profoundly-sensitive information containing backend databases.

4.5 Proxy Server and Filtering

For the optimal quires taken into account, filtering model is considered to evacuate the uncommon characters which utilize proxy filtering model and are transferred to the server. This server can likewise be used to expand the security for a business. According to the previous study [27], a server can give network address interpretation which makes the individual users and PCs on the system mysterious at the time of utilizing internet. The ordinary strings are to be attacked while the characters get changed into hexadecimal, ASCII and Unicode.

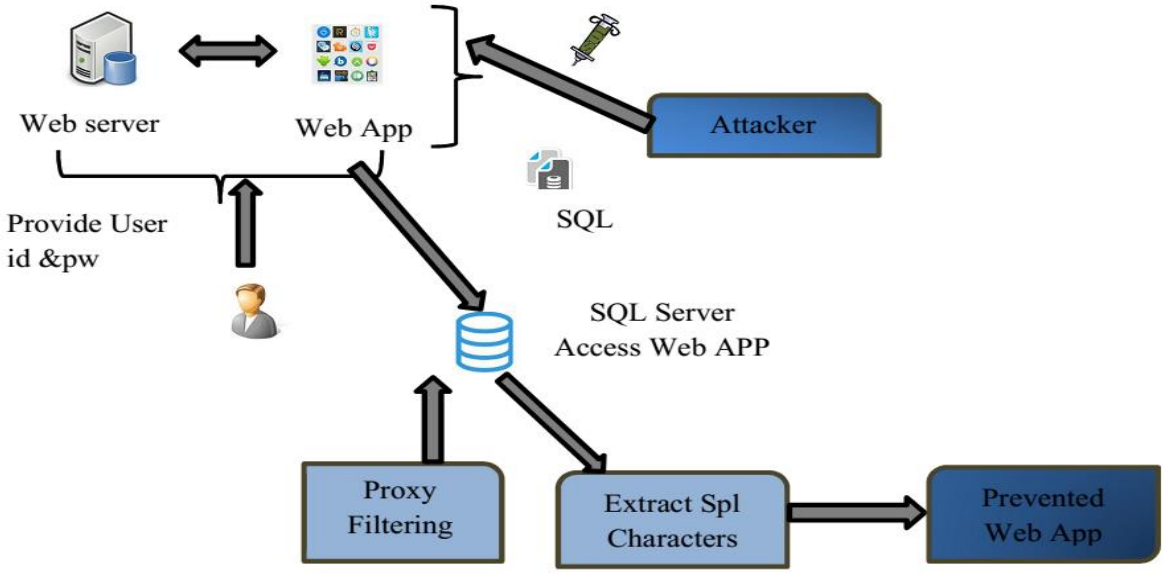

Fig 3: Injection and Prevention in Webpage application

As a result of this, the input inquiry is gotten away from the channel which checks the question for some awful character results. This filtering captures the web demands at a proxy for ideal SQL and DI detection and counteractive action as shown in the figure 3. It has the upside of capacity presence to decode the obfuscated internet traffic for careful investigation.

4.6 Webpage database security

For security in webpage database, the filtered SQL and DI are utilized to access the databases' basic Web applications by adding sudden SQL [29] to the web application code. The augmenting bar that conveys such web applications start running the comparing bar for a number of objective attacks. To enhance the security, efficiency, usefulness, andeasy usage of the database, FHEis utilized.

4.6.1 Fully Homomorphic Encryption

The complete homomorphic encryption is carried out on several nodes to minimize the computation time. FHE [30] is a kind of encryption that enables specific calculations to be performed on cipher query and restore an encoded outcome. The decoded resultant is equivalent to that of the aftereffect of directing the activity on plain query. It permits the calculation on encrypted data that lacks to precede to decryptionandthe following framework occurs, if the customer decodes the outcome, which is in the encrypted data. Figure 4 illustrates the complete process. In webpage security, the typical questions that are re-infused to enhance the security are four processes such as Key Initialization, encryption, decryption, and analysis.

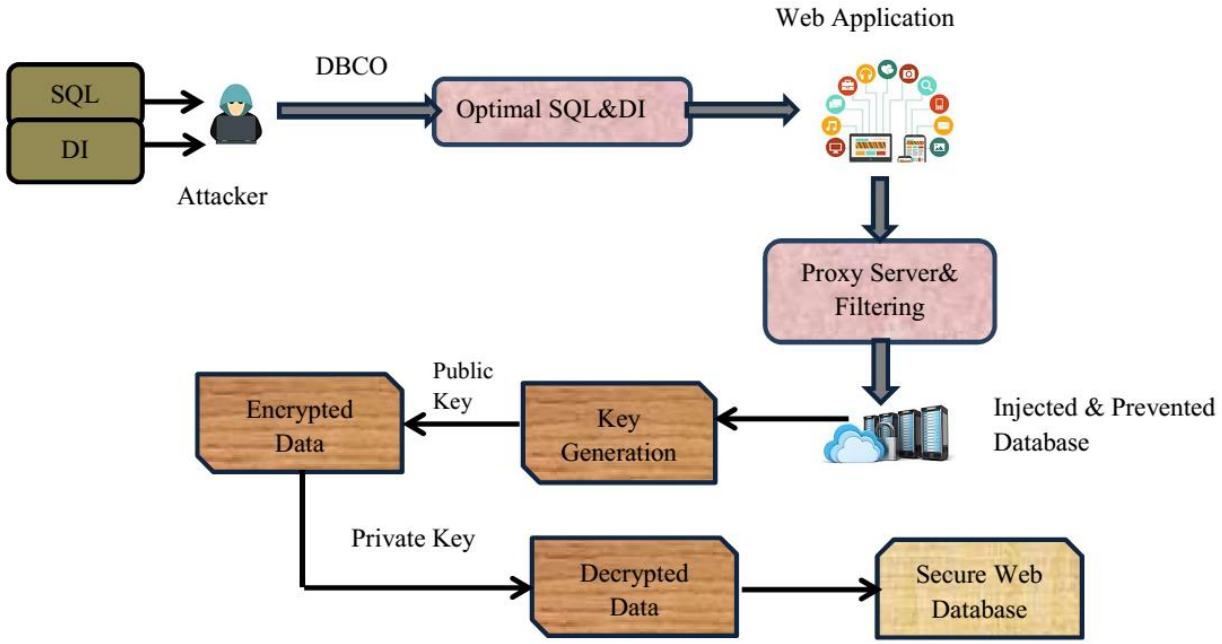

Fig 4: Graphical Model for Webpage security with Attack Injection

Key Initialization: Initialencryption key is kept secret which consists of a set of relatively prime moduli, two sets of vectors and optimal queries with the corresponding database.

Encryption: The encryption method is functional only for secret data which is referred as primary data. At this point, the ideal public key should encode every byte of ancipher query.

Decryption: This process uses the secret key and the cipher query returns the original information in a web database from this process, while the evaluation of a permitted circuit yields the correct result.

Procedure for optimal SQL and DI Injection Security

Let assume two large prime Numbers w, e randomly.

Take LCM o selected prime numbers $r = we$ and controlling parameter $\beta = (w-1, e-1)$.

Encryption: Let take a random integer value $v : v \in D_i$

$Enc(e, p_k)$

Public Encryption key $p_k = (p, i)$

Convert Plain query to Cipher Query $C = Q(i^e * v^k \bmod p_k{}^2)$

Cipher query $\Rightarrow c_p - query$

Decryption: $dec(c_p - query, S_k)$

The cipher query to decrypt where $c_i \in D^*_{k^2}$

Cipher to Plain Conversion

Compute plain query $Decr = \dfrac{Q(c_i^{\beta} \bmod p_k{}^2)}{Q(i^{\alpha} \bmod p_k{}^2)} \bmod p_k$

Once the Information is secured the security process will be ended.

The security of query focuses around the connivant attack of information proprietor and web server and by the course of action, one can expect the query point with high-security level in AQL and DI query injection processes.

Analysis

This FHEconsiders the following data sources such as the assessment key, encoded, decoded queries and the tuple of information sources that can be a mix of ciphertexts and the past assessment results. Homomorphic activities are executed on the encoded query that conveys the new encrypted information while decoding yields similar functionality.

4.7 Similarity Measure

In the wake of securing the webpage database, the cosine similarity measure is used to discover the comparability of actual data and secured data. It explores how the two vectors are identified using one another with the help of determining the cosine angle between these vectors. It is communicated by

$$Cos\,Sim = D_s \bullet D_A \Big/ |\vec{D}_s| \bullet |\vec{D}_A| \quad (3)$$

Here $\bullet$ denotes the product of two data. They capture the similarity through cosine distance between the vectors. Then, the individuals get the ranked list of simulation outcome that is sorted by affinity of a query.

5.0 RESULTS AND ANALYSIS

The proposed SQL and DI attacks' injection webpage database security model was implemented in Net beans platform and JAVA language along with cloud sim which acted as the web server here. The proposed innovative approach was executed, and its performance measures were analysed and compared with other security algorithms.

Database: Web database applications offers maximum number of facilities through the Internet. Web-based email, online purchase, forums and bulletin boards, corporate URLs, sports and news websites were database relied sites. For developing an smart web site, there is a requirement to deploy a database application. Among the decided web pages,SQL and DI are emneddedand eliminated to predict the examining task.

Fig 5: Detection Accuracy Vs. Web application

The detection accuracy of different web applications such as jobs, events, bookstore, payments, and games wereanalyzedand shown in the figure 5. For each web application, the accuracies of SQL page, DI page, detection, and prevention were examined and compared. Considering the job application, the SQL page attained 79% while DI page attained70%. The query detected was in the range of 97.23% and prevented was 96.89%. Similarly, the parameters were analyzed for other applications like events, bookstore, payments, and games. The graph concludes that the prevention rate was higher in the events of web application compared to other.

Table 1: Attack injection and detection for proposed model

Database Size(kb)	Total number of queries		Injected		Detected		Prevented	
	SQL	DI	SQL	DI	SQL	DI	SQL	DI
10	13	10	9	4	7	3	6	1
20	11	9	7	6	4	3	1	1
30	9	13	3	4	2	2	0	0
40	14	15	8	8	6	6	3	2
50	12	10	5	6	3	3	1	0

Table 1 illustrates the attack injection and detection for different database sizes of the proposed model. For 10 kb size data, the total number of the queries given to SQL was 13 and DI was 10, the injected queries were 9 for SQL and 4 for DI, the detected queries were 7 for SQL and 3 for DI and the prevented queries were 6 for SQL and 1 for DI. Likewise, the proposed model was investigated for database sizes such as 20, 30, 40 and 50 kb. The minimum detected and prevented queries were zero which was achieved in 30kb database.

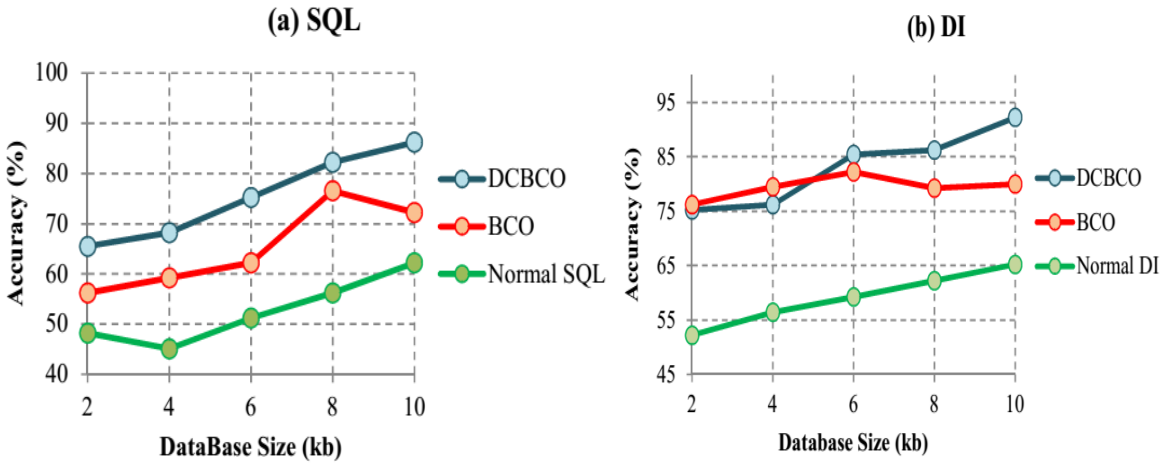

Fig 6: Optimal Query Vs. Accuracy

Figure 6 explains the accuracy of the proposed model versus optimal query. Figures 6 (a) and (b) describe SQL accuracy and DI accuracy respectively. For query 2, the accuracies of DCBCOwas 67%, BCOwas 58%, and normal SQL was 49%. When the performance measures were compared, the proposed DCBCOattained greater accuracy. For different number of queries, the accuracies of DCBCO, BCO, and normal DI were analyzed and compared.

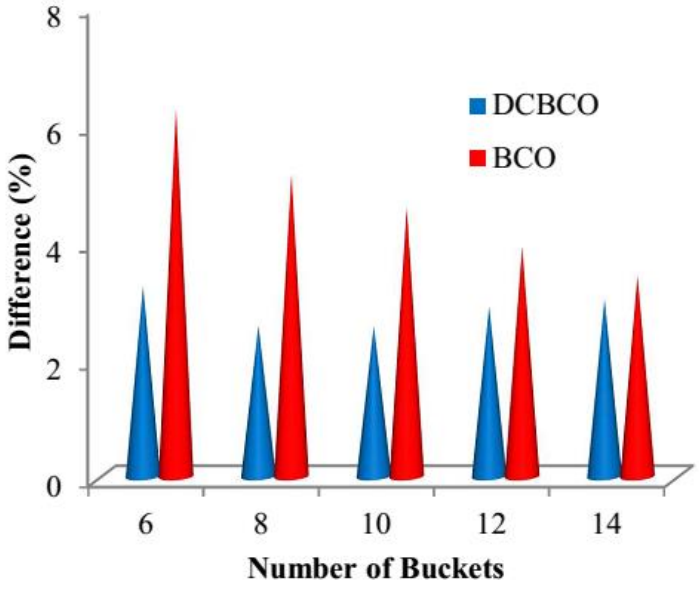

Fig 7: Fitness evaluation

The fitness evaluation, i.e. difference value was analyzedand shown in the figure 7 by varying the bucket size value. For bucket size 6, the fitness value attained for BCOwas 3.2 and DCBCOwas 6.3. For bucket size 8, the fitness value achieved for BCOwas 2.6 and DCBCOwas 5.2. For bucket size 10, the fitness value attained for BCOwas 2.6 and DCBCOwas 4.5. For bucket size 12, the fitness value achieved for BCOwas 2.8 and DCBCOwas 3.8. At last, for bucket size 14, the fitness value attained for BCOwas 3.0 and DCBCOwas 3.4. The optimal fitness value was accomplished in DCBCO technique for all the bucket sizes.

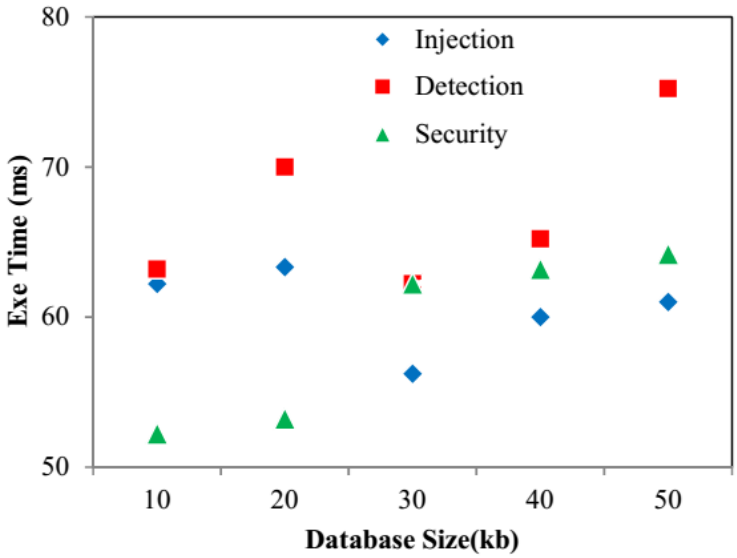

Fig 8: Execution time analysis

The execution time analyses of different database sizeswasanalyzed and compared among injection, detection, and security in the figure 8. The computational processing time was analyzed for three methods such as query injected, detected and its security. For 10 kb database, the time taken for injecting the query was in the range of 62 ms to 60 ms, the detection range was 62 ms to 78 ms and security attained 51 ms to 64 ms. Similarly, the computational time was analyzed for database of different sizessuch as 20kb, 30kb, 40kb, and 50kb.

Table 2: Web application security analysis

Database Size	Similarity		Security Level		
	Cosine	Euclidean distance	FHE	ECC	AES
10	0.22	0.59	91.66	82.22	65.12
20	0.32	0.54	89.22	73.22	69.45
30	0.41	0.62	96.22	76.45	59.45
40	0.43	0.72	92.11	79.15	56.12
50	0.35	0.77	90.1	80	54.45

Table 2 describes the security analysis of web application by examining similarity measures such as cosine and Euclidean distance. The security level was analyzed for three encryption algorithms such asFHE, ECC, and AES. The maximum similarity measure attained in the 50 kb database was 0.77, and the highest security level was achieved in the FHE algorithm 96.22 for 30 kb database.

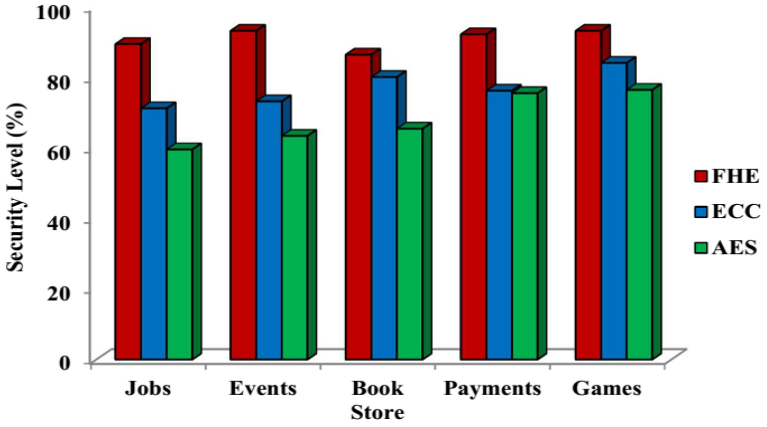

Fig 9: Security level for Web app

The security level of diverse web applicationswas investigated and related in the figure 9 with encryption algorithms such as FHE, ECC, and AES. The highest security level was attained in the events' web app and games application in comparison with other web apps. For events web app, the proposed FHE encryption algorithm achieved 94% security level, ECC achieved 74%, and AES accomplished 62%. Similarly, the security level of jobs, bookstore, payments, and games are depicted in the figure 10. The bar graph states that the projecyedFHE algorithm attained the maximum security level compared to ECC and AES algorithms.

6.0 CONCLUSION

In this paper, we have examined the optimal SQL and DI attacks in webpage security. The optimal dummy was chosen by DBCOandwascompared with BCOwhich offered better fitness results. A combination of SQL and DI injection detection mechanismshelpsin developing safe and robust model for developing the web application in a optimal fashion with the help ofderived resultssince it has been a normal when compared with the presented approach. FHEwas utilized to analyze the security level of the current work. The proposed technique analyzed all the generated optimal SQL and DI queries related to user input and caught the first structure of the query statement. As per the implementation results, it is noted that the proposed method achieved 93.56% of the security level of prevented webpage implication-based databases. The effect on the businesses must be comprehended to decrease the risks involved in SQL and DI injection assaults. In future,the query detection and prevention methodscan be used with optimization techniques.

ACKNOWLEDGEMENT

Dr. K. Shankar sincerely acknowledge thefinancial support of RUSA–Phase 2.0 grant sanctioned vide Letter No. F. 24-51/2014-U, Policy (TNMulti-Gen), Dept. of Edn. Govt. of India, Dt. 09.10.2018.

REFERENCES

[1] Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J. and Lv, W., 2019. Edge Computing Security: State of the Art and Challenges. *Proceedings of the IEEE*, *107*(8), pp.1608-1631.

[2] Kar, D., Panigrahi, S. and Sundararajan, S., 2016. SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. *Computers & Security*, *60*, pp.206-225.

[3] Bogatov, D., Kollios, G. and Reyzin, L., 2018. A Comparative Evaluation of Order-Preserving and Order-Revealing Schemes and Protocols. *IACR Cryptology ePrint Archive*, *2018*, p.953.

[4] Voitovych, O.P., Yuvkovetskyi, O.S. and Kupershtein, L.M., 2016, September. SQL injection prevention system. In *2016 International Conference Radio Electronics & Info Communications (UkrMiCo)* (pp. 1-4). IEEE.

[5] Xiao, L., Matsumoto, S., Ishikawa, T. and Sakurai, K., 2016, November. SQL Injection Attack Detection Method using Expectation Criterion. In Computing and Networking (CANDAR), 2016 Fourth International Symposium on (pp. 649-654). IEEE.

[6] Lee, I., Jeong, S., Yeo, S. and Moon, J., 2012. A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 55(1-2), pp.58-68.

[7] Ma, J., Chai, K., Xiao, Y., Lan, T. and Huang, W., 2011, September. High-Interaction Honeypot System for SQL Injection Analysis. In Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on (Vol. 3, pp. 274-277). IEEE.

[8] Bittal, V. and Banerjee, S., 2016. Prevention Guidelines of SQL Injection Database Attacks: An Experimental Analysis. In Emerging Research in Computing, Information, Communication and Applications (pp. 23-32). Springer, New Delhi.

[9] Uwagbole, S.O., Buchanan, W.J. and Fan, L., 2017, May. Applied machine learning predictive analytics to SQL injection attack detection and prevention. In Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on (pp. 1087-1090). IEEE.

[10] Balasundaram, I. and Ramaraj, E., 2012. An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching. Procedia Engineering, 30, pp.183-190.

[11] Rani, D.R., Kumar, B.S., Rao, L.T.R., Jagadish, V.S. and Pradeep, M., 2012. Web security by preventing sql injection using encryption in stored procedures.

[12] Singh, Y. and Kaur, P., 2015. Extended Security Techniques on Web Applications. International Journal of Computer Science and Mobile Computing, 4(6), pp.741-749.

[13] Hamdi, M., Safran, M. and Hou, W.C., 2014, March. A Security Novel for a Networked Database. In Computational Science and Computational Intelligence (CSCI), 2014 International Conference on (Vol. 1, pp. 279-284). IEEE.

[14] Sadeghian, A., Zamani, M. and Abdullah, S.M., 2013, September. A taxonomy of SQL injection attacks. In Informatics and Creative Multimedia (ICICM), 2013 International Conference on (pp. 269-273). IEEE.

[15] Ali, A.B.M., Abdullah, M.S. and Alostad, J., 2011. SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. Procedia Computer Science, 3, pp.453-458.

[16] Ping, C., 2017, December. A second-order SQL injection detection method. In 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (pp. 1792-1796). IEEE.

[17] Jang, Y.S. and Choi, J.Y., 2014. Detecting SQL injection attacks using query result size. Computers & Security, 44, pp.104-118.

[18] Kulkarni, S. and Urolagin, S., 2012. Review of attacks on databases and database security techniques. International Journal of Emerging Technology and Advanced Engineering, 2(11), pp.253-263.

[19] Kar, D., Panigrahi, S. and Sundararajan, S., 2016. SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. Computers & Security, 60, pp.206-225.

[20] McWhirter, P.R., Kifayat, K., Shi, Q. and Askwith, B., 2018. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. Journal of information security and applications, 40, pp.199-216.

[21] Dalai, A.K. and Jena, S.K., 2017. Neutralizing SQL Injection Attack Using Server Side Code Modification in Web Applications. Security and Communication Networks, 2017.

[22] Sonewar, P.A. and Thosar, S.D., 2016, August. Detection of SQL injection and XSS attacks in three tier web applications. In Computing Communication Control and automation (ICCUBEA), 2016 International Conference on (pp. 1-4). IEEE.

[23] Katole, R.A., Sherekar, S.S. and Thakare, V.M., 2018, January. Detection of SQL injection attacks by removing the parameter values of SQL query. In 2018 2nd International Conference on Inventive Systems and Control (ICISC) (pp. 736-741). IEEE.

[24] Dinu, P.S., Kumar, D.S. and Rahman, M.A., 2015. Preventing SQL injection attacks using cryptography methods. International Journal of Scientific Research Engineering and Technology (IJSRET), 4(5), pp.582-5.

[25] Rani, D.R., Kumar, B.S., Rao, L.T.R., Jagadish, V.S. and Pradeep, M., 2012. Web security by preventing sql injection using encryption in stored procedures.

[26] Basta, C., Elfatatry, A. and Darwish, S., 2016. Detection of SQL Injection Using a Genetic Fuzzy Classifier System. International Journal of Advanced Computer Science and Applications, 7(6), pp.129-137.

[27] Atefeh Tajpour, Suhaimi Ibrahim and Maslin Masrom, SQL Injection Detection and Prevention Techniques, Journal of Advancements in Computing Technology Volume 3, Number 7, August 2011.

[28] Davidović, T., 2016. Bee colony optimization Part I: The algorithm overview. Yugoslav Journal of Operations Research, 25(1).

[29] Alwan, Z.S. and Younis, M.F., 2017. Detection and Prevention of SQL Injection Attack: A Survey. International Journal of Computer Science and Mobile Computing, 6(8), pp.5-17.

[30] Shankar, K. and Lakshmanaprabu, S.K., 2018. Optimal key based homomorphic encryption for color image security aid of ant lion optimization algorithm. International Journal of Engineering & Technology, 7(1.9), pp.22-27.

[31] Sachi Nandan Mohanty, K. C. Ramya, S. Sheeba Rani, Deepak Gupta, K. Shankar, S. K. Lakshmanaprabu, Ashish Khanna, "An efficient Lightweight integrated Blockchain (ELIB) model for IoT security and privacy", Future Generation Computer Systems, Volume 102, Page(s): 1027-1037, January 2020.

[32] K. Shankar, SK. Lakshmanaprabu, Deepak Gupta, Ashish Khanna, Victor Hugo C. de Albuquerque, "Adaptive Optimal Multi Key Based Encryption for Digital Image Security", Concurrency and Computation: Practice and Experience, December 2018. DOI: https://doi.org/10.1002/cpe.5122

[33] Mohamed Elhoseny and K. Shankar, "Reliable Data Transmission Model for Mobile Ad Hoc Network Using Signcryption Technique", IEEE Transactions on Reliability, Page(s): 1-10, June 2019. In Press. DOI: https://doi.org/10.1109/TR.2019.2915800

[34] K. Shankar, Mohamed Elhoseny, R. Satheesh Kumar, S. K. Lakshmanaprabu, Xiaohui Yuan, "Secret image sharing scheme with encrypted shadow images using optimal homomorphic encryption technique", Journal of Ambient Intelligence and Humanized Computing, December 2018. https://doi.org/10.1007/s12652-018-1161-0

[35] Mohamed Elhoseny, K. Shankar, S. K. Lakshmanaprabu, AndinoMaseleno, N. Arunkumar, "Hybrid optimization with cryptography encryption for medical image security in Internet of Things", Neural Computing and Applications - Springer, October 2018. https://doi.org/10.1007/s00521-018-3801-x

[36] K. Shankar, Mohamed Elhoseny, E. Dhiravidachelvi, SK. Lakshmanaprabu, Wanqing Wu, "An Efficient Optimal Key Based Chaos Function for Medical Image Security", IEEE Access, Vol.6, Issue.1, page(s): 77145-77154, December 2018.https://doi.org/10.1109/ACCESS.2018.2874026

[37] T. Avudaiappan, R. Balasubramanian, S. SundaraPandiyan, M. Saravanan, S. K. Lakshmanaprabu, K. Shankar, "Medical Image Security Using Dual Encryption with Oppositional Based Optimization Algorithm", Journal of Medical Systems, Volume 42, Issue 11, pp.1-11, November 2018.https://doi.org/10.1007/s10916-018-1053-z