Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 13 (2018) pp. 056-065

A Secure Access Authentication Scheme for Multiserver Environments using Neural Cryptography

Sayantica Pattanayak[1], **Simone A. Ludwig**[2]

[1]North Dakota State University, Department of Computer Science,
Fargo, ND 58105, USA
sayantica.pattanayak@ndsu.edu

[2]North Dakota State University, Department of Computer Science,
Fargo, ND 58105, USA
simone.ludwig@ndsu.edu

Abstract: **With the rapid growth of network technologies, remote servers provide resources to be accessed over an open network around the world. Mainly due to the convenience of the Internet, distant users can share information with each other. In a distributed environment, secure communication in insecure communication networks is a very important issue that needs to be addressed. Hence, user authentication and secret key distribution become the most important security services for communication networks. A common feature of conventional password authentication schemes is that they use a verification table. The verification table consists of users identities and the encrypted passwords. The verification table is securely stored by the server. If the verification table is stolen or modified by an adversary, the system will be breached. Also, the conventional password authentication system is applicable to a single server. In this paper, we designed a remote password authentication scheme for a multiserver environment. Our remote password based authentication method is based on artificial neural networks. In the first part of the paper we will show how users will communicate with different servers securely. In the second part of the paper we experimented with 540 passwords applying different classifiers. The experimental results show how the accuracy of our model can be even further improved. Furthermore, our proposed model is a more efficient and accurate authentication scheme for multiserver environments compared to other models.**

Keywords: Authentication, Neural Cryptography, Neural Networks, Multiserver, Man-in-the-middle-attack, Replay Attack.

I. Introduction

The use of the internet has increased spectacularly over the past decades. Nowadays, privacy and security are important issues. More and more security systems are added to access control for resisting illegitimate users. With the rapid increase of all types of information systems and the explosive use of the widespread Internet services, there are many avenues for business and educational information exchanges. These are protecting information and information systems from unlawful access, preventing information theft, and eliminating information system interruption or destruction. Therefore, in order to prevent an illegal user from invading the computer system, a user would need to provide an identity to a system as a proof of being a legitimate user before he/she logs into the system. So far, there are many methods proposed to identify the legitimacy of each login user such as password, fingerprint, and typing sequence. This type of remote user authentication is widely used to identify legitimate users in the internet.

An example of a remote user authentication method is the password-based user authentication scheme, which is the most widely-used and inexpensive mechanism. It is a key security feature to gain access to confidential files, bank accounts, photos and many other private files. It relieves the user from carrying physical keys and smart cards. The user only has to memorize the ID and the password. However, the user has to remember various IDs and passwords and thus, the user will maintain a table with user IDs and passwords. This table however can be accessed by a malicious user who can alter the information. To prevent altering the information, the authors in [1] came up with a new scheme. The new scheme proposed a new password authentication method where the traditional password table or verification table is transformed into two polynomials and two trapdoor one-way functions. However, this type of authentication scheme is time consuming. To log in to multiple servers, the user has to log in repeated times. Thus, the login with the same password to different servers poses a threat. Hence, this approach is not applicable to multiple servers.

In order to avoid security threats, a new remote password authentication scheme has to be developed. The new scheme has to be suitable for multiserver environments so that users can log in to multiple severs at once. In this paper, we propose a password authentication scheme based on Neural Networks. This system identifies legitimate users in real time using a pattern classification technique. Our proposed scheme

is applicable to multiserver network architectures. Our new scheme also shows how to securely establish a shared secret key, which is the key that will be used for encrypting/ decrypting the password. Before heading to our proposed scheme we shall briefly review the related work in this field (Section 2). Section 3 includes our proposed approach. Section 4 outlines the experimentation setup. Here, we have described different classifiers implemented to investigate the different results obtained. Section 5 presents the evaluation results of the experiments conducted. As our new scheme is a password authentication scheme we evaluated the security and privacy of the network as outlined in Section 6. Section 7 contains the conclusion.

II. Related Work

In this section, related work in the area of remote password based authentication schemes are introduced. The authors in [2] describe an efficient and secure authentication scheme for multi-server environments. Their scheme uses a hashing function to implement the mutual verification and session key agreement. The scheme does not manage the secret key table of the users and yet achieves users' anonymity. Their scheme is also nounce-based to avoid time synchronization problems. This protocol uses only a cryptographic one-way hash function for the implementation. In the same year, Hsiang and Shih [3] found that Liao and Wangs' protocol [2] is susceptible to insider attack, masquerade attack, server spoofing attack, and registration center spoofing attack, and does not provide mutual authentication. To overcome these drawbacks, the authors proposed an improved scheme over Liao and Wangs' scheme [2]. Then in 2010, the authors in [4] showed that Hsiang and Shihs' scheme [3] is insecure against the replay attack, impersonation attack and stolen smart card attack.

To overcome these problems Amin [5] proposed an efficient dynamic ID-based remote user password authentication scheme for multi-server environments. Amin claimed that his scheme could resist off-line identity guessing attack, off-line password guessing attack, privileged insider attack, user impersonation attack, many logged-in users' attack, smart card stolen attack, and session key recovery attack. However, the authors in [6] showed that the scheme proposed by Amin is vulnerable to off-line identity guessing and off-line password guessing with the smart card stolen attack.

Going back to 2003, the authors in [7] proposed a multi-server authentication protocol based on the ElGamal digital signature scheme [8] that uses simple geometric properties of the Euclidean and discrete logarithm problem concept. The server does not require to keep any verification table but the use of public keys makes this protocol computation intensive. In that same year, the authors in [9] proposed two multi-server password authentication protocols in which the user has to communicate in parallel with all authentication servers. They proved that these protocols are provably secure in the standard model. The attacker has to compromise a minimum threshold number of servers to gain any meaningful information regarding the password of a user. These two protocols differ in the way the client interacts with the different servers. However, in these schemes, the servers are equally exposed to the user as well as to the attacker. The authors in [10] proposed a password based two-server authentication protocol in which only one server was exposed to the users. The use of public keys makes this system computationally intensive. Moreover, it uses Secure Socket Layer (SSL) to establish a session key between a user and the front-end server to provide authentication but it provides only unilateral authentication.

So far we reviewed the work in password authentication based on neural networks. In 1994, the password authentication scheme based on a neural networks was initially proposed in [1]. This paper presents a new multilayer neural networks approach to identify computer users. The input vectors were made up of the time intervals between successive keystrokes created by users while typing a known sequence of characters. In 1997, the same authors [1] presented their work as a continuation of their previous work. In their previous paper only interkey times was used as feature vectors. Here, the authors used key hold time as features vectors. They found that hold times were more efficient than inter key times and the best identification performance was achieved by using both time measurements. However, both the schemes cannot withstand the replay attack [11].

Another password authentication approach based on neural networks was proposed in [12]. In this approach, a neural network is trained with the back-propagation (BP) algorithm to store the user IDs and the corresponding encrypted passwords. In this method, the system stores the weights of the trained neural network instead of the verification table [13]. As a result, the security of the system is increased. However, the scheme is not applicable for multiserver environments. In [7], the authors came up with a password authentication system based on neural networks. The scheme is applicable to multiserver environments. The authors used three types of neural network models to evaluate their performance. They used the Diffie Hellman key exchange protocol to send the password from a user to the Trusted Authority. However, the scheme is vulnerable to the man-in-the-middle attack. To overcome this drawback our proposed scheme uses the three party Diffie Hellman Key exchange protocol. Hence, our scheme is not susceptible to the man-in-the-middle-attack.

III. Our Proposed Approach

We propose a remote password authentication scheme using Neural Networks. This scheme is designed for multiserver environments. In our previous paper [14], we have shown how Alice can communicate with only Bob secretly. In this paper, we developed a scheme in which Alice can communicate with multiple servers. To do that she has to register herself with a Central Authority. The Central Authority will give an access key to Alice to securely communicate with multiple servers. Figure 1 shows how Alice is communicating with different servers.

Our scheme has three participants. The participants are Trusted Authority (TA), Users, Servers. The entire process is divided into three phases: 1) Registration phase, 2) Login phase, and 3) Validation phase.

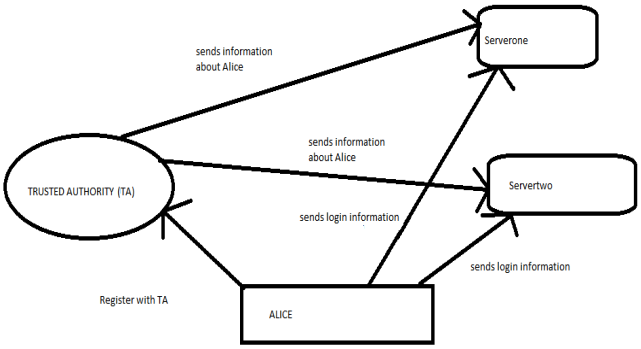

Figure. 1: Alice's communication with two different servers

A. The Registration Phase

In the registration phase the new user first registers with the Trusted Authority (TA) by sharing some legitimate information. The user is granted registration for certain servers. The steps of the registration phase are as follows:

1. Using the Diffie Hellman Key exchange protocol [15], the Users, TA and the serviceable servers create a shared secret key. The User (U_i) computes
   $ID_i = E_k\ (pw)$
   and sends ID_i to the TA. Here pw is the password created by the user. Without the key k no one can compute ID_i.

2. In this step, the TA computes the password of user i as (pw) using
   $pw = D_k\ (ID_i)$
   Then, TA creates a training pattern using the password of the user i. TA adds the training pattern of the new user to create a neural network. The input units of the training pattern are the password characters. The password characters can consist of English letters and/or numerals. Each password is made up of eight characters. Then, each of the password characters are mapped into a value ranging from 0 to 62 according to a mapping table. The mapping table consists of characters like lowercase letters, uppercase letters and numerals from 0 to 9. Each of these characters are assigned a number from 0 to 62. After mapping the characters, the training set for the neural network is created. The training set consists of an output too. The output of the training set includes the number of serviceable servers. Our system has two servers denoted as "serverone" and "servertwo". For example, if a user wants to log in to both the servers, then the output will be "serverboth". Once the training process is over, the TA sends ID_i to the user.

B. Login Phase

In the login phase, suppose that U_i wants to log in to a server. The steps are as follows:

1. The user obtains a time sequence T, which is like a time stamp.

2. Then, the user computes W as such
   $W = g^{pw^T}$ mod p

3. Afterwards, the user delivers ID_i, W and T to the server.

C. Validation Phase

In the validation phase, the server receives W, ID, T at time T'. Now, the server performs the following tasks to validate the user login request:

1. First, the server calculates ΔT. ΔT denotes the expected legal time interval for the transmission delay between the login terminal and the system servers. Then, the server checks the validity of the time stamp. If the time interval between T and T' is greater than ΔT, the server rejects the request.

2. If the time stamp T is within the valid period ΔT, the server decrypts ID_i using the shared secret key
   $pw = D_k\ (ID)$.

3. The Server after obtaining the password pw, verifies if the following equation holds
   $W = g^{pw^T}$ mod p.

4. If the previous verification holds, the server validates the user. To provide service to the user, the server first maps each of the eight characters of the password into a value according to the mapping table. Then, the server sends this values as an input to the neural network to obtain the output. The output represents that the user is authorized to use the server.

IV. Experimentation

In the experimentation phase, we used the training pattern to test the learning ability and performance of our neural network model. Our method is a supervised learning method. We used the python programming language to implement the model. We also used the sklearn machine learning library which provides state-of-the-art machine learning algorithms. We assumed the multiserver has two servers denoted as "serverone" and "servertwo". Using our remote password authentication scheme, the user can log in to any one of the servers or both the servers. We also assumed that our multiserver authentication scheme has 540 users. Each user password consists of eight characters. Therefore, our neural network model has eight inputs and one output. The output of the neural network consists of "serverone", "servertwo" and "serverboth". For instance, the password of U_1 is "kf12ghty" and the user is granted registration for server one and server two. Then, the expected output of the neural network will be "serverboth".

A. Training

Our neural network is based on the multilayer perceptron model (MLP) [16]. Each training pattern has 9 values. The feature vectors are each represented by "a", "b", "c", "d", "e", "f", "g", "h". In this experiment, we assume that we

have 540 users and two servers. Hence, there are 8 inputs and one output. The training set is then the input to train the network.

B. Classification

After the training phase, each server validates if the pw, Id and W are correct. If ID, pw and W are correct, the server accepts the request. Then, the server will input the password as feature vectors to compute the classification output. To compare the accuracy and performance of our neural network we used different classifiers. The classifiers that we used are as follows [17]:

1. GaussianNB is based on Naive Bayes Methods. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. GaussianNB implements the Gaussian Naive Bayes algorithm for classification.

2. Decision Tree is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

3. Support Vector machine is denoted as SVM and is a set of supervised learning methods used for classification, regression and outlier detection. These SVM are effective in high dimensional spaces as different kernel functions can be specified for the decision functions. In addition, they are versatile too. Support Vector Classifiers supports kernels like "rbf", "sigmoid", "poly" and "linear". Table 1 list the different instances used with their corresponding parameters.

4. Gradient Boosting (GB) is a machine learning technique for regression and classification. GB builds an additive model in a forward stage-wise fashion.

5. A Random Forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and controls over-fitting.

6. The Extra Tree class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and controls over-fitting.

7. The Logistic Regression class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', and 'lbfgs' solvers. It can handle both dense and sparse input.

8. K-nearest Neighbors implements learning based on the k-nearest neighbors of each query point, where k is an integer value specified by the user.

Table 1: SVM instances with corresponding parameters

Classifier	Parameters
SVC 1	kernel="rbf",C=1,=0.8
SVC 2	kernel="rbf",C=0.001,=1
SVC 3	kernel="sigmoid",C=1,=0.8
SVC 4	kernel="sigmoid",C=0.001,=1
SVC 5	kernel="Linear",C=1
SVC 6	kernel="Linear",C=0.001
SVC 7	kernel="poly",C=1,=0.8,degree=3
SVC 8	kernel="poly",C=0.001,=1,degree=3
SVC 9	kernel="poly",C=0.001,=1,degree=2
SVC 10	kernel="poly",C=0.001,=1,degree=1
LinearSVC 1	LinearSVC,multi-class="ovr",C=1
LinearSVC 2	LinearSVC,multi-class="crammer",C=11

V. Evaluations And Results

For the experiments, we evaluated the accuracy of the different classifiers mentioned above. Each classifier is executed ten times and then the average of them was taken. Then, we compared the results of each classifier. Table 2 shows the first ten iterations of different classifiers.

Since Support Vector Classifiers (SVC) have different kernels, we have listed all the kernels in our classification table (Table 3). The table shows the average iterations of all the classifiers mentioned above. From the table, we can see that Gradient Boosting, Random Forest, Logistic Regression and Kneighbors outperforms the other classifiers.

Table 3: Classification Accuracy of Different Classifiers

Classifiers	Accuracy
GaussianNB	0.80324
Decisian Tree	0.86180
Gradient Boosting	**0.90231**
Random Forest	**0.90695**
Extra Tree	0.88310
Logistic Regression	**0.91643**
KNeighbors	**0.90138**
LinearSVC 1,c=1	0.78700
LinearSVC 2,c=1	0.48840
SVC 1	0.88950
SVC 2	0.30902
SVC 3	0.20995
SVC 4	0.37466
SVC 5	0.69537
SVC 6	0.52016
SVC 7	0.79677
SVC 8	0.85000
SVC 9	0.85550
SVC 10	0.56110

Figure 2 shows the box plot of the classifiers accuracy with their standard deviations. The figures gives us a sketch of significance difference of the classifiers.
However, to get a detailed understanding of the statistical significance we used the Tukey's significance difference test. Table 4 shows the way the classifiers are grouped into different classes. From the table, we can say that the means which do not share the same classes are statistically significantly different. Classifiers such as Logistic Regression, Random Forest, Gradient Boosting, Kneighbors, SVC 1, Extra Tree, Decision tree, SVC 10, SVC 8, GaussianNB, SVC 7, LinearSVC 1, SVC 5 are statistically significantly different from SVC 2, SVC 4 and SVC 3. Also GaussianNB, SVC

Table 2: Classification accuracy of different classifiers - Iterations 1-10

Iterations	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
GaussianNB	0.8657	0.8287	0.6829	0.7617	0.8333	0.8218	0.7708	0.7708	0.8426	0.8542
DecisionTree	0.9796	0.8981	0.8611	0.8519	0.8125	0.8241	0.8704	0.8567	0.9120	0.8519
GradientBoosting	0.9097	0.9097	0.8843	0.9097	0.9190	0.8912	0.8912	0.8889	0.9097	0.9097
RandomForest	0.9144	0.9120	0.9213	0.9167	0.8958	0.8912	0.8981	0.9028	0.9144	0.9028
ExtraTree	0.8634	0.8889	0.8958	0.8727	0.8819	0.8773	0.8750	0.9028	0.9020	0.8704
LogisticRegression	0.9051	0.9005	0.9236	0.9097	0.9120	0.9213	0.9329	0.9259	0.9051	0.9282
Kneighbors	0.9120	0.9213	0.8843	0.9074	0.8750	0.8819	0.9167	0.9074	0.8958	0.9120
LinearSVC 1	0.8380	0.5116	0.9282	0.7986	0.8958	0.7593	0.9144	0.8333	0.9190	0.4722
LinearSVC 2	0.6898	0.7824	0.1806	0.6227	0.4074	0.6481	0.5579	0.1389	0.5486	0.3079
SVC 1	0.8935	0.9074	0.8750	0.9074	0.8843	0.8634	0.8981	0.9074	0.8704	0.8889
SVC 2	0.9282	0.0602	0.0255	0.9259	0.0579	0.0440	0.9282	0.0208	0.0463	0.0532
SVC 3	0.9282	0.0162	0.9282	0.0417	0.0208	0.0162	0.0231	0.0255	0.0556	0.0440
SVC 4	0.0622	0.9282	0.0162	0.0208	0.0255	0.0519	0.0532	0.0208	0.0509	0.9236
SVC 5	0.6111	0.7824	0.8264	0.6644	0.7917	0.7014	0.7037	0.6296	0.4907	0.2523
SVC 6	0.3889	0.4468	0.5602	0.6088	0.4792	0.7616	0.4931	0.5556	0.4190	0.4884
SVC 7	0.7593	0.7639	0.8079	0.8218	0.8356	0.8380	0.7963	0.7083	0.8218	0.8148
SVC 8	0.8819	0.8333	0.8241	0.8657	0.8380	0.8843	0.9005	0.7454	0.8912	0.8356
SVC 9	0.8611	0.8472	0.8102	0.8750	0.8218	0.8773	0.8310	0.8981	0.8727	0.8611
SVC 10	0.5949	0.5972	0.6574	0.4722	0.4884	0.4792	0.6505	0.5718	0.5532	0.5463

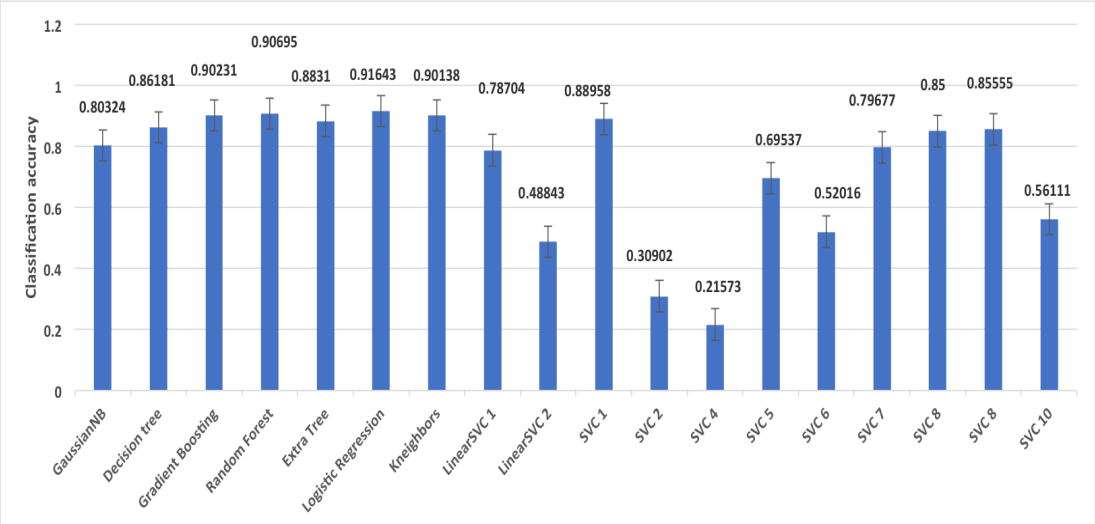

Figure. 2: Classification accuracy chart of different classifiers

7, LinearSVC 1 are statistically significantly different from LinearSVC 2, SVC 2, SVC 4 and SVC 3. Table 5, 6 and 7 shows the pairwise comparison of different classifiers with respect to their statistical significance difference and p value. We denoted the statistical significant difference as "positive" if their p value is less than or equal to 0.005.

Table 4: Grouping Information of different classifiers

Factor	N	Mean	Class
Logistic Regression	10	0.91643	A
Random Forest	10	0.90695	A
Gradient Boosting	10	0.90231	A
Kneighbors	10	0.90138	A
SVC 1	10	0.88958	A
Extra Tree	10	0.88310	A
Decision tree	10	0.86181	A
SVC 10	10	0.85555	A
SVC 8	10	0.85000	A
GaussianNB	10	0.80320	A,B
SVC 7	10	0.79680	A,B,C
LinearSVC 1	10	0.78700	A,B,C
SVC 5	10	0.69540	A,B,C,D
SVC 9	10	0.56110	B,C,D,E
SVC 6	10	0.52020	C,D,E
LinearSVC 2	10	0.48840	D,E,F
SVC 2	10	0.30900	E,F
SVC 4	10	0.21600	F
SVC 3	10	0.21000	F

VI. Security Analysis

In this section, we analyzed the security part of our neural network model. Since our new remote authentication method is based on the generation and distribution of the keys, we should be aware that there can be possible attacks on our neural network model. In this section, we will investigate privacy, nonforgability and replay resistance.

A. Privacy

In our authentication scheme we used the Diffie Hellman Key (DHK) exchange protocol. To keep the password secret and yet transfer the password to the server and TA, the generation of the keys is done by the DHK exchange protocol. A third party can try to change the shared secret key while the user and the server are communicating (man-in-the-middle-attack). Then, the third party has to change the key while the server and TA or TA and the User are communicating. It is not possible for the same third party to be present in both places at one time. Also, the third party has to continue to be in the middle. If the third party is ever absent, then the user and the server will get to know about his/her presence. Also, for our security the key is established once.

We choose our prime number p to be very large, so that it is not possible for any adversary to traverse through all the prime numbers in polynomial time. The message W cannot be broken. The security benefit comes from the difficulty of solving the discrete logarithm [18] problem. Also, the trusted authority gives the key only to the server/servers the user wants to log in. If the user wants to log in to a different server, the user again has to go through the registration phase to receive a different key.

B. Nonforgability

A legal user wants to access a non-serviceable server. The server will accept the ID to decrypt it. Then, the server will provide it as input to the neural network to receive its output. If the output is different then the server will not accept the request. Hence, our new remote server authentication/validation scheme is nonforgeable.

C. Replay Resistance

A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it. To prevent the replay attack our scheme associates a time T with ID. When an adversary replays a intercepted message to pretend a valid user, he/she has to pass Step 1 of the validation phase. The adversary will create a timestamp $T*$ such that $T" - T* \leq \Delta$T. Once T is changed to $T*$ then W changes and hence the password, thus the adversary will fail the validation phase. However, if ΔT is too large, the server cannot resist the replay attack. Hence, it is very important to choose an appropriate ΔT.

VII. Conclusions

In this paper, we designed a remote password authentication scheme for multiserver environments as follows. The user will register first with a trusted authority to become a legitimate user. Each legitimate user has its own ID and password. Then, the user types his/her user ID and password to log in to any of the servers. Later, during the authentication phase the servers validate the legitimacy of the remote login user. The remote password authentication scheme for multiserver environments is based on artificial neural networks. To overcome the problem of the man-in-the-middle-attack we created our shared secret key using the three party Diffie Hellman Key exchange protocol.

For the experimentation, we use 540 users with different passwords of eight characters. Each of these eight characters represent the input feature vectors of our neural network. The output represents a number of serviceable servers. Each user can log in to any one of the servers or both the server at one time. After training the neural network, we compared different classifiers in terms of efficiency and accuracy.

Our findings shows that Gradient Boosting, Random Forest, Logistic Regression and K neighbors outperforms the other classifiers. However, to identify whether a classifier is statistically significantly different from the other we used the Tukeys test method. The test method found that Logistic Regression, Random Forest, Gradient Boosting and Kneighbors are not statistically significantly different from SVC 1, Extra Tree, Decision tree, SVC 8, SVC 10 and Gaussian NB. From tables 5, 6 and 7 we get a more accurate picture of statistical significant difference of each classifier as compared to other classifier.

Future work will focus on a decentralized system so that we will have only users and the servers without the trusted authority. This will make the password authentication scheme more secure since the user should only communicate and register with the servers without sharing any legitimate informa-

Table 5: Difference of means to calculate statistical significant difference

Difference of Levels	**Difference of means**	**P value**	**Statistical Significance Difference**
Decision tree - GaussianNB	0.0586	1	Negative
Gradient Boosting - GaussianNB	0.0991	0.99896	Negative
Random Forest - GaussianNB	0.1037	0.99815	Negative
Extra Tree - GaussianNB	0.0799	0.99995	Negative
Logistic Reg - GaussianNB	0.1132	0.99471	Negative
Kneighbors - GaussianNB	0.0981	0.99908	Negative
LinearSVC 1 - GaussianNB	-0.0162	1	Negative
LinearSVC 2 - GaussianNB	-0.3148	0.01077	Positive
SVC 1 - GaussianNB	0.0863	0.99984	Negative
SVC 2 - GaussianNB	-0.4942	5.8E-05	Positive
SVC 3 - GaussianNB	-0.5933	5.8E-05	Positive
SVC 4 - GaussianNB	-0.5875	5.8E-05	Positive
SVC 5 - GaussianNB	-0.1079	0.99701	Negative
SVC 6 - GaussianNB	-0.2831	0.04199	Positive
SVC 7 - GaussianNB	-0.0065	1	Negative
SVC 8 - GaussianNB	0.0468	1	Negative
SVC 9 - GaussianNB	0.0523	1	Negative
SVC 10 - GaussianNB	-0.2421	0.17933	Negative
Gradient Boosting - Decision tree	0.0405	1	Negative
Random Forest - Decision tree	0.0451	1	Negative
Extra Tree - Decision tree	0.0213	1	Negative
Logistic Reg - Decision tree	0.0546	1	Negative
Kneighbors - Decision tree	0.0396	1	Negative
LinearSVC 1 - Decision tree	-0.0748	0.99998	Negative
LinearSVc 2 - Decision tree	-0.3734	0.00061	Positive
SVC 1 - Decision tree	0.0278	1	Negative
SVC 2 - Decision tree	-0.5528	5.8E-05	Positive
SVC 3 - Decision tree	-0.6519	5.8E-05	Positive
SVC 4 - Decision tree	-0.6461	5.8E-05	Positive
SVC 5 - Decision tree	-0.1664	0.81444	Negative
SVC 6 - Decision tree	-0.3417	0.003	Positive
SVC 7 - Decision tree	-0.0650	1	Negative
SVC 8 - Decision tree	-0.0118	1	Negative
SVC 9- Decision tree	-0.0063	1	Negative
SVC 10 - Decision tree	-0.3007	0.02017	Positive
Random Forest - Gradient Boosting	0.0046	1	Negative
Extra Tree - Gradient Boosting	-0.0192	1	Negative
Logistic Reg - Gradient Boosting	0.0141	1	Negative
Kneighbors - Gradient Boosting	-0.0009	1	Negative
LinearSVC 1 - Gradient Boosting	-0.1153	0.99348	Negative
LinearSVC 2 - Gradient Boosting	-0.4139	0.00011	Positive
SVC 1 - Gradient Boosting	-0.0127	1	Negative
SVC 2 - Gradient Boosting	-0.5933	5.8E-05	Positive
SVC 3 - Gradient Boosting	-0.6924	5.8E-05	Positive
SVC 4 - Gradient Boosting	-0.6866	5.8E-05	Positive
SVC 5 - Gradient Boosting	-0.2069	0.44581	Negative
SVC 6 - Gradient Boosting	-0.3821	0.0004	Positive
SVC 7 - Gradient Boosting	-0.1055	0.99771	Negative
SVC 8 - Gradient Boosting	-0.0523	1	Negative
SVC 9 - Gradient Boosting	-0.0468	1	Negative
SVC 10 - Gradient Boosting	-0.3412	0.00307	Positive
Extra Tree - Random Forest	-0.0239	1	Negative
Logistic Reg - Random Forest	0.0095	1	Negative
Kneighbors - Random Forest	-0.0056	1	Negative
LinearSVC 1 - Random Forest	-0.1199	0.98986	Negative

Table 6: Difference of means to calculate statistical significant difference

Difference of Levels	Difference of means	P value	Statistical Significance Difference
SVC 1 - Extra Tree	-0.5741	0.000058	Positive
SVC 3 - Extra Tree	-0.6732	0.000058	Positive
SVC 4 - Extra Tree	-0.6674	0.000058	Positive
SVC 5 - Extra Tree	-0.1877	0.630259	Negative
SVC 6 - Extra Tree	-0.3629	0.001031	Positive
SVC 7 - Extra Tree	-0.0863	0.999837	Negative
SVC 8 - Extra Tree	-0.0331	1	Negative
SVC 9 - Extra Tree	-0.0275	1	Negative
SVC 10 - Extra Tree	-0.3220	0.007728	Positive
Kneighbors - Logistic Reg	-0.0151	1	Negative
LinearSVC 1 - Logistic Reg	-0.1294	0.977455	Negative
LinearSVC 2 - Logistic Reg	-0.4280	0.00008	Positive
SVC 1 - Logistic Reg	-0.0269	1	Negative
SVC 2 - Logistic Reg	-0.6074	0.000058	Positive
SVC 3 - Logistic Reg	-0.7065	0.000058	Positive
SVC 4 - Logistic Reg	-0.7007	0.000058	Positive
SVC 5 - Logistic Reg	-0.2211	0.32243	Negative
SVC 6 - Logistic Reg	-0.3963	0.000209	Positive
SVC 7 - Logistic Reg	-0.1197	0.990088	Negative
SVC 8 - Logistic Reg	-0.0664	0.999997	Negative
SVC 9 - Logistic Reg	-0.0609	0.999999	Negative
SVC 10 - Logistic Reg	-0.3553	0.001515	Positive
LinearSVC 1 - Kneighbors	-0.1143	0.994052	Negative
LinearSVC 2 - Kneighbors	-0.4129	0.000114	Positive
SVC 1 - Kneighbors	-0.0118	1	Negative
SVC 2 - Kneighbors	-0.5924	0.000058	Positive
SVC 3 - Kneighbors	-0.6914	0.000058	Positive
SVC 4 - Kneighbors	-0.6856	0.000058	Positive
SVC 5 - Kneighbors	-0.2060	0.454487	Negative
SVC 6 - Kneighbors	-0.3812	0.000415	Positive
SVC 7 - Kneighbors	-0.1046	0.997942	Negative
SVC 8 - Kneighbors	-0.0514	1	Negative
SVC 9 - Kneighbors	-0.0458	1	Negative
SVC 10 - Kneighbors	-0.3403	0.003212	Positive
LinearSVC 1 - LinearSVC 2	-0.2986	0.022071	Positive
SVC 1 - LinearSVC 1	0.1025	0.998396	Negative
SVC 2 - LinearSVC 2	-0.4780	0.000059	Positive
SVC 3 - LinearSVC 1	-0.5771	0.000058	Positive
SVC 4 - LinearSVC 2	-0.5713	0.000058	Positive
SVC 5 - LinearSVC 1	-0.0917	0.999628	Negative
SVC 6 - LinearSVC 2	-0.2669	0.078013	Positive
SVC 7 - LinearSVC 1	0.0097	1	Negative
SVC 8 - LinearSVC 2	0.0630	0.999999	Negative
SVC 9 - LinearSVC 1	0.0685	0.999995	Negative
SVC 10 - LinearSVC 2	-0.2259	0.284575	Negative
SVC 1 - LinearSVC 1	0.4011	0.000171	Positive
SVC 2 - LinearSVC 2	-0.1794	0.707607	Negative
SVC 3 - LinearSVC 1	-0.2785	0.050346	Positive
SVC 4 - LinearSVC 2	-0.2727	0.06285	Negative
SVC 5 - LinearSVC 1	0.2069	0.445806	Negative
SVC 6 - LinearSVC 2	0.0317	1	Negative
SVC 7 - LinearSVC 1	0.3083	0.01442	Positive
SVC 8 - LinearSVC 2	0.3616	0.001105	Positive
SVC 9 - LinearSVC 1	0.3671	0.000835	Positive
SVC 10 - LinearSVC 2	0.0727	0.999987	Negative

Table 7: Difference of means to calculate statistical significant difference

Difference of Levels	**Difference of means**	**P value**	**Statistical Significance Difference**
SVC 2 - SVC 1	-0.5806	0.000058	Positive
SVC 3 - SVC 1	-0.6796	0.000058	Positive
SVC 4 - SVC 1	-0.6738	0.000058	Positive
SVC 5 - SVC 1	-0.1942	0.567748	Negative
SVC 6 - SVC 1	-0.3694	0.000743	Positive
SVC 7 - SVC 1	-0.0928	0.999561	Negative
SVC 8 - SVC 1	-0.0396	1	Negative
SVC 9 - SVC 1	-0.0340	1	Negative
SVC 10 - SVC 1	-0.3285	0.00569	Positive
SVC 3 - SVC 2	-0.0991	0.998964	Negative
SVC 4 - SVC 2	-0.0933	0.999529	Negative
SVC 5 - SVC 2	0.3863	0.000324	Positive
SVC 6 - SVC 2	0.2111	0.407299	Negative
SVC 7 - SVC 2	0.4878	0.000058	Positive
SVC 8 - SVC 2	0.5410	0.000058	Positive
SVC 9 - SVC 2	0.5465	0.000058	Positive
SVC 10 - SVC 2	0.2521	0.130564	Negative
SVC 4 - SVC 3	0.0058	1	Negative
SVC 5 - SVC 3	0.4854	0.000059	Positive
SVC 6 - SVC 3	0.3102	0.013264	Positive
SVC 7 - SVC 3	0.5868	0.000058	Positive
SVC 8 - SVC 3	0.6400	0.000058	Positive
SVC 9 - SVC 3	0.6456	0.000058	Positive
SVC 10 - SVC 3	0.3512	0.001869	Positive
SVC 5 - SVC 4	0.4796	0.000059	Positive
SVC 6 - SVC 4	0.3044	0.017145	Positive
SVC 7 - SVC 4	0.5810	0.000058	Positive
SVC 8 - SVC 4	0.6343	0.000058	Positive
SVC 9 - SVC 4	0.6398	0.000058	Positive
SVC 10 - SVC 4	0.3454	0.002494	Positive
SVC 6 - SVC 5	-0.1752	0.744437	Negative
SVC 7 - SVC 5	0.1014	0.998607	Negative
SVC 8 - SVC 5	0.1546	0.890284	Negative
SVC 9 - SVC 5	0.1602	0.857492	Negative
SVC 10 - SVC 5	-0.1343	0.967608	Negative
SVC 7 - SVC 5	0.2766	0.054134	Positive
SVC 8 - SVC 5	0.3298	0.00533	Positive
SVC 9 - SVC 5	0.3354	0.004078	Positive
SVC 10 - SVC 5	0.0410	1	Negative
SVC 7 - SVC 6	0.0532	1	Negative
SVC 8 - SVC 6	0.0588	1	Negative
SVC 9 - SVC 6	-0.2357	0.217453	Negative
SVC 10 - SVC 6	0.0056	1	Negative
SVC 9 - SVC 8	-0.2889	0.03319	Positive
SVC 10 - SVC 8	-0.2944	0.026349	Positive

tion with the servers. However, the servers will authenticate the user to communicate with the servers.

References

[1] M.S. Obaidat, D.T. Macchairolo. A multilayer neural network system for computer access security, *IEEE Transactions on Systems, Man, and Cybernetics,* pp. 806-813, 1994.

[2] Y.P. Liao, S.S Wang. A secure dynamic ID based remote user authentication scheme for multi-server environment, *Computer Standards & Interfaces,* pp. 24-29, 2009.

[3] H.C. Hsiang, W.K. Shih. Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment, *Computer Standards & Interfaces,* pp. 1118-1123, 2009.

[4] S.K. Sood, A.K. Sarje, K. Singh. A secure dynamic identity based authentication protocol for multi-server architecture, *Journal of Network and Computer Applications,* pp. 609-618, 2011.

[5] R. Amin. Cryptanalysis and Efficient Dynamic ID Based Remote User Authentication Scheme in Multi-server Environment Using Smart Card, *IJ Network Security,* pp. 172-181, 2016.

[6] H.T. Pan, C.S Pan, S.C. Tsaur, M.S. Hwang. Cryptanalysis of Efficient Dynamic ID Based Remote User Authentication Scheme in Multi-server Environment Using Smart Card, *Computational Intelligence and Security (CIS), 12th International Conference on. IEEE,* pp. 590-593, 2016.

[7] L.H. Li, L.C Lin, M.S Hwang. A remote password authentication scheme for multiserver architecture using neural networks, *IEEE Transactions on Neural Networks,* pp. 1498-1504, 2001.

[8] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE transactions on information theory* pp. 469-472, 1985.

[9] M.D. Raimondo, R. Gennaro. Provably secure threshold password-authenticated key exchange, *International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg,* pp. 507-523, 2003.

[10] J.G. Brainard, A. Juels, B. Kaliski, M. Szydlo. A New Two-Server Approach for Authentication with Short Secrets, *USENIX Security Symposium,* pp. 201-214, 2003.

[11] J. Yang, J. Park, H. Lee, K. Ren, K. Kim. Mutual authentication protocol, emphWorkshop on RFID and lightweight crypto, 2005.

[12] S.Z. Reyhani, M. Mahdavi. User authentication using neural network in smart home networks, *International Journal of Smart Home,* pp. 147-154, 2007.

[13] L. Iuon-Chang, H.H. Ou, M.S. Hwang. A user authentication system using back-propagation network, *Neural Computing & Applications,* pp. 243-249, 2005.

[14] S. Pattanayak, S.A. Ludwig. Encryption Based on Neural Cryptography, *International Conference on Health Information Science. Springer, Cham,* pp. 321-330, 2017.

[15] H. K. Pathak, M. Sanghi. Simple Three Party Key Exchange Protocol via Twin Diffie-Hellman Problem, *IJ Network Security,* pp. 256-264, 2013.

[16] F. Rosenblatt. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.

[17] Scikit-learn - Machine Learning in Python, *scikit-learn.org*, last retrieved in May 2018.

[18] E. Bresson, O. Chevassut, D. Pointcheval, Q. Jean-Jacques. Provably authenticated group Diffie-Hellman key exchange, *Proceedings of the 8th ACM conference on Computer and Communications Security ACM,* pp. 255-265, 2001.

Author Biographies

Sayantica Pattanayak obtained her MS degree from the Department of Electrical and Computer Engineering at North Dakota State University, USA in 2015. Sayantica is currently pursuing her PhD in the Department of Computer Science at North Dakota State University, USA. Her research area of interest includes neural networks neural cryptography and cryptography.

Simone A. Ludwig is a Professor of Computer Science at North Dakota State University, USA. She received her PhD degree and MSc degree with distinction from Brunel University (UK), in 2004 and 2000, respectively. Her research interests lie in the area of computational intelligence including swarm intelligence, evolutionary computation, neural networks, and fuzzy reasoning. Application areas of the applied computational intelligence methods/algorithms are data mining (including big data), image processing, and cloud computing.