ISSN 2278-3091

Volume 9 No.2, March - April 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse137922020.pdf

https://doi.org/10.30534/ijatcse/2020/137922020

Efficient Cloud Authentication Scheme using Single Sign-On Nature in Hands with Branca Strategy

Remya Chandran[1], Dr.A.Sasi Kumar[2]

[1]Ph.D Research Scholar, Department of Information Technology, School of Computing Sciences, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, Tamil Nadu, India.

[2]Professor, Department of Information Technology, School of Computing Sciences, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, Tamil Nadu, India.

[1]nivedika@gmail.com, [2]askmca@yahoo.com

ABSTRACT

The technological world is surround with Internet oriented services and its architecture is purely based on MultiServer Platform oriented as well as the security concerns are more important to deal with MultiServer platforms as like many web concerns to deal with (for example: Google, FaceBook and so on). All these concerns follow a strategy called Single-Sign On (SSO), in which it provides efficient solutions to deal the authentication issues based on token or key based scenarios. In this SSO based system, users have to sign into system by using authenticated username and password with one time and the remaining processes will be handled by using generated tokens. In the past system, there are several methodologies to associate with this Single-Sign On norm, but all are falling under certain problems and security issues. These all issues found because of using traditional security mechanisms such as key-hashing, data-hashing and so on. So, that a new technique is required to resolve these issues and provide an intelligent token generation scheme. In this paper, an efficient cloud oriented authentication security system is followed, which is based on Single-Sign On logic in association with latest and powerful token generation mechanism called Branca. Branca usually generates a token based on details provided by user and produces a secret key with current date/time as well as the key is tokenized by using encryption mechanism, which is called Authenticated Encrypted Token (AET). An Authenticated Encrypted Token creates a path to secure the message, which cannot be visible to the intruders/sniffers to alter or acquire it. The name Branca is formed to attract, which is for "IETF-XChaCha20-Poly1305-AEAD" type messages association-with additional version number and time-stamp. The name Branca is a best option for secure authentication mechanism by means of its authenticated-and-encrypted Application Programming Interface tokens. Branca features does not specify any additional payload-formats and compare to other options, with the help of this Branca technique user can have modern encryption schemes and smaller token size to process their data.

Key words: Single-Sign On, SSO, Branca, Authenticated Encrypted Token, AET

1.INTRODUCTION

In today's internet world, each and every individual belong multi-server environment to deal with the communication and data preservation needs, which is handled by many internet services and commercial internet based organizations such as Google, Facebook, Twitter and so on. All are applying some promising strategies to provide efficient authentication norms to their clients and users, which is called as Single Sign-On strategy. In industry there are several authentication schemes available to clients with the ability sign on using one set of username and password alleviating the need of multiple identities and multiple passwords. Although promising, SSO mechanisms need to be extra robust and provide utmost authentication for their users. Authentication Key based schemes are the most popular and well-known robust scheme to provide efficient security mechanism over cloud environments to make feel the clients and users on secure level. Some of the unidirectional authentication security keys provide strong key nature to prevent the user and data in safer manner. Due to the unidirectional nature of the authentication channel between the service provider and the client in SSO and the lack of a recent authentication key, researchers have pointed out vulnerabilities in such schemes leading to attacks such as impersonation attacks. In this paper, we proposed an efficient solution that effectively handles the Single Sign-On (SSO) scheme, which is called, Branca Strategy. Branca is a secure and easy way to use (key) token format which makes it hard to shoot or guess by hackers or intruders in the cloud environment. It uses IETF XChaCha20-Poly1305 AEAD symmetric encryption to create encrypted and tamperproof tokens/keys.

Payload itself is an arbitrary sequence of bytes and it can use for example a JSON object, plain text string or even binary data serialized by Message Pack or Protocol Buffers. The main objective of the proposed system (Branca) is enhancing the security nature of the cloud environment, to provide the easy solution of authentication problems as well as related issues and finally the goal is to provide all these features with small token size. The token format of Branca is usually contains a header, cipher text and an authentication tag and in which the header consists of version, timestamp and nonce and putting them all together we get following structure.

Version (1B) || Timestamp (4B) || Nonce (24B) || Cipher text (*B) || Tag (16B)

String representation of the above binary token must use base62 encoding with the following character set.

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabc defghijklmnopqrstuvwxy

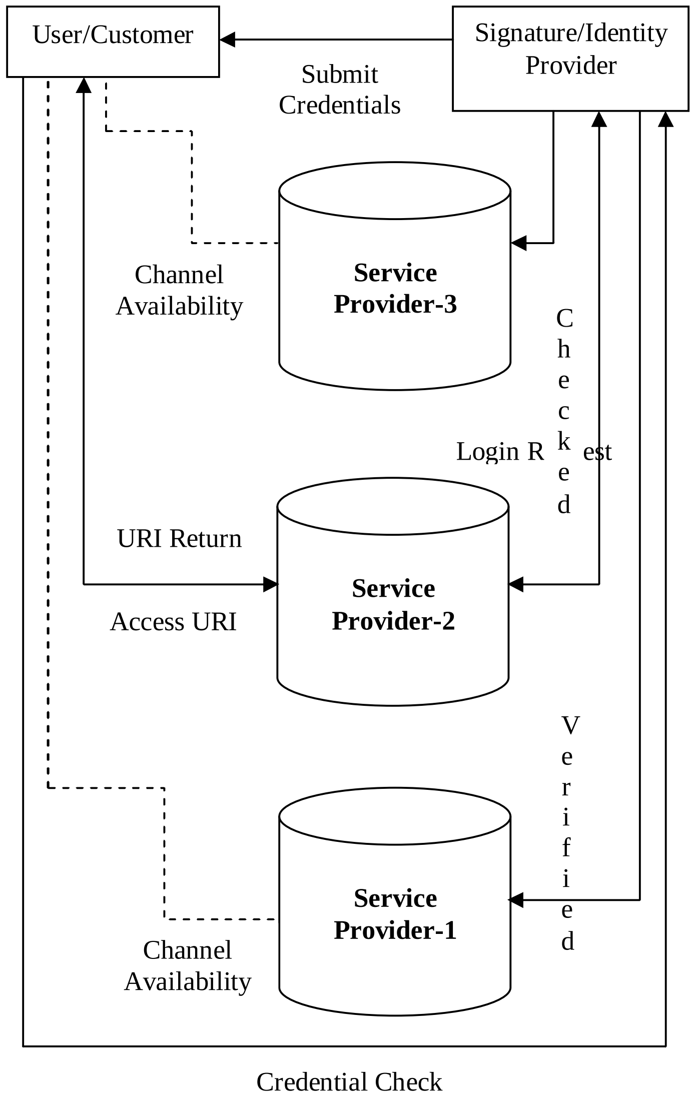


Figure 1: Architectural View of Single-Sign On

2. SYSTEM ANALYSIS

A. Existing System Summary

Because of the needs of communication environments and social Medias, people require to authenticate into multiple-servers with proper credentials. An improvement of Cloud-Computing-Nature and Internet Enabled Services such as IoT gives boom to the Single-Sign On Methodology. This methodology acts like a token based approach, which generates token for processing and allow users to login into system with username and password but in background further processes are handled by tokens [1][4][5]. However in this methodology the tokenization process is carried out over network medium, so there are lots of chances to get the token be leaked [4]. Generally Single-Sign On scheme operates based on three-difference key parts such as: (i) Service Access Time - user permission is essential, (ii) Service Provider and (iii) Customers/Client - in front end [1][2][3]. In past systems [2][3][6] authors found that the logical mistakes occurred over the implementations of Single-Sign On methodology to work with traditional security mechanisms such crypto keys and so on. Anyway, most of the traditional approaches follow Merkle Has Tree based approach for processing the crypto tokens [2]. In that [2] authors found that the user felt attacked in two cases such as: Identity Hijacking Attack and Data Hijacking Attack. These two different attacks causes major damage in past systems, in that Identity Hijacking Attack purely focuses on user's login credentials, which it attains the input from the user and pass those authentication input to the hijacker present into the network or outside to the network. The second category called Data Hijacking Attack, in which the attacker/hijacker trying to get the data from the user/client end. But most of the time this attack is happened only for corrupting the actual data, because usually all the data passed via server is in the form of crypto-text. However, the attack possibilities are high enough in the past systems. So, that a new token generation principles are required to provide high level of security mechanisms to the customers to make them feel free and highly efficient in working with that.

B. Proposed System Summary

In the proposed system, our fundamental objective is to build a powerful resolution that proficiently handles the security of Single-Sign On methodology. In this paper, we give uncommon consideration to the potential reasons for certain Single-Sign On vulnerabilities. From a crypto-service perspective, access to the Single-Sign On requires the Service-Handlers, Customers just as Identity-Manager to convey their 'communications to open and private keys as found in most of the plans talked about in the past implementations [6][7][8][9]. The security of the entirety condition is dependent upon these keys. On the off chance that in any capacity whatsoever these keys are undermined, there is the problem of taken one's identity, secret key renouncement and their preferences. In this proposed system, a Keyless-Signature Strategy is introduced, that depends on using Branca logic and Authenticated

Encrypted Token (AET) principles. In Table 1, the documentations utilized in the paper are presented.

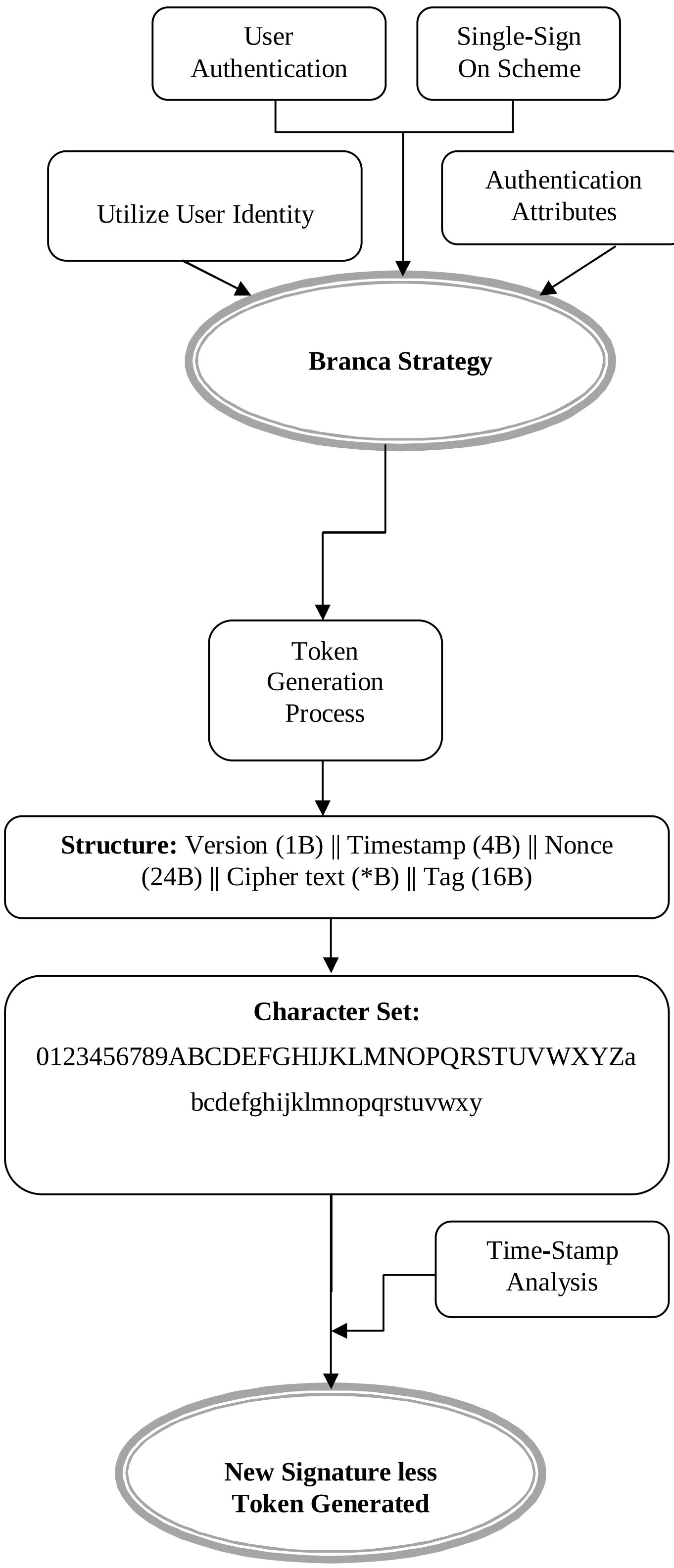

User Authentication
Single-Sign On Scheme
Utilize User Identity
Authentication Attributes
Branca Strategy
Token Generation Process
Structure: Version (1B) || Timestamp (4B) || Nonce (24B) || Cipher text (*B) || Tag (16B)
Character Set:
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZa bcdefghijklmnopqrstuvwxy
Time-Stamp Analysis
New Signature less Token Generated

Figure 2: Proposed System Architectural View

Table 1: Single-Sign On Parametric-Symbols

Parametric-Symbols	Summary
IDP	Middle Identity-Provider
Cl	Client/Customer
SP	Service Provider
SH	Service Handlers
Seed'	Secret-Tokens Generated by Service-Providers
LO	Client Logout Activity
A	Authenticated Token
Co	Computational Cost

3. SYSTEM IMPLEMENTATION

A. Attack Model

The proposed system model considers the following attack models while processing the authentication request with Branca logics in the accompanying advances. The following are the different types of attack possibilities over communications between client and server scenario.

(i) Attacker/Hijacker tries to attain the contact details of the particular user by means of sending a request URI to respective user.
(ii) Attacker/Hijacker tries to act like a normal host and monitor the networks by means of sniffing nature.
(iii) Attacker/Hijacker tries to hijack the messages via session values stored into the respective client machine.
(iv) Attacker/Hijacker sends cross-linked messages to client end to confuse the input and attain the authentication-credentials.
(v) Attacker/Hijacker starts sniffing the data in network caches and rescues that for their destructive purposes.

B. Single-Sign On Tokenless Key Generation Procedure

The Tokenless Key Generation Procedure of Single-Sign On methodology associated with the proposed logic is illustrated in detail over this section. Before that the key modules of Single-Sign On logic is analyzed as follows:

(i) User-Details Provider: The User-Details Provider module handles confirmation and gives validation declarations to the Service-Providers and Customers. The User-Details Provider keeps track of a Crypto-Schedule, which is pre-determined and refreshed intermittently over particular period of time (in the accompanying advance we will appear at the point when the Crypto-Schedule is refreshed), not-with-standing enlisting each sign in by the customer in a Crypto-Tree.

(ii) Customer: The primary transactions between the customer and the User-Details Provider include the creation and passing of verification details ("username" and "password"). This progression can be SSL/TSL made sure about and the primary sign in and each consequent sign in are utilized as contribution for the crypto-tree. When the foundation of the crypto-tree is determined, the base are connected and encrypted to create the new key for the proposed strategy.

(iii) Service Provider: Like the customer, the Service-Provider also communicates with the User-Details Provider to check and get verification attestations concerning a specific customer. Traffic between the Service-Provider and Customer is made sure about by utilizing an uni-directional crypto-tie, which is intended to deliver a bi-directional verification channel.

The User-Details Provider is liable for refreshing the Crypto-Keys on particular time period based on different login schedules. Each time the crypto-schedule is refreshed, the root esteem is given over to the Service-Providers also, the customer as the new key for verification and the key demonstrations as a benchmark to approve the correspondence between entities. Thus, rather than sharing an open key among customers and servers as-well-as using private keys, the protection of which is basic to the system, the key is utilized for confirmation of realness of the conveying entities. In that capacity, if a attacked Service-Provider or Customer attempts to get a grip of a customer's authentication details, they would need to have a refreshed and new form of the key.

```
---------------------------------------------------------------------------
Algorithm: Keyless Token-Generation
---------------------------------------------------------------------------
Input: User Details
Output: Encoded Token
Step-1: Gather the passing query from the client end.
Step-2: Apply the collected Result over JSON query logics.
Step-3: Check if the applied data length is greater than 0 or less than 0.
Step-4: If it is greater than 0, then generate the Token Data.
Step-5: Generate the Token-ID by gather the user-id from the input query.
Step-6: Generate the Token-Name by gather the user's first name from the input query.
Step-7: Generate the Token-Mail by gather the user's mail-id from the input query.
Step-8: Generate the Token-Address by gather the user's address from the input query.
Step-9: Generate the Time Stamp with the formulation of
Floor = Date /1S.
Step-10: Token Generated with the base of Step-5 to 9.
Step-11: Assign the Generated Token to JSON for further verifications.

---------------------------------------------------------------------------
```

The following details will clearly explains the Pseudocode used to generate the token in real-time manner with the help of above mentioned algorithm logics (Keyless Token-Generation).

```
---------------------------------------------------------------------------
Pseudocode: Keyless Token-Generation
---------------------------------------------------------------------------
String result="";
result="SELECT * FROM users WHERE email='" + req.body.email + "'";
resultOut = JSON.stringify(result);
resultOut = JSON.stringify(result);
data = JSON.parse(resultOut);
console.log(data.insertId);
if (data.insertId.length != 0) {
          tokenData = {
            user_id: data.insertId,
            name: req.body.first_name,
            email: req.body.email,
            address: req.body.address,
            timestamb: Math.floor(new Date() / 1000)
          }
data = JSON.stringify(tokenData);
const token = branca.encode(data);
console.log(token);
res.header('auth_token', token).send({ code: 200, token: token });
---------------------------------------------------------------------------
```

C. BRANCA: Single-Sign On Token Based Authentication Procedure

In this section of the paper, we introduce a Single-Sign On Token based Authentication Procedure in detail. The following algorithm clearly explains the concept of SSO Token based Authentication Norms.

```
---------------------------------------------------------------------------
Algorithm: Branca - SSO Token-Based Authentication
---------------------------------------------------------------------------
Input: User Authentication Credentials
Output: Boolean Result
Step-1: Create a Constant type Variable Ex. key.
Step-2: Take an object for Branca.
Step-3: Collect the Authentication token from the Client end.
Step-4: Decode the collected token by using Branca object.
Step-5: Ex. Branca.Decode(Collected_Object);
Step-6: Verify the decoded token and store the result into user defined variable.
Step-7: Check the decoded token timestamp with present time.
Step-8: If the timestamp matches with the token timestamp, then provide the positive result to user.
Step-9: Otherwise return the token expiry alert to the respective user.
Step-10: Token based SSO Authentication process Completed.

---------------------------------------------------------------------------
```

The following details will clearly explain the Pseudocode used to authenticate the user in real-time manner with the help of above mentioned algorithm logics (Single-Sign On Token based Authentication).

Pseudocode: Branca - SSO Token-Based Authentication

```
const key = "supersecretkeyyoushouldnotcommit";
const branca = require("branca")(key);
module.exports = function auth(req, res, next) {
let token = req.cookies['auth_token']
console.log(token)
 if (!token) return res.status(401).send({ message: 'You Must Login First' });
try {
      const verified = branca.decode(token);
      console.log("token section")
      console.log(verified.toString())
      let data = verified.toString();
      decodedToken = JSON.parse(data)
      console.log(decodedToken.timestamb)
      if (decodedToken.timestamb * 1000 < Date.now()) {
         req.user = data;
         next();
      } else {
         return res.status(400).send("token expired")
      }
   } catch (err) {
      res.status(400).send({ message: "invalid token" })
   }
}
```

D. Encryption and Decryption Strategies of Branca SSO Tokens

In this section, the process of encryption and decryption will be clearly discussed and the proposed encryption strategy is AES-JS, which is a standard and powerful implementation-of-Javascript Advanced Encryption Standard cipher-conversion algorithm and vice versa as well for de-cipher. This cipher conversion and de-conversion algorithm supports multiple key types such as 128, 256 and so on. In the proposed system of SSO based Branca implementation, this AES-JS algorithm is associated to provide more security in the results. The following algorithm definitions clearly describe the nature and flow of the algorithm.

Algorithm: AES-JA Encryption

```
Input: Plain Text
Output: Cipher-Text
Step-1: Create a Constant type Function, Ex. encryptData.
Step-2: Pass the argument Data to process.
Ex. Argument = Data.
Step-3: Initialize the variable for key definitions.
Step-4: Assign the 16 Key values as an input to the variable mentioned in Step-3.
Step-5: Declare a byte variable to collect the plain data by using the function called “aesjs.utils.utf8.toBytes()”.
Step-6: Assign the Key specification to the AES-JS function.
Step-7: Encrypt the converted data bytes by using “aesCtr.encrypt()” function.
Step-8: Assign the Encrypted data to the byte variable for precedence.
Step-9: Step-8 is achieved by using fromBytes function, such as “aesjs.utils.hex.fromBytes()”.
Step-10: Encrypted Data Generated. Ex. encryptedHexData.
```

The following details will clearly explain the Pseudocode used to authenticate the user in real-time manner with the help of above mentioned algorithm logics (AES-JS Encryption).

Pseudocode: AES-JA Encryption

```
const encryptData = (data) => {
      let key_128 = [67, 1, 2, 34, 4, 5, 6, 7, 8, 9, 10, 22, 12, 13, 14, 87];
      let dataBytes = aesjs.utils.utf8.toBytes(data);
      let aesCtr = new aesjs.ModeOfOperation.ctr(key_128, new aesjs.Counter());
      let encryptedDataBytes = aesCtr.encrypt(dataBytes);
      let encryptedHexData
      = aesjs.utils.hex.fromBytes(encryptedDataBytes);
      return encryptedHexData;
}
```

The following algorithm definitions clearly describe the nature and flow of the AES-JS decryption algorithm nature.

Algorithm: AES-JA Decryption

```
Input: Cipher Text
Output: Plain-Text
Step-1: Create a Constant type Function, Ex. decryptData.
Step-2: Pass the argument Encrypted_Data to process. Ex. Argument = encryptedHexData.
Step-3: Initialize the variable for key definitions.
Step-4: Assign the 16 Key values as an input to the variable mentioned in Step-3.
Step-5: Declare a byte variable to collect the encrypted data bytes by using the function called “aesjs.utils.hex.toBytes”.
Step-6: Assign the Key specification to the AES-JS function.
Step-7: Decrypt the Encrypted_Data bytes by using “aesCtr.decrypt()” function.
Step-8: Assign the decrypted data to the byte variable for precedence.
Step-9: Step-8 is achieved by using fromBytes function, such as “aesjs.utils.utf8.fromBytes()”.
Step-10: Decrypted Data Retained.
```

The following details will clearly explain the Pseudocode used to authenticate the user in real-time manner with the help of above mentioned algorithm logics (AES-JS Decryption).

Pseudocode: AES-JA Decryption

```
const decryptData = (encryptedHexData) => {
    let key_128 = [67, 1, 2, 34, 4, 5, 6, 7, 8, 9, 10, 22, 12, 13,
    14, 87];
    let encryptedBytes =
    aesjs.utils.hex.toBytes(encryptedHexData);
    let aesCtr = new aesjs.ModeOfOperation.ctr(key_128,
    new aesjs.Counter());
    let decryptedBytes = aesCtr.decrypt(encryptedBytes);
    let decryptedText =
    aesjs.utils.utf8.fromBytes(decryptedBytes);
    console.log(decryptedText);
    return decryptedText;
}
```

4.RELATED WORKS

In the year of 2016, the authors "M. Ge, K.-K. R. Choo, H. Wu, and Y. Yu [10]", proposed a paper titled "Survey on key revocation mechanisms in wireless sensor networks [10]", in that they described such as: in modern network based or internet enabled service scenarios, attacks and its resemblance is really high, so that the protection mechanisms need to be improved accordingly. Data Confidence is considered as a core in this paper [10] as well as the authors illustrate the status of key-revocation principles over decentralized-hybrid environments. The concept of rekeying in wireless sensor-networks are also analyzed over the paper [10].

In the year of 2015, the authors "D. He and D. Wang [11]", proposed a paper titled "Robust biometrics-based authentication scheme for multiserver environment [11]", in that they described such as: the importance of authentication schemes with crypto-principles. In this paper, the concept is achieved based on two-different transaction precedences over open-network medium, such as: (i) Authentication using Password based approach and (ii) Authentication using RFID-SmartCard principles. The major issues regarding the proposed approach created two problems, which is ellaborated by means of the following two questions, such as: (a) Did you forget the Password ? and (b) Did you miss your Smart-Card?. So, that the proposed approach need to be revised with some advanced principles based on MultiServer concepts with the help of elliptic-curve-cryptography scheme [11]. And the proposed scheme demonstrate the logics of "Burrows Abadi Needham" as well in results [11].

In the year of 2015, the authors "V. Odelu, A. K. Das, and A. Goswami [12]", proposed a paper titled "`A secure biometrics-based multiserver authentication protocol using smart cards [12]", in that they described such as: HeWang scheme and its associated principles are robust against vulnerabilities with temporary session-memory based attack and impersonation-attack. But the paper itself clearly state these principles not stated the user-anonimities [12]. Moreover, HeWang strategy can't give the client disavowal when the SmartCard is lost/stolen or client's verification parameter is uncovered. Aside from these, HeWang strategy has some structure imperfections, for example, wrong secret password based login and its outcomes, and wrong password update during updation stage. We at that point propose another safe multiserver confirmation convention utilizing bio-metric based SmartCard and "Elliptic Curve-Cryptography" with greater security functionalities. Utilizing the Burrows Abadi Needham rationale, to show that the proposed schema gives security confirmation [12].

Table 2: Comparative Study

Merkle Hash Tree with SSO Provision	**Proposed Branca Technique Provision**
Lack of Data Concentration, due to that data lost will happen suddenly when the data overflow occurred [16].	All the data processing is handled based on temporary authentication mode only, so that the data lost cannot happen. Even if it happens, the presently working data may lose instead of all.
Host Address Missing, due to such missing of address leads data un controllability over social media streams [14][16].	Each and every time of authentication Branca approach creates a new identity to user and periodically updates the address automatically. So the host identification is not at all a problem to proposed approach of Branca.
Cryptographic hash functions are complicated while making the data to hash and while de-hashing it will be simple. It leads the data to be corrupted a any time without the known causing [15][16].	Uses advanced cryptographic and randomly changing key nature for encryption and decryption functions, so proposed methodology follows efficient crypto logics as well.
Storing huge data into the remote servers with week security measures as mentioned in the past research summary in this paper leads to make the data more complicate and privacy is a questionable one [10[11][12].	Data storage security is really high and which will be briefly elaborated in proposed system research summary over this paper in previous section.
Past hashing systems associated with merkle tree methodology in SSO using external authentication privacy norms such as Biometric appliance and single keyword data protective mechanisms such as OTP and so on, which leads the data and authentication dependency over server [9][10][11].	Branca approach follows completely independent data and security logic, which uses separate authentication norms. If biometric need to be used means that will be an additional feature of Branca, not a dependable feature.

5. RESULTS AND DISCUSSION

The proposed empirical results are implemented using the genuine digital medical image processing tool called MATLAB, which is used to design the proposed system with DEFS algorithm and all the specifications of DEFS and the Subspace ensemble learning classification are clearly demonstrated in above sections and the outcomes of-the implemented-system is clearly mentioned as follows. The following figure Figure.3 shows that the regression ranges for Margin-Based Feature-Selection process.

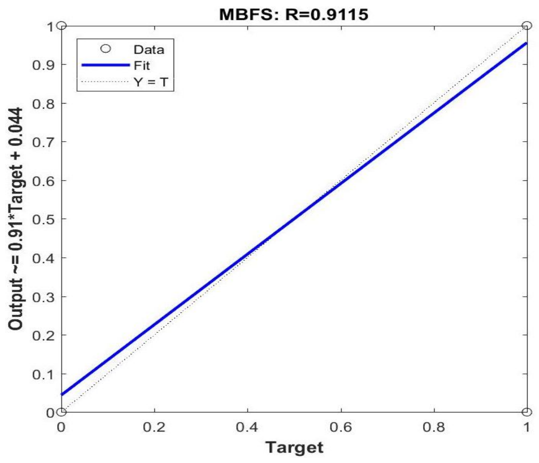
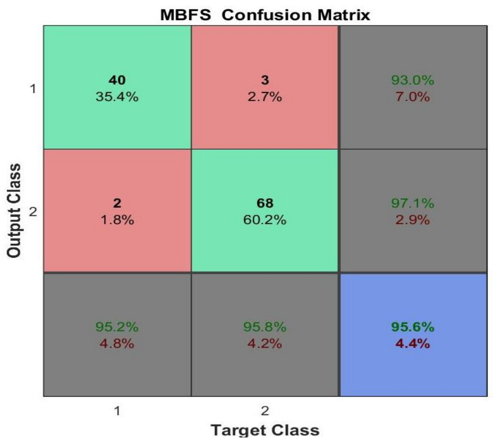


Figure 3: Regression for MBFS

The following figure Figure.4 illustrates the Confusion Matrix scenario of the Margin-Based Feature-Selection process.


MBFS Confusion Matrix
Output Class
Target Class

	1	2	
1	40 35.4%	3 2.7%	93.0% 7.0%
2	2 1.8%	68 60.2%	97.1% 2.9%
	95.2% 4.8%	95.8% 4.2%	95.6% 4.4%

Figure 4: Confusion Matrix for MBFS

The following figure Figure.5 illustrates the Performance measurement of ten different parameters for Margin-Based Feature-Selection process.

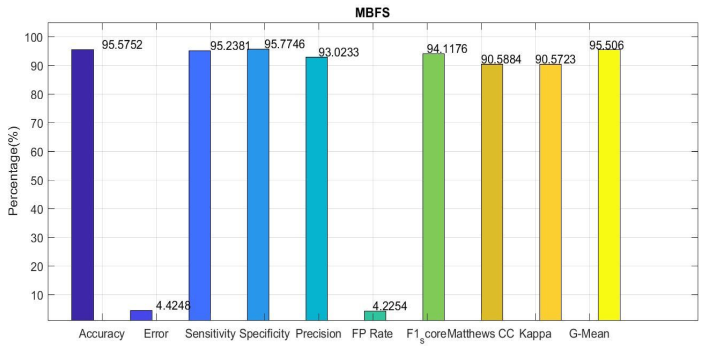


Figure 5: Performance measurement of 10 parameters for MBFS

The following figure Figjure.6 illustrates that the regression ranges for Differential-Evolution-Feature-Selection process.

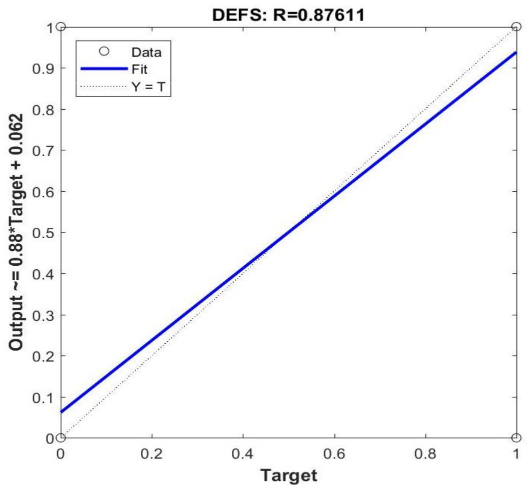
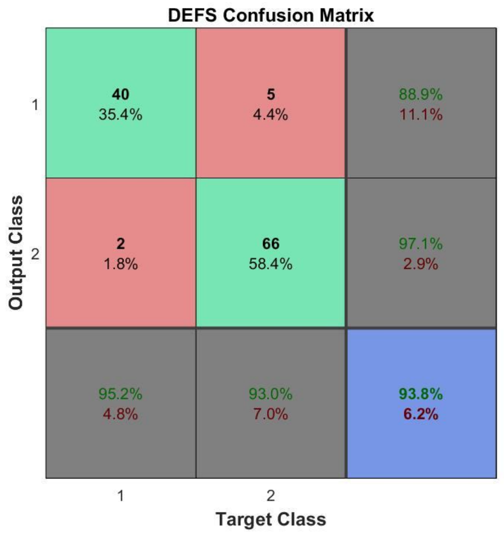


Figure 6: Regression for DEFS

The following figure Figure.7 illustrates the Confusion Matrix scenario of the Differential-Evolution-Feature-Selection process.


DEFS Confusion Matrix
Output Class
Target Class

	1	2	
1	40 35.4%	5 4.4%	88.9% 11.1%
2	2 1.8%	66 58.4%	97.1% 2.9%
	95.2% 4.8%	93.0% 7.0%	93.8% 6.2%

Figure 7: Confusion Matrix for DEFS

The following figure Figure.8 illustrates the Performance measurement of ten different parameters for Differential-Evolution-Feature-Selection process.

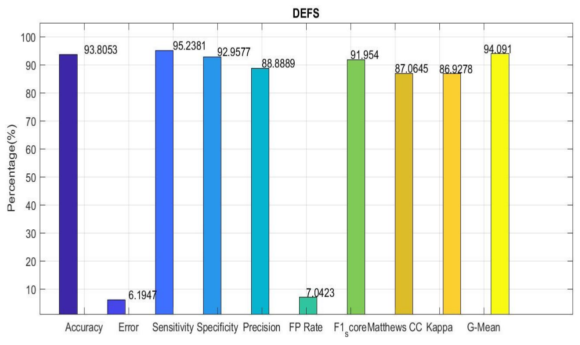


Figure 8: Performance measurement of 10 parameters for DEFS

6. CONCLUSION AND FUTURE SCOPE

In this paper, an intelligent Cloud Authentication Scheme using Single Sign-On Nature based Branca Strategy is implemented, which is more suitable to the MultiServer Authentication-strategies well. Many past server side schemes are available for Single-Sign On scheme in literature', but all are suffered under certain lackings such as uni-directional authentication norms, insufficient security parameters, poor performance and so on. In the proposed SSO based Branca application provides an efficient way to resolve all the above mentioned problems clearly under sevral cases such as speed, security and reliability. The proposed system follows the encryption strategy of Authenticated Encrypted Token (AET), which is operating under the principles of powerful JavaScript based Advanced Encryption Standard (AES-JS). The proposed system provides bi-directional authentication norms and high-security enabled credential data maintenance port, which illustrates the robustness of the proposed Branca based Single-Sign On strategy.

REFERENCES

[1] 2012, CSID; CSID Conducts Study of Consumer Password Habits, Finds Disconnect in Practices and Mindset. (Journal, Electronic), 1149.
[2] R. Wang, S. Chen, and X. Wang, "Signing me onto your accounts through facebook and google: A traffic-guided security study of commercially deployed single-sign-on web services," in Security and Privacy (SP), 2012 IEEE Symposium on, 2012, pp. 365-379.
https://doi.org/10.1109/SP.2012.30
[3] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, G. Pellegrino, and A. Sorniotti, "An authentication flaw in browser-based Single Sign-On protocols: Impact and remediations," Computers & Security, vol. 33, pp. 41-58, 2013.
https://doi.org/10.1016/j.cose.2012.08.007
[4] Y. Cao, Y. Shoshitaishvili, K. Borgolte, C. Kruegel, G. Vigna, and Y. Chen, "Protecting Web-based Single Sign-on Protocols against Relying Party Impersonation Attacks through a Dedicated Bidirectional Authenticated Secure Channel," in Research in Attacks, Intrusions and Defenses, ed: Springer, 2014, pp. 276-298.
[5] L. Lamport, "Password authentication with insecure communication," Communications of the ACM, vol. 24, pp. 770-772, 1981.
https://doi.org/10.1145/358790.358797
[6] W.-B. Lee and C.-C. Chang, "User identification and key distribution maintaining anonymity for distributed computer networks," Comput Syst Sci Eng, vol. 15, pp. 211-214, 2000.
[7] T.-S. Wu and C.-L. Hsu, "Efficient user identification scheme with key distribution preserving anonymity for distributed computer networks," Computers & Security, vol. 23, pp. 120-125, 2004.
https://doi.org/10.1016/j.cose.2003.09.005
[8] Y. Yang, S. Wang, F. Bao, J. Wang, and R. H. Deng, "New efficient user identification and key distribution scheme providing enhanced security," Computers & Security, vol. 23, pp. 697-704, 2004.
[9] Y.-P. Liao and S.-S. Wang, "A secure dynamic ID based remote user authentication scheme for multi-server environment," Computer Standards & Interfaces, vol. 31, pp. 24-29, 2009.
https://doi.org/10.1016/j.csi.2007.10.007
[10] M. Ge, K.-K. R. Choo, H. Wu, and Y. Yu, ``Survey on key revocation mechanisms in wireless sensor networks," J. Netw. Comput. Appl., vol. 63, pp. 2438, Mar. 2016.
[11] D. He and D. Wang, ``Robust biometrics-based authentication scheme for multiserver environment," IEEE Syst. J., vol. 9, no. 3, pp. 816823, Sep. 2015.
[12] V. Odelu, A. K. Das, and A. Goswami, ``A secure biometrics-based multiserver authentication protocol using smart cards," IEEE Trans. Inf. Forensics Security, vol. 10, no. 9, pp. 19531966, Sep. 2015.
https://doi.org/10.1109/TIFS.2015.2439964
[13] M.-C. Chuang and M. C. Chen, ``An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics," Expert Syst. Appl., vol. 41, no. 4, pp. 14111418, Mar. 2014.
https://doi.org/10.1016/j.eswa.2013.08.040
[14] D. Mishra, A. K. Das, and S. Mukhopadhyay, ``A secure user anonymity preserving biometric-based multi-server authenticated key agreement scheme using smart cards," Expert Syst. Appl., vol. 41, no. 18, pp. 81298143, 2014.
https://doi.org/10.1016/j.eswa.2014.07.004
[15] Y. Lu, L. Li, H. Peng, and Y. Yang, ``A biometrics and smart cards-based authentication scheme for multi-server environments," Secur. Commun. Netw., vol. 8, no. 17, pp. 32193228, 2015.
https://doi.org/10.1002/sec.1246
[16] https://www.codementor.io/blog/merkle-trees-5h9arzd3n8
[17] Remya Chandran and A. Sasi Kumar, "An Efficient Keyless Signature and Improved Version of Merkle Signature Scheme-CMSS", International Journal of Engineering and Advanced Technology (IJEAT), Volume 8, Issue 5, June 2019.