PROVABLY SECURED TWO SERVER HASH PASSWORD AUTHENTICATION

[1]T.S.Thangavel [2]Dr. A. Krishnan

[1] AP/ Dept of M.Sc(IT), K.S.Rangasamy College of Technology, Tamilnadu, India .

tsthangavel123@yahoo.in

[2] Dean , K.S.Rangasamy College of Technology, Tamilnadu, India.

ABSTRACT

The techniques of secured socket layer (SSL) with client-side certificates for commercial web sites rely on a relatively weak form of password authentication. Browser sends a user's plaintext password to a remote web server using SSL is vulnerable to attack. In common password attacks, hackers exploit the fact that web users often use the same password at many different sites. This has drawn attention on the need for new hash function designs. In addition the authentication systems which uses passwords stored in a central server is easily prone to attack. To overcome the problem of single server password attacks, the multi-server systems were proposed in which user communicates in parallel with several or all of the servers. Such system requires a large communication bandwidth, complex deployment, needs for synchronization at the user and quite expensive. Optimized two server system is proposed in our work.
The proposal of our work presents a user interface, browser extension password hash, strengthens web password in a two server authentication system. The hash is implemented using a Pseudo Random Function keyed by the password. Since the hash output is tailored to meet server password requirements, the resulting hashed password is handled normally at the server with no server modifications. The two server authentication system is interfaced with client supported hash passwords with server session keys. The two server system contains, the front end service server interacts directly to the user and the back end control server visible to the service server. The users contact only the service server but these two servers are responsible for the authentication of the user. The user has a password which is transformed into two long secrets which are held by service server and control server.

Keywords: *Hash Function, Pseudo random Function,, Password Hash, Two Server Password Authentication , Control Server, Service server,*

1. INTRODUCTION

A random password generator is software program or hardware device that takes input from a random or pseudo-random number generator and automatically generates a password. Random passwords can be generated manually, using simple sources of randomness such as dice or coins, or they can be generated using a computer. While there are many examples of "random" password generator programs available on the Internet, generating randomness can be tricky and many programs do not generate random characters in a way that ensures strong security. A common recommendation is to use open source security tools where possible, since they allow independent checks on the quality of the methods used. Note that simply generating a password at random does not ensure the password is a strong password, because it is possible, although highly unlikely, to generate an easily guessed or cracked password.

A password generator can be part of a password manager. When a password policy enforces complex rules, it can be easier to use a password generator based on that set of rules than to manually create passwords. In situations where the attacker can obtain an encrypted version of the password, such testing can be performed rapidly enough so that a few million trial passwords can be checked in a matter of seconds. The function rand presents another problem. All pseudo-random number generators have an internal

memory or state**.** The size of that state determines the maximum number of different values it can produce, an n-bit state can produce at most 2^n different values. On many systems rand has a 31 or 32 bit state, which is already a significant security limitation. Some computer operating systems provide much stronger random number generators.

Most password-based user authentication systems place total trust on the authentication server where passwords or easily derived password verification data are stored in a central database. These systems could be easily compromised by offline dictionary attacks initiated at the server side. Compromise of the authentication server by either outsiders or insiders subjects all user passwords to exposure and may have serious problems. To overcome these problems in the single server system many of the systems has been proposed such as multi-server systems, public key cryptography and password systems, threshold password authentication systems, two server password authentication systems.

The proposed work continues the line of research on the two-server paradigm in [3], extend the model by imposing different levels of trust upon the two servers, and adopt a very different method at the technical level in the protocol design. As a result, we propose a practical two-server password authentication and key exchange system that is secure against offline dictionary attacks by servers when they are controlled by adversaries. The proposed scheme is a password-only system in the sense that it requires no public key cryptosystem and, thus, no PKI. This makes the system very attractive considering PKIs are proven notoriously expensive to deploy in real world. Moreover, the proposed system is particularly suitable for resource constrained users due to its efficiency in terms of both computation and communication. Our work, generalize the basic two-server model to an architecture of a single back-end server supporting multiple front-end servers and envision interesting applications in federated enterprises.

2. LITERATURE REVIEW

Computer applications may require random numbers in many contexts. Random numbers can be used to simulate natural or artificial phenomena in computer simulations, many algorithms that require randomness have been developed that outperform deterministic algorithms for the same problem, and random numbers can be used to generate or verify passwords for cryptography-based computer security systems. The present invention relates to the use of random numbers in such security systems, called as cryptographic applications.[1] Security system based on English text passwords, are susceptible to a dictionary attack. Thus, in order to ensure the utmost security, it is essential that the security system implements a method for generating a random number that appears completely random. In this manner, a completely random password or cryptographic key presents no opening or prior knowledge that can be exploited by an hostile agent.[2].

A chaotic system is one with a state that changes over time in a largely unpredictable manner. To use the chaotic system to generate a random
number, there is some means of converting the state of the system into a sequence of bits (i.e., a binary number). These chaotic systems can be converted to produce binary numbers by using standard techniques. [4] For instance, a pseudo-random binary string can be generated from the digital recording of static noise via a digital microphone. Alternatively, a noisy diode can be sampled at a suitable frequency and converted into a digital signal, or a picture of an area of the sky can be taken and subsequently scanned and digitized. These resulting binary strings that are generated over time are generally random in nature. [15].

However, web password hashing is often implemented incorrectly by giving the remote site the freedom to choose the salt. For example, HTTP1.1 Digest Authentication defines password hashing as follows i.e., digest = Hash (pwd, realm, nonce, username,,,) where realm and nonce are specified by the remote web site. Hence, using an online attack, a phisher could send to the user the realm and nonce the phisher received from the victim site. The user's response provides the phisher with a valid password digest for the victim site. Password hashing implemented in Kerberos 5 has a similar vulnerability. The first systems we are aware of that provide proper web password hashing are the Lucent Personal Web Assistant [2] and a system from DEC SRC [1]. It was necessary to build PwdHash as a browser extension so that we could alter passwords before SSL encryption. Although it might be feasible to build a proxy that forges SSL certificates on the fly (essentially mounting a man in the middle attack on SSL), such a proxy would not be able to identify or

protect passwords that are typed into mock password fields.

Number of existing applications including Mozilla Firefox provide convenient password management by storing the user's web passwords on disk, encrypted under some master password.[5] When the user tries to log in to a site, the application asks for the master password and then releases the user's password for that site. Thus, the user need only remember the master password. The main drawback compared to Password Hash is that the user can only use the web on the machine that stores his passwords. However, password management systems do provide stronger protection against dictionary attacks when the user chooses a unique, high entropy password for each site. However, many users may fail to do this.

Most of the existing password systems were designed over a single server, where each user shares a password or some password verification data (PVD) with a single authentication server (e.g., [7], [8], [9]). These systems are essentially intended to defeat offline dictionary attacks by outside attackers and assume that the sever is completely trusted in protecting the user password database. Unfortunately, attackers in practice take on a variety of forms, such as hackers, viruses, worms, accidents, mis-configurations, and disgruntled system administrators. Once an authentication server is compromised, all the user passwords or PVD fall in the hands of the attackers, who are definitely effective in offline dictionary attacks against the user passwords. To eliminate this single point of vulnerability inherent in the single-server systems, password systems based on multiple servers were proposed.

The system in [11], believed to be the first multi-server password system, splits a password among multiple servers. However, the servers in [11] need to use public keys. An improved version of [10] was proposed in [12], which eliminates the use of public keys by the servers. Further and more rigorous extensions were due to [13], where the former built a t-out-of-n threshold PAKE protocol and provided a formal security proof under the random oracle model [14] and the latter presented two provably secure threshold PAKE protocols under the standard model. While the protocols are theoretically significant, they have low efficiency and high operational overhead. In these multi-server password systems, either the servers are equally exposed to the users and a user has to communicate in parallel with several or all servers for authentication, or a gateway is introduced between the users and the servers.

Recently, Brainard et al. [6] proposed a two-server password system in which one server exposes itself to users and the other is hidden from the public. While this two-server setting is interesting, it is not a password-only system: Both servers need to have public keys to protect the communication channels from users to servers. As we have stressed earlier, this makes it difficult to fully enjoy the benefits of a password system. In addition, the system in [1] only performs unilateral authentication and relies on the Secure Socket Layer (SSL) to establish a session key between a user and the front-end server. Subsequently, Yang et al. [14][16] extended and tailored this two-server system to the context of federated enterprises, where the back-end server is managed by an enterprise headquarter and each affiliating organization operates a front-end server. An improvement made in [14] is that only the back-end server holds a public key.

3. SECURED HASH BASED PSEUDO RANDOM PASSWORD SCHEME

The proposed methodology of the secure hash password system contains one-way hash functions that can process a message to produce a condensed representation called a message digest. This algorithm enables the determination of a message's integrity, any change to the message will, with a very high probability, results in a different message digest. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers. The algorithm is described in two stages, preprocessing and hash computation. Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message schedule from the padded message and uses that schedule, along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest. The design principle of hash functions is iterating a compression function (here denoted *F*), which

takes as input *s* bits and returns *r* bits (with $s > r$). The resulting function is then chained to operate on strings of arbitrary length (Fig 1). The validity of such a design has been established and its security is proven not worse than the security of the compression function.

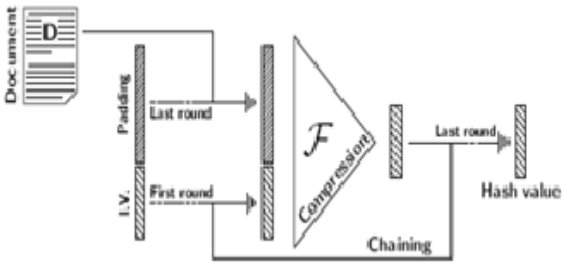

Fig 1: Iterative hash function structure

Compression Hash function Algorithm

Input: s bits of data
1. split the s input bits in w parts $s_1, \ldots, s_w$ of $\log_2(\frac{n}{w})$ bits;
2. convert each s_i to an integer between 1 and $\frac{n}{w}$;
3. choose the corresponding column in each $\mathcal{H}_i$;
4. add the w chosen columns to obtain a binary string of length r.

Output: r bits of hash

The core of the compression function is a random binary matrix *H* of size $r \times n$. The parameters for the hash function are *n* the number of columns of *H*, *r* the number of rows of *H* and the size in bits of the function output, and *w* the number of columns of *H* added at each round. Random password generators normally output a string of symbols of specified length. These can be individual characters from some character set, syllables designed to form pronounceable passwords, or words from some word list to form a passphrase. The program can be customized to ensure the resulting password complies with the local password policy, say by always producing a mix of letters, numbers and special characters. The strength of a random password can be calculated by computing the information entropy of the random process that produced it. If each symbol in the password is produced independently, the entropy is just given by the formula

$$H = L \log_2 N = L \frac{\log N}{\log 2}$$

where N is the number of possible symbols and L is the number of symbols in the password. The function $\log_2$ is the base-2 logarithm. H is measured in bits.

Symbol set	N	Entropy/symbol
Digits only (0-9) (e.g. PIN)	10	3.32 bits
Single case letters (a-z)	26	4.7 bits
Single case letters and digits (a-z, 0-9)	36	5.17 bits
Mixed case letters and digits (a-z, A-Z, 0-9)	62	5.95 bits
All standard U.S. keyboard characters	94	6.55 bits
Diceware word list	7776	12.9 bits

Thus an eight character password of single case letters and digits would have 41 bits of entropy (8 x 5.17). The same length password selected at random from all U.S. computer keyboard characters would have 52 bit entropy; however such a password would be harder to memorize and might be difficult to enter on non-U.S. keyboards. A ten character password of single case letters and digits would have essentailly the same strength (51.7 bits). Any password generator is limited by the state space of the pseudo-random number generator, if one is used. Thus a password generated using a 32-bit generator has a maximum entropy of 32 bits, regardless of the number of characters the password contains.

4. SECURED TWO SERVER PASSWORD AUTHENTICATION SCHEME

Three types of entities are involved in our system, i.e., users, a service server (SS) that is the public server in the two server model, and a control server (CS) that is the back-end server. In this setting, users only communicate with SS and do not necessarily know CS. For the purpose of user authentication, a user U has a password which is transformed into two long secrets, which are held by SS and CS, respectively. Based on their respective shares, SS and CS together

validate users during user login. CS is controlled by a passive adversary and SS is controlled by an active adversary in terms of offline dictionary attacks to user passwords, but they do not collude (otherwise, it equates the single-server model).

A passive adversary follows honest-but-curious behavior, that is, it honestly executes the protocol according to the protocol specification and does not modify data, but it eavesdrops on communication channels, collects protocol transcripts and tries to derive user passwords from the transcripts, moreover, when an passive adversary controls a server, it knows all internal states of knowledge known to the server, including its private key (if any) and the shares of user passwords. In contrast, an active adversary can act arbitrarily in order to uncover user passwords. Besides, we assume a secret communication channel between SS and CS for this basic protocol. This security model exploits the different levels of trust upon the two servers. This holds with respect to outside attackers. As far as inside attackers are concerned, justifications come from our application and generalization of the system to the architecture of a single control server supporting multiple service servers, where the control server affords and deserves enforcing more stringent security measurements against inside attackers. The back-end server is strictly passive and is not allowed to eavesdrop on communication channels, while CS in our setting is allowed for eavesdropping.

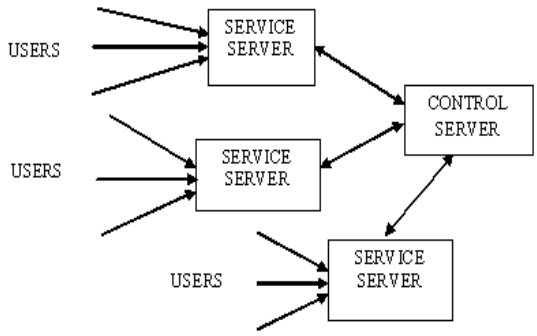

Fig 4: Generalized Two Server Architecture of a single control server with multiple service server.

5. IMPLEMENTATION OF SECURED TWO SERVER HASH PASSWORD SYSTEM

The user contacts only the service server but both the control and service servers are responsible for the authentication of the user. The user has a password which is transformed into two long secrets which are held by service server and control server. Both the system using their respective shares validate user during the login. The servers compute function to verify the user and finally a session key is being established between the user and service server for the confirmation of the user and the server. The service server (Fig 6) which is an active adversary acts arbitrarily to uncover the passwords and could control the corruption of the password, the control server which is a passive adversary acts according to the protocol specification. (Fig 5)

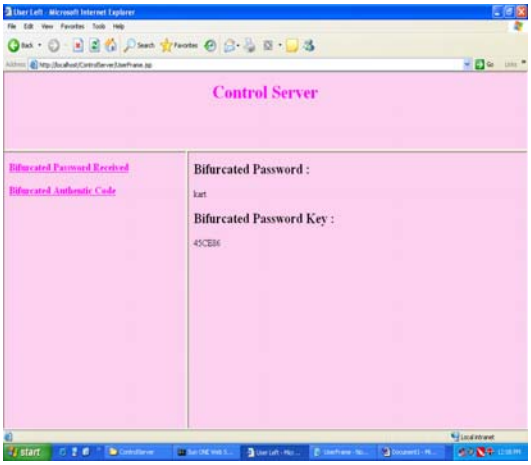

Fig 5: Control Server Key generation for user password (bifurcated)

In the offline dictionary attacks, where the successful logins between the user and the server is recorded by the intruder and it tries the passwords in the dictionary against login transcripts and this is overcome in the system by control server as passive adversary and service server as active adversary. In the system, the communication and the computations are more efficient. The user can use the same password to register to different service server, the service server connect either to distinct control servers or to the same control server. This is a highly desirable feature since it makes the system user friendly. The system could be

adapted to any existing FTP and web applications that are available today by adding a control server to it where these are managed by the administrative domain.

In our experimental implementation, a password is split into two random numbers. Therefore, a user can use the same password to register to different service servers; they connect either to distinct control servers or to the same control server. This is a highly desirable feature since it makes the system user friendly. A big inconvenience in the traditional password systems is that a user has to memorize different passwords for different applications. The system has no compatibility problem with the single-server model. This is of importance, as most of the existing password systems use a single server.

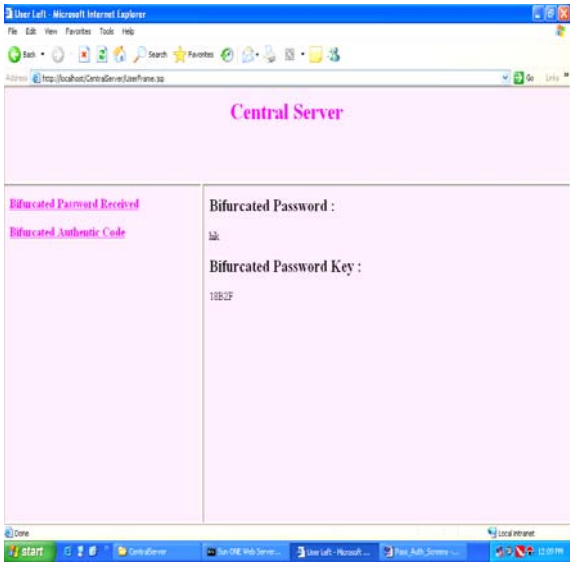

Fig 6: Service Server Key generation for user password (bifurcated)

The generalization as well as the applications of the two-server password system well support the underlying security model, in the sense that the enterprise headquarter naturally assume adequate funds and strong security expertise and, therefore, affords and is capable of maintaining a highly trustworthy control server against both inside attackers and outside attackers. Without the concern of a single point of vulnerability, affiliating organizations that operate service servers are offloaded to some extent from strict security management.

6. PASSWORD SECURITY PERFORMANCE ON TWO SERVER SYSTEM

The exponentiations dominate each party's computation overhead, the two server password authentication system only count the number of exponentiations as the computation performance. The digits before "/" denote the total number of exponentiations performed by each party, and the digits following "/" denote the number of exponentiations that can be computed offline. One round is a one-way transmission of messages. The proposed two protocols demonstrate performance quite efficient in terms of both computation and communication to all parties. Take U, for example, it needs to calculate 3 and 4 exponentiations in the two protocols, respectively, and 2 of them can be performed offline. This means U only computes 1 and 2 exponentiations in real time in the respective protocols, the communication overhead for U is particularly low in terms of both bits and rounds. The table 1 listed below indicates the computation performance in terms of time and success rate (number of rounds) of the two server password authentication and single server authentication.

Table 1: Performance measure on two server and Single server password authentication scheme

Password Security Scheme	Time of Authenticity (MilliSecond)	Success Rate (Percentage %)
Two Server Hash Password	10	96
Single Server Hash	8	87

7. DISCUSSIONS

With two-server password system, single point of vulnerability, is totally eliminated. Without compromising both servers, no attacker can find user passwords through offline dictionary attacks. The control server being isolated from the public, the chance for it being attacked is substantially minimized, thereby increasing the security of the overall system. The system is also resilient to offline dictionary attacks by outside attackers. This allows users to use easy to

remember passwords and still have strong authentication and key exchange. The system has no compatibility problem with the single-server model. The generalization of the two-server password system well supports the underlying security model. In reality, adversaries take on a variety of forms and no security measures and precautions can guarantee that a system will never be penetrated.

By avoiding a single point of vulnerability, it gives a system more time to react to attacks. The password-based authentication and key exchange system that is built upon a novel two-server model, where only one server communicates to users while the other server stays transparent to the public. Compared with previous solutions, our system possesses many advantages, such as the elimination of a single point of vulnerability, avoidance of PKI, and high efficiency. The browser extended secured password authentication tool generates the hashed password using either SHA1 or MD5 hashing algorithm depending on the choice you make. It will display the hashed password in the read only text box, it can also copy the hashed password to clipboard on your choice for easy paste operation.

8. CONCLUSION AND FUTURE WORK

The proposed two server hash password authentication techniques are designed to provide novice users with the benefits of password practices that are otherwise only feasible for security experts. The two server hash password system also overcomes the online and offline dictionary attacks that are prevailing in the single and multi-server systems. It is particularly suitable for resource-constrained users due to its efficiency in terms of both computation and communication. It is also possible for the servers to associate with multiple clients. Providing customized passwords, reduce the threat of password attacks with no server changes and little or no change to the user experience. Provably secure hash functions based password authentication scheme construction provides features such as both the block size of the hash function and the output size are completely scalable.

The developed two-server password authentication architecture has control server and service server. The control server is controlled by a passive adversary while the service server is controlled by an active adversary. A single point of vulnerability, as in the existing password systems, is totally eliminated. Without compromising both servers, no attacker can find user passwords through offline dictionary attacks. The control server being isolated from the public, the chance for it being attacked is substantially minimized, thereby increasing the security of the overall system. The system has no compatibility problem with the single-server model. In contrast to existing multi-server password systems, the two server system has great potential for practical applications. It can be directly applied to fortify existing standard single-server password applications, e.g., FTP and Web applications. The proposal of our work can be further improved in the direction of providing a feasible solution for the case of dynamic session variations between two party message communication system without compromising the credibility of the data being transferred. This can be accomplished using quantum key cryptographic technique.

REFERENCES

[1] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, Client-side defense against web based identity theft, In Proceedings of Network and Distributed Systems Security (NDSS), 2004.

[2] J. A. Halderman, B.Waters, and E. Felten A convenient method for securely managing passwords, 14th International World Wide Web Conference, 2005.

[3] Yanjiang Yang, Robert H. Deng, and Feng Bao, “A Practical Password-Based Two Server Authentication and Key Exchange System,” IEEE Transaction on Secure and Dependable Computing,Vol.3, No.2,April-June 2006

[4] Muxiang Zhang, Analysis of the SPEKE password-authenticated key exchange protocol, IEEE Communications Letters, Vol. 8, No. 1, pp. 63-65, January 2004.

[5] Z. Zhao, Z. Dong, Y. Wang, “Security analysis of a password-based authentication protocol proposed to IEEE 1363,” Theoretical Computer Science, Vol. 352, No. 1, pp. 280–287, 2006.

[6] J. Brainard, A. Juels, B. Kaliski, and M. Szydlo, “A New Two Server Approach for

Authentication with Short Secrets," Proc. USENIX Security Symp, 2003.

[7] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password Based Protocols Secure against Dictionary Attacks," Proc. IEEE Symp. Research in Security and Privacy, pp. 72-84, 1992.

[8] S. Bellovin and M. Merritt, "Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise," ACM Conf. Computer and Comm. Security, pp. 244-250, 1993.

[9] M. Bellare, D. Pointcheval, and P. Rogaway, Authenticated Key Exchange Secure Against Dictionary Attacks, Advances in Cryptology (Eurocrypt '00), pp. 139-155, 2000.

[10] M. Bellare and P. Rogaway, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, Proc. ACM Computer and Comm. Security, pp. 62-73, 1993.

[11] W. Ford and B.S. Kaliski Jr., Server-Assisted Generation of a Strong Secret from a Password, IEEE Ninth Int'l Workshop Enabling Technologies, 2000.

[12] D.P. Jablon, Password Authentication Using Multiple Servers, RSA Security Conf., pp. 344-360, 2001.

[13] P. Mackenzie, T. Shrimpton, and M. Jakobsson, "Threshold Password-Authenticated Key Exchange," Proc. Advances in Cryptology (Eurocrypt '02), pp. 385-400, 2002.

[14] Y.J. Yang, F. Bao, and R.H. Deng, "A New Architecture for Authentication and Key Exchange Using Password for Federated Enterprises", Proc 20th Int'l Federation for Information Processing Int'l Information Security Conf (SEC '05), 2005.

[15] F. Hao, P. Zieli´nski, A 2-round anonymous veto protocol, Proceedings of the 14th International Workshop on Security Protocols, SPW'06, Cambridge, UK, May 2006.

[16] Abdalla M., Catalano D., Chevalier C., and Pointcheval D., "Efficient Two-Party Password-Based Key Exchange Protocol in the UC Framework", Springer-Verlag Berlin, PP. 335 – 351, 2008.

AUTHOR BIOGRAPHY

[1]**T.S.Thangavel** received the Bsc degree in Computer Science (Bharathiyar University) in 1991 and the Msc degree in computer science(Bharathidasan University) in 1993 and the Mphil degree in Computer Science (Bharathidasan university) in 2003. He is pursuing the PhD degree in department of science and humanities (Anna university). He is working as an assistant professor in MSC(IT) department at K.S.Rangasamy College of Technology, Tiruchengode

[2]**Dr. A. Krishnan** received his Ph.D degree in Electrical Engineering from IIT, Kanpur. He is now working as an Academic Dean at K.S.Rangasamy College of Technology, Tiruchengode and research guide at Anna University Chennai. His research interest includes Control system, Digital Filters, Power Electronics, Digital Signal processing, Communication Networks. He has been published more than 250 technical papers at various National and International Conference and journals.