ISSN:2231-0711

Priyanka Chandragiri et al | IJCSET |February 2013 | Vol 3, Issue 2, 48-53

Clattering Recall Using Self-Encryption Scheme in a Distributed Data Storage System

Priyanka Chandragiri,
Dept of CSE
Christu Jyothi Institute of Technology & Sciences,
Warangal

A. Sowmya
Dept of IT
Christu Jyothi Institute of Technology & Sciences,
Warangal

***Abstract*-In this paper presents a novel data encryption and storage scheme to address this challenge. Treating the data as a binary bit stream, our self-encryption (SE) scheme generates a key stream by randomly extracting bits from the stream. The length of the key stream depends on the user's security requirements. The bit stream is encrypted and the cipher text is stored on the mobile device, whereas the key stream is stored separately. This makes it computationally not feasible to recover the original data stream from the cipher text alone. Our scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s). Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including: data update, delete and append. Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks. An algorithm is described which guarantees reliable storage of data in a distributed system, even when different portions of the data base, stored on separate machines, are updated as part of a single transaction. The algorithm is implemented by a hierarchy of rather simple abstractions, and it works properly regardless of crashes of the client or servers. Some care is taken to state precisely the assumptions about the physical components of the system (storage, processors and communication).In which crash recovery can be performed through self encryption scheme.**

Keywords: **Data security, stream cipher, wireless network**

I. INTRODUCTION

We consider the problem of crash recovery in a data storage system that is constructed from a number of independent computers. The portion of the system that is running on some individual computer may crash, and then be restarted by some crash recovery procedure. This may result in the loss of some information that was present just before the crash. The loss of this information may, in turn, lead to an inconsistent state for the information permanently stored in the system. For example, a client program may use this data storage system to store balances in an accounting system. Suppose that there are two accounts, called A and B, which contain $10 and $15 respectively. Further, suppose the client wishes to move $5 from A to B.

The client might proceed as follows:

read account A (obtaining $10)
read account B (obtaining $15)
write $5 to account A
write $20 to account B

Now consider a possible effect of a crash of the system program running on the machine to which these commands are addressed. The crash could occur after one of the write commands has been carried out, but before the other has been initiated. Moreover, recovery from the crash could result in never executing the other write command. In this case, account A is left containing $5 and account B with $15, an unintended result. The contents of the two accounts are inconsistent. There are other ways in which this problem can arise: accounts A and B are stored on two different machines and one of these machines crashes; or, the client itself crashes after issuing one write command and before issuing the other.
In this paper we present an algorithm for maintaining the consistency of a file system in the presence of these possible errors. We begin, in section 2, by describing the kind of system to which the algorithm is intended to apply. In section 3 we introduce the concept of an atomic transaction. We argue that if a system provides atomic transactions, and the client program uses them correctly, then the stored data will remain consistent. The remainder of the paper is devoted to describing an algorithm for obtaining atomic transactions. Any correctness argument for this (or any other) algorithm necessarily depends on a formal model of the physical components of the system. Such models are quite simple for correctly functioning devices. Since we are interested in recovering from malfunctions, however, our models must be more complex. Section 4 gives models for storage, processors, and communication, and discusses the meaning of a formal model for a physical device. Starting from this base, we build up the lattice of abstractions shown in figure 1. The second level of this lattice constructs better behaved devices from the physical ones, by eliminating storage failures and eliminating communication entirely. The data storage system is constructed from a number of computers; the basic service

provided by such a system is reading and writing of data bytes stored in the system and identified by integer addresses. There are a number of computers that contain client programs (clients), and a number of computers that contain the system (servers); for simplicity we assume that client and server machines are disjoint. Each server has one or more attached storage devices, such as magnetic disks. Some facility is provided for transmitting messages from one machine to another. A client will issue each read or write command as a message sent directly to the server storing the addressed data, so that transfers can proceed with as much concurrency as possible. For instance, due to constraints imposed by limited computing power, storage space, and battery lifetime, a light-weight rather than computing intensive and complex encryption algorithm is desired in Distributed data Storage system. It is very challenging to protect the weakly encrypted information, which might end up in the hands of an adversary, who could then use powerful cryptanalysis tools to break the encryption. Therefore, security solutions developed for general distributed data storage systems cannot be adopted directly for this new frontier. The most challenging part of mobile device data protection lies in the conflicting requirements for the data encryption scheme. While it should be computationally infeasible for adversaries to decrypt the data in captured mobile devices, the encryption/decryption operation should be reasonably efficient for legitimate users. Furthermore, the required computations should not consume too much energy so as to minimize battery drain.

This research proposes a novel stream cipher scheme called self-encryption (SE) to address this dilemma. Treating the data set as a binary bit stream, we generate the key stream by extracting n bits in a pseudorandom manner based on a user's unique PIN and a nonce. The length of the key stream n is flexible and depends on the security requirements. Then we encrypt the remaining bit stream using this key stream. The encrypted remainder is stored in the mobile device, whereas the key stream is stored separately. It is very difficult to recover the original data stream from the cipher text even if an adversary has the knowledge of the encryption algorithm. The variable length key stream makes brute force attacks infeasible, and the decrypted data stream is still unrecognizable unless the key stream bits are inserted to the original position.

II. RELATED WORK

Securing sensitive and/or private data in mobile communication has been an important topic in security research community. Our research is relative to two main areas: modern stream cipher design and distributed data security.

A. Modern Stream Cipher Design

Stream ciphers are widely used to protect sensitive data at fast speeds. Although block ciphers have been attracting more and more attention, stream ciphers still are very important, particularly for military applications and to the academic research community. Stream ciphers are more suitable in environments where tight resource constraints are applied, i.e. in wireless mobile devices or wireless sensor networks. When there is a need to encrypt large amount of streaming data, a stream cipher is preferred. In recent years, a lot of efforts have been reported in this area and many interesting new stream ciphers have been proposed and analyzed. A popular trend in stream cipher design is to turn to block-wise stream ciphers like RC4, SNOW 2.0, and SCREAM. In order to improve the time-data-memory tradeoff for stream cipher, a concept of Hellman's time-memory tradeoff has been applied and it achieved obvious improvements.

The Goldreich Levin one-way function hard-core bit construction has been enhanced into a more efficient pseudo-random number generator BMGL with a proof of security. Efficient hardware implementations of stream ciphers are important in both high-performance and low-power applications. This is the main trend of the stream cipher development in the future. Researchers have pointed out that RF1D (Radio Frequency Identification) could be one of the next killer applications for hardware-oriented stream ciphers. The second phase of the eSTREAM project in particular focused stream ciphers suited toward hardware implementation and currently there are eight families of hardware-oriented stream ciphers.

Normally there are two input parameters to a stream cipher, the password and an initialization vector (IV). In contrast with the user password being kept secret, the IV is public. As a consequence, attacks against the IV setup of stream cipher have been very successful. Due to the weakness with the IV setup, more than 25% of the stream ciphers submitted to the eSTREAM project in May 2005 have been broken. Some seemingly robust academic designs were broken also due to problems with the IV setup.

In this paper, we will investigate an alternative design approach for the self-encryption stream cipher scheme to avoid the shortcomings incurred by using public IV. Also, the robustness of a fixed length key stream has been weakened as the computing power which an adversary possesses has been growing. Instead, a variant length key stream will make brute force attacks computationally infeasible. To reach this goal, this paper will also introduce a novel key stream generation scheme.

B. Distributed Data Security

Effective data protection solution is one of the essential security requirements that affect the acceptance of next generation pervasive computing and the mobile device utilization in enterprise networks. The rapid increase of sensitive data and the growing number of government regulations require long-term data retention to storage security. During the data's life cycle, there are a lot of potential attack points. In past decades, many researchers have contributed in this area. Among the reported works, here we skip the great achievements in network file system since they are not very relative to the proposed project. Instead, we will briefly introduce the recent progress in security services for distributed data storage protection. Data should be protected during the whole life cycle. Authentication and authorization are the preliminary requirements in most data security

systems. In general, authentication can be implemented using techniques such as passwords, digital signatures, or MAC (Message Authentication Code). Authorization can be performed by certificates, access control, etc. Considering the risks of system crash or denial-of-service, availability is required in most commercial systems. Typical solution is to make duplicated backup. However, replication increases the cost of consistency maintenance.

The essential task of data security is to prevent any unauthorized third party from revealing or modifying the data. Confidentiality can be achieved by using encryption, while data integrity can be achieved by using digital signatures and/or MAC. During transmit the data can be protected by using protocols such as SSL and IPSec. Meanwhile, at the storage the data confidentiality can be achieved using user encryption schemes. Variant cipher schemes are proposed for this purpose including the new designs we mentioned in previous section, Estream project. To be robust against cryptanalysis, the key sharing and key management are also critical part in the context. Special care has to be taken while storing, archiving, and deleting key materials. Another important research area is the key recovery system, which helps the users to decrypt the cipher text under certain conditions.

Considering the constraints in mobile devices and the asymmetric power available to the adversary, there is no existing solution can be adopted directly to address the data security question in mobile devices. The proposed project is to investigate more robust stream cipher scheme by exploring more flexible key stream generation methods and more secure key stream management approach. The detailed discussion of our proposed research works is presented in the next section.

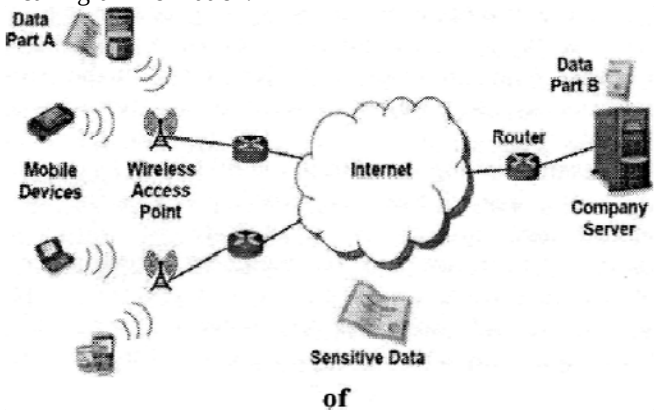
III. SELF-ENCRYPTION SCHEME

This section consists of two parts. First of all, we will introduce a framework under which the sensitive and/or private data are separated and stored in a distributed manner. Secondly, we will specify the detailed design of our SE scheme.

A. Framework of SE Scheme

Considering the fact that generally mobile devices do not possess as many resources as normal computers, it is very challenging to prevent an adversary from breaking the embedded cryptographic algorithm when the mobile devices are captured. It is also not desirable to implement a complex computing intensive encryption/decryption scheme in a Distributed system. Therefore, the rationale of this project is to investigate a novel light-weight approach to protect the information effectively even if an adversary has good knowledge of the encryption algorithm and many more resources to break the cryptography.

To reach this goal, our essential idea is that an adversary can only obtain part of the data from the distributed system alone, which is not enough to reveal any useful information. As illustrated by a scenario shown in Figure 1, the sensitive data is broken into two parts using our self-encryption stream cipher scheme. The major part (Part A: cipher text) is stored in the mobile device carried by the company employee, and the minor part (Part B: key stream + other parameters) is protected in the secure server of the company. Part A is encrypted using part B. When the user needs to access the data, he or she has to input a correct PIN to pass the authentication procedure. Then the server will send part B to decrypt part A and merge them together to recover the original plaintext. When a data is lost, at most the adversary can access the part A, from which it is computationally infeasible to get meaningful information.



of

Fig.1 Overview of the Self Encryption Framework

B. Self-Encryption Scheme

Similar as general stream cipher, the proposed SE stream cipher also encrypts the plaintext and decrypts the cipher text by adding bitwise a key stream:

Cipher text = Plaintext © Key stream (1)

The key stream generator consists of two parts, a hash function H and a random number generator G. The hash function takes the user's PIN and a nonce as input and the output is an integer seed, which is used as the seed of the random number generator G. The output random number sequence {ro, n, ..., rn-i} indicates which bits are selected and abstracted from the message (plaintext) to form the key stream. Therefore, we have:

seed = H(PIN, nonce) (2)

{ro, n, ..., rn-i} = G(seed) (3)

where {ro, n, ... rn-i} is a random number sequence generated by the random number generator G. Since the random numbers could beyond the length of the message, and the length of the message body decreases as bits are abstracted, the pointers to the key stream bits need to be normalized following the changing message size. Hence, among the n abstracted bits {r'o, r'i, ... r'n-i}, the position of the /c-th bit is:

r'k= a mod (m-/() (4)

The length of the random number sequence 77, which is also the length of key stream, is determined by the size (number of bits) of the message m and the security requirement. A longer the key stream provides more robust cipher to protect a larger size message. To support this flexibility, we define parameter security level sl as the security level and A as the minimum

length unit difference between two consecutive security levels. A is a percentage instead of a fixed bit number. This design leads to a unique length of each key stream depending on the concrete message size. It makes the brute force attacks much difficult as the working load for key stream guess is increased exponentially. The key stream length n is calculated as:

n=m* Si *A if Si is not zero (5)

To illustrate the use of equation (5), assume A = 5%, for example, then the length of the keystream can be 5% of the original message size when sl = 1, 10% when sl = 2, 75% when sl = 3, and so on. When sl = 0, a default fixed key stream length is adopted, where n = 256 bits. Actually, further experimental and theoretical analysis will be conducted to set the optimal value of A. The use of security level sl should be specified in more detail in the design of the SE protocol in the future.

Figure 2 presents the working flow of the proposed SE stream cipher. When the user has finished editing or reading the document, the following works are performed. The seed of the random number generator is calculated by the hash function taking the user's PIN and a nonce as the input. Then, according to the size of the sensitive document and the security level, a sequence of random numbers is generated with length n. By treating the file as a binary stream, this random number sequence indicates which bits in the data file are abstracted to form the key stream.

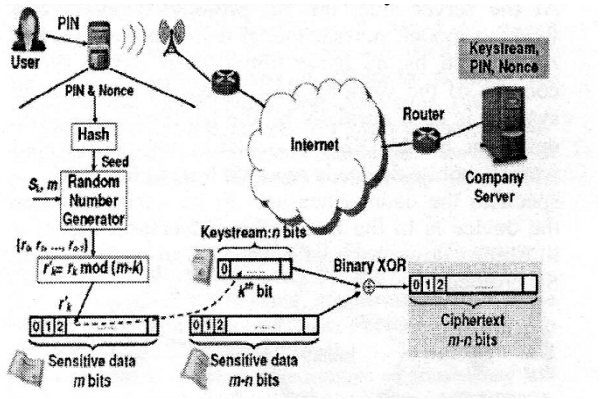


Fig.2 SE Scheme Working Flow Illustration.

Then the cipher text is calculated as normal stream cipher does. The cipher text is stored in the mobile device, the key stream, user's PIN, and the nonce are stored the secure server in the company. We will investigate the tradeoff between the performance and security regarding the information to be transferred back to the server. For instance, maybe it is more secure not to transfer the user's PIN and nonce, instead, backing up the sequence {r'o, r'l, ...r'n-i} is better.

Comparing to existing stream cipher schemes, computationally the proposed SE scheme is much more robust. The length of the keystream is not fixed except when the default value (256) is adopted, if the user selected security level sl = 0. This raises the bar of brute force attackers, the complexity is increased to O(2m).

Furthermore, to recover the original data stream, the adversary needs to insert every bit of the key stream back correctly. The permutation in this operation is:

Pm = m x(m-1) x(m-2) x ..x(m-n+1) (6)

The complexity of this part is O(mn). Then the total complexity is O(2m/r?n), which is much robust than the reported modern stream cipher schemes.

IV. SE PROTOCOL DESIGN

To secure the sensitive data in mobile devices, a protocol set is mandatory to support the functionalities of the SE stream cipher, the AD agent, and the server. In addition, the protocol specifies the behavior of the whole system. At the distributed data storage side, the major functions include:

Setting up connection with the remote server;

2) Retrieving the key stream and nonce for local decryption;

3) Generating a new key stream with a new nonce and encrypting the document; and

4) Transferring the updated key stream and new nonce back to server.

At the server side, the SE protocol supports two working model: normal model and emergent model. As implied by its name, the normal model (NM) consists of the working flow when the data storage system is used normally by the legitimate user. The emergent model (EM) is a status that is triggered when a mobile device is reported lost. In fact, EM specifies the countermeasures to be executed when the device is in the hand of an adversary. Figure 3 illustrates flow charts of both sides in our proposed SE protocol.

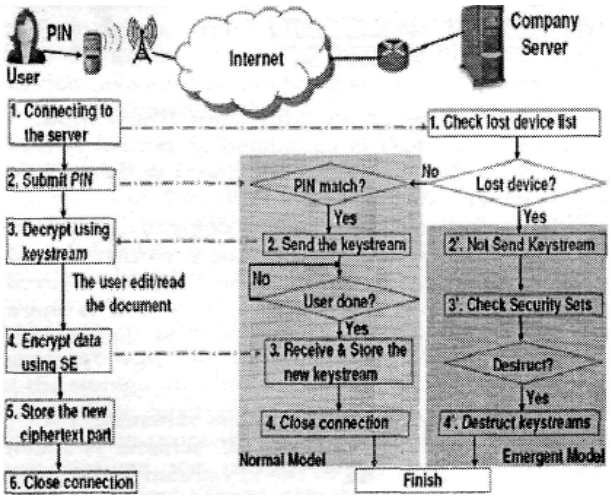


Figure 3. SE Protocol Working Flow Chart.

When a data storage system is turned on and trying to setup connection with the server through the network, the first action the server takes is to check whether this mobile device is reported lost. For this purpose, the server maintains a list of reported lost devices. When the mobile device is not in the lost list, the server continues working in the normal model.

As presented along the path in the middle of Figure 3, the server checks the user's PIN, provides the key stream and nonce to mobile device allowing legitimate user edit/read the document. When a user finishes her work, a new key stream and nonce are sent back and stored in the server. During this procedure, if the input PIN error happens three times, the server will suspend the account but won't enter the emergent model. In contrast, if the device matches a record in the lost list, the server enters the emergent model. It will ignore the received PIN and automatically reject the requirement of key stream materials. The further activities depend on the user's security setting, If the user has explicitly required, the server will destruct the decryption materials permanently.

V. CONCLUSIONS AND DISCUSSIONS

Lack of effective protection of sensitive data in mobile devices is a major concern that prevents the mobile devices from being used widely as part of enterprise networks or personal area networks. The proposed SE system will remove the barrier and enable employees to enjoy the high efficiency and convenience brought by mobile devices. It will lead to another wave of prosperity of wireless networks and pervasive computing. Physical attacks have been proved effective in breaking some well designed ciphers in practice [13]. Unfortunately, it is challenging to designers to theoretically investigate the robustness of a cipher scheme against various physical attacks. To address this problem, a prototype is going to be implemented on top of reconfigurable hardware devices (i.e. FPGAs). Particularly we will study the behavior of our SE prototype under local non-invasive attacks including timing analysis and differential power analysis (DPA) [20].

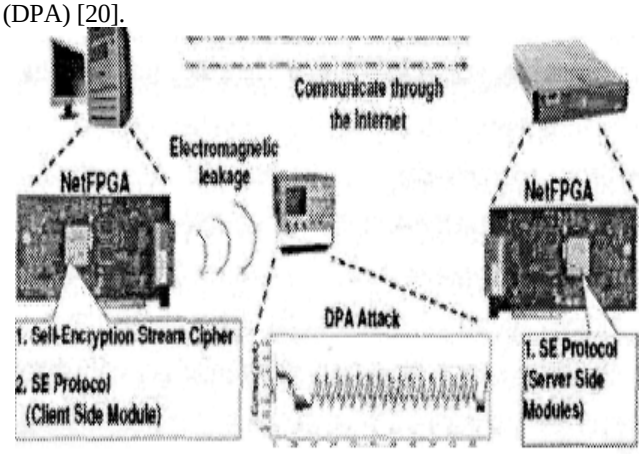


Figure 4. Prototype Implementation & Experiment Platform Construction.

Figure 4 presents the prototype implementation and physical attack study system architecture. At the server side, we plan to implement the SE protocol on a NetFPGA board inserted in a Dell 2950 server. As the mobile device side, we are considering to implement the SE stream cipher scheme and SE protocol on another NetFPGA board inserted in a PC, which is connected to the network through wireless connection.

Devices such as oscillo graph will be used to monitor and record the electromagnetic leakage when the SE stream cipher is being executed to encrypt/decrypt the data. As shown in middle of Fig. 4, an adversary may perform DPA attacks by analyzing the variance of leaking electromagnetic wave. Actually, we expect that our SE stream is not vulnerable to DPA attacks due to the uniqueness of each key stream and a much larger key stream space. However, we are also prepared to improve the implementation if vulnerabilities are observed on the prototype.

Aside from investigating the potential security vulnerability, we will study the performance issues using the prototype in the context of real applications. Considering the resource constraints in the typical mobile devices, the proposed SE stream cipher is hardware-oriented and aims at light-weighted design. We will explore the tradeoffs between the performance and resource utility by the SE system. We have defined a facility (transactions) which clients can use to perform complex updates to distributed data in a manner that maintains consistency in the presence of system crashes and concurrency. Our algorithm for implementing transactions requires only a small amount of communication among servers. This communication is proportional to the number of servers involved in a transaction, rather than the size of the update. We have described the algorithm through a series of abstractions, together with informal correctness arguments.

VI. REFERENCES

[1] J. Al-Muhtadi, D. Mickunas, and R. Campbell, "A Lightweight Reconfigurable Security Mechanism for 3G/4G Mobile Devices," *IEEE Wireless Communications*, April 2002.

[2] D. J. Bernstein, "Which eSTREAM ciphers have been broken?" http://www.ecrypt.eu.org/ stream/, submitted 2008-02-21.

[3] A. Biryukov, "Block Ciphers and Stream Ciphers: The State of the Art," *Lecture Notes in Computer Science, in Proceedings of the COSIC Summer course*, 2003.

[4] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Proceedings of Asiacrypt'OO*, no. 1976 in Lecture Notes in Computer Science, pp. 1-13, Springer-Verlag, 2000.

[5] W. Daniel, T. Pintaric, F. Ledermann, S. Dieter, "Towards Massively Multi-User Augmented Reality on Handheld Devices", *International Conference on Pervasive Computing, Munich*, Germany, 2005.

[6] D, E. Denning and D. K. Branstad, "A Taxonomy for Key Escrow Systems," *Communications of the ACM*, Vol. 39, Issue 3, 1996.

[7] eSTREAM, ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream. *Notes in Computer Science*, pp. 239-255, edited by W. Fumy, Springer-Verlag, 1997.

[8] T. Good and M. Benaissa, "Hardware performance of eStream phase-Ill stream cipher candidates," *the State of the Art of Stream Ciphers Workshop*- (SASC'08), Lausanne, Switzerland, Feb. 13-14, 2008.

[9] K. Greene, "Securing Cell Phones," *Technology Review*, MIT, Wednesday, Aug. 01, 2007.

[10] J. Hastad and M. Naslund, "Improved analysis of the BMGL keystream generator," in *Proceedings of the Second NESSIE Workshop*, 2001.

[11] Eswaren, K. P. et al. The notions of consistency and predicate locks in a database system. Comm. ACM 19, 11. 624-633, (Nov 1976).

[12] Gifford, D.K. Violet: An experimental decentralized system. Submitted to 7th Symposium on Operating System Principles, 1979.

[13] Gray, J.N. Notes on data base operating systems. In Operating Systems, An Advanced Course, American Elsevier, 1978.

[14] N. Fournel, M. Minier, and S. Ubeda, "Survey and Benchmark of Stream Ciphers for Wireless Sensor Networks," the *Workshop in Information*

Security Theory and Practices (WISTP'07), Crete, Greece, May 8-11, 2007.

[15] A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL Protocol, Version 3.0," *Internet draft.* Networking Group, March 1996.

.

AUTHORS

Priyanka Chandragiri had received Master of Technology(software engineering) from Vageedevi Engineering College, affiliated to JNTU, Bollikunta, Warangal. Currently working at Christu Jyothi Institute of Technology & sciences, Jangaon, Warangal,A.P, as an Assistant Professor in department of CSE. Her Research area includes wireless and Sensor Networks..

A.Soumya working as an Sr. Assistant Professor in department of IT, Christu Jyothi Institute of Technology &sciences, Jangaon, Warangal, A.P, received Master of technology(software Engineering)from Jayamukhi Institute of Technological Sciences, affiliated to JNTU, Narsampet. Her Research area includes wireless and Sensor Networks