Virtual Energy-Efficient Encryption and Keying (VEEEK) for Wireless Sensor Networks

K. Ravi Chythanya,
M. Tech (Computer Science and Engineering) Student,
S. R. Engineering College,
Warangal (A.P), India,
chythu536@gmail.com.

S.P.Anandaraj.,M.Tech(HON's),(Ph.D).,
Sr.Assistant Professor in CSE Dept,
S. R. Engineering College,
Warangal (A.P), India,
anandsofttech@gmail.com.

S. Padmaja,
Assistant Professor, Computer Science and Engineering,
Sree Chaitanya College of Engineering,
Karimanagar (A.P), India,
mahepadamaja@gmail.com.

***Abstract*—The sensor nodes are resource-limited wireless devices; therefore the designing of protocols for WSNs (Wireless Sensor Networks) which are cost-efficient and secure is a major challenging problem in today's networking and mobile computing area. The communication cost is the most important factor in the sensor's energy consumption; here we are introducing a virtual energy-efficient encryption scheme for WSNs which reduces the transmissions between nodes which are required for rekeying to avoid stale keys. The main goal is to save energy in WSNs besides this we have minimal transmission is also imperative in some military applications of WSNs in which the adversary could be monitoring the wireless spectrum. This energy-efficient encryption is a secure communication framework in which an algorithm is used to encode the sensed data. This algorithm is worked based on a permutation codes generated via RC5 Encryption mechanism. The key to RC5 algorithm encrypts blocks of plaintext of length 32 bits into blocks of cipher text of the same length. And the key length ranges from 0 to 2040 bits. Thus, the key will be changed for every packet and different keys are used for successive packets. In this paper, we used the above algorithm to send the messages secretly in military applications while the wireless spectrum is adversely monitoring by the enemies.**

Keywords: Wireless Sensor Networks, Security, WSN security, wireless spectrum.

1. Introduction

The wireless sensor networks are playing a key role in variety of application scenarios. The typical application scenarios include environmental, military, and commercial enterprises [2]. In another aspect, the under water sensor nodes are very useful in oceanographic data collection, pollution monitoring, assisted navigation, military surveillance, and mine reconnaissance operations. The improvements in technology will give more sensor applications in our daily lives [3]. From the security aspect, the authentication and accurate data surrounding sensor nodes must be provided and to the sink to trigger time critical responses such as troop movement, evacuation, and first response deployment [4]. Resilient protocols must be used against the false data which are injected into the sensor networks by the malicious nodes. Otherwise, the cost for propagating false data or redundant data is very high. The protocol builders must be cautious about the limited resource utilization onboard the sensors more efficiently [5].

In this paper, we focus on the keying mechanisms for Wireless Sensor Networks. There are two types of keying mechanisms for WSNs: *static* and *dynamic*. *In static key management*, the sensors are loaded with a

fixed number of keys when the deployment of network is completed. In this key management schemes, key management functions (i.e., key generation and distribution) are handled statically.

In the dynamic key management, schemes perform keying functions (rekeying) either periodically or on demand as needed by the network. The sensors dynamically exchange the keys to communicate. The dynamic schemes are more attack resilient than the static ones, one significant disadvantage is the communication overhead will be increased because of keys which are being refreshed or redistributed from time to time in the sensor networks. The overhead associated with refreshing keys to avoid them becoming stale is to be minimized. Because from a sensor's energy consumption point of view, the communication cost and the message transmission costs are very important issues in a Wireless Sensor network deployment [6]. That is, the less *chatty* option gives fewer chances to the malicious entities to eavesdrop or intercept packets [1].

The purpose of this paper is to develop an efficient and secure communication framework for WSN applications**.** VEEEK's secure communication framework provides a technique to verify data in line and drop false packets from malicious nodes, thus maintaining the health of the sensor network. VEEEK dynamically updates keys without exchanging messages for key renewals and embeds integrity into packets as opposed to enlarging the packet by appending message authentication codes (MACs). Each sensed data is protected using RC5 encryption scheme. The key to the encryption scheme dynamically changes as a function of the residual virtual energy of the sensor, thus requiring no need for rekeying. Therefore, a one-time dynamic key is used for one message generated by the source sensor and different keys are used for the successive packets of the stream. The nodes forwarding the data along the path to the sink are able to verify the authenticity and integrity of the data and to provide non-repudiation. The protocol is able to continue its operations in a high-error-prone deployment area like under water. The contributions of this paper are as follows:

1. A dynamic route filtering mechanism that does not exchange explicit control messages for re-keying;

2. Provision of one-time keys for each packet transmitted to avoid stale keys;

3. A modular and flexible security architecture with a simple technique for ensuring authenticity, integrity, and non-repudiation of data without enlarging packets with MACs; and

4. A robust secure communication framework that is operational in dire communication situations and over unreliable medium access control layers.

2. Background and Motivation

One significant aspect of confidentiality research in WSNs entails designing efficient key management schemes. This is because regardless of the encryption mechanism chosen for WSNs, the keys must be made available to the communicating nodes (e.g., sources and sink(s)). The keys could be distributed to the sensors before the network deployment or they could be redistributed (rekeying) to nodes on demand as triggered by keying events. The former is static key [8] management and the latter is dynamic key [9] management. There are myriads of variations of these basic schemes in the literature. In this work, we only consider dynamic keying mechanisms in our analysis since VEEEK uses the dynamic keying paradigm. The main motivation behind VEEEK is that the communication cost is the most dominant factor in a sensor's energy consumption [5], [6]. Thus, in this section, we present a simple analysis for the rekeying cost with and without the transmission of explicit control messages.

Dynamic keying schemes go through the phase of rekeying either periodically or on demand as needed by the network to refresh the security of the system. With rekeying, the sensors dynamically exchange keys that are used for securing the communication. Hence, the energy cost function for the keying process from a source sensor to the sink while sending a message on a particular path with dynamic key-based schemes can be written as follows (assuming computation cost, Ecomp, would approximately be fixed):

$$E_{Dyn} = \left(E_{K_{disc}} + E_{comp}\right) * E[\eta_h] * \frac{\chi}{\tau}, \qquad (1)$$

Where χ is the number of packets in a message, τ is the key refresh rate in packets per key, E_{Kdisc} is the cost of shared key discovery with the next hop sensor after initial deployment, and $E[\eta_h]$ is the expected number of hops. In the dynamic key-based schemes, τ may change periodically, on demand, or after a node-compromise. A good analytical lower bound for $E[\eta_h]$ is given as

$$E[\eta_h] = \frac{D - t_r}{E[d_h]} + 1, \qquad (2)$$

where D is the end-to-end distance (m) between the sink and the source sensor node, tr is the approximated transmission range (m), and E[d_h] is the expected hop distance (m). Finally, EKdisc, can be written as follows:

$$E_{Kdisc} = \{E[N_e] * E_{node}) * M - 2 * E_{node}\}, \quad (3)$$

$$E_{node} = E_{tx} + E_{rx} + E_{comp}, \quad (4)$$

3. Semantics of VEEEK

The VEEEK framework is comprised of three modules: Virtual Energy-Based Keying, Crypto, and Forwarding. The virtual energy-based keying process involves the creation of dynamic keys. Contrary to other dynamic keying schemes, it does not exchange extra messages to establish keys. A sensor node computes keys based on its residual virtual energy of the sensor. The key is then fed into the crypto module. The crypto module in VEEEK employs a simple encoding process, which is essentially the process of permutation of the bits in the packet according to the dynamically created permutation code generated via RC5. The encoding is a simple encryption mechanism adopted for VEEEK. However, VEEEK's flexible architecture allows for adoption of stronger encryption mechanisms in lieu of encoding.

Last, the forwarding module handles the process of sending or receiving of encoded packets along the path to the sink. A high-level view of the VEEEK framework and its underlying modules are shown in Fig. 1. These modules are explained in further detail below.

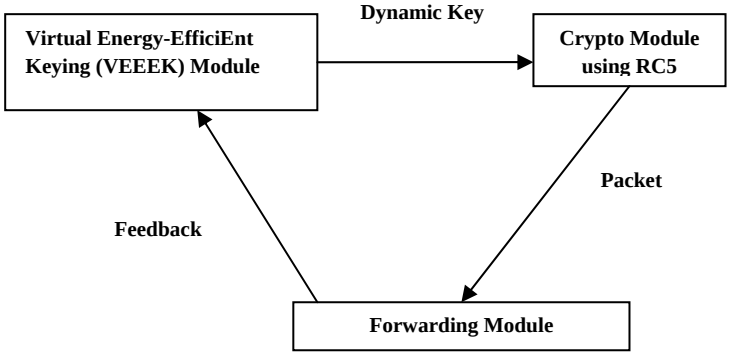

Fig 1: Modular Structure of VEEEK

3.1 Virtual Energy-Efficient Keying Module

The virtual energy-based keying module of the VEEEK framework is one of the primary contributions of this paper. It is essentially the method used for handling the keying process. It produces a dynamic key that is then fed into the crypto module. The current value of the virtual energy, E_{vc}, in the node is used as the key to the key generation function F. During the initial deployment, each sensor node will have the same energy level E_{ini}, therefore, the initial key, K_1, is a function of the initial virtual energy value and an Initialization Vector (IV). The IV s are pre-distributed to the sensors. Subsequent keys, K_j, are a function of the current virtual energy, E_{vc}, and the previous key K_{j-1}. VEEEK's virtual energy-based keying module generates a new unique key as the data is sensed and ensures that each packet is associated with a new key generated. Generated dynamic key is passed to the crypto module. Each node computes and updates the transient value of its virtual energy after performing some actions. Each action (or state traversal) on a node is associated with a certain predetermined cost. The set of actions and their associated energies for VEEEK includes packet reception (E_{rx}), packet transmission (E_{tx}), packet encoding (E_{enc}), packet decoding (E_{dec}) energies, and the energy required to keep a node alive in the idle state (E_a), transient value of the virtual energy, E_v, is computed by decrementing the total of these predefined associated costs, E_v, from the previous virtual energy value.

In order to successfully decode and authenticate a packet, a receiving node must keep track of the energy of the sending node to derive the key needed for decoding. In VEEEK, the operation of tracking the energy of the sending node at the receiver is called watching and the energy value that is associated with the watched sensor is called Virtual Perceived Energy (Ep) as in [7]. Eg. node A starts with the value of 2,000mJ as the first key to encode the packet (key generation based on the virtual energies is explained in the crypto module). Node A sends the first packet and decrements its virtual energy to 1,998 mJ. After node B receives this first packet, it uses the virtual perceived energy value (E_p ¼ 2,000 mJ) as the key to decode the packet, and its E_p (1,998 mJ) after sending the packet.

Algorithm 1. Compute Dynamic Key

1: ComputeDynamicKey(E_{vc}, ID_{clr})
2: **begin**
3: $j \leftarrow tx_{cnt}^{ID_{clr}}$
4: **if** $j = 1$ **then**
5: $K_j \leftarrow F(E_{ini}, IV)$
6: **else**
7: $K_j \leftarrow F(K_{(j-1)}, E_{vc})$
8: **end if**
9: **return** K_j
10: **end**

3.2. Crypto module:

The module performs simple encoding operation using the permutation of bits in the packet, according to the dynamically created permutation code via the RC5 encryption mechanism. The key to RC5 is created by virtual energy-based keying module. The purpose of the crypto module is to provide simple confidentiality of the packet header and payload while ensuring the authenticity and integrity of sensed data without incurring transmission overhead of traditional schemes.

The packets in VEEEK consists of the ID (i-bits), type (t-bits) (assuming each node has a type identifier), and data (d-bits) fields. Each node sends these to its next hop. RC5 encryption algorithm takes the key and the packet fields (byte-by-byte) as inputs and produces the result as a permutation code. The concatenation of each 8-bit output becomes the resultant permutation code. Thus, instead of the traditional approach of sending the hash value (e.g., message digests and message authentication codes) along with the information to be sent, we use the result of the permutation code value locally. To ensure correctness, the receiver compares the plaintext ID with the decoded ID. The benefits of this simple encoding scheme are:

1) Since there is no hash code or message digest to transmit, the packet size does not grow, avoiding bandwidth overhead on an already resource-constrained network, thus increasing the network lifetime.

2) The technique is simple and ideal for devices with limited resources (e.g., PDAs).

3) The input to the RC5 encryption mechanism, namely, the key, changes dynamically without sending control messages to rekey.

3.3 Forwarding Module

The final module in the VEEEK communication architecture is the forwarding module. It is responsible for the sending of packets (reports) initiated at the current node (source node) or received packets from other sensors (forwarding nodes) along the path to the sink. The operations of the forwarding module are explained in this section

3.3.1 Source Node Algorithm

When an event is detected by a source node, the next step is for the report to be secured. The source node uses the local virtual energy value and an IV (or previous key value if not the first) to construct the next key. For this dynamic key generation process is primarily handled by the VEEEK module. The source sensor fetches the current value of the virtual energy from the VEEEK module which is used as input into the RC5 algorithm inside the crypto module to create a permutation code for encoding the <ID|type|data> message. The encoded message and the clear text ID of the originating node are transmitted to the next hop (forwarding node or sink) using the following format: [ID; {ID; type; data} $_{Pc}$], where $[x]_{Pc}$ constitutes encoding x with permutation code Pc. The local virtual energy value is updated and stored for use with the transmission of the next report. Sensor fetches the current value of the virtual energy from

3.3.2 Forwarder Node Algorithm

Once the forwarding node receives the packet it will first check its watch-list to determine if the packet came from a node it is watching. If the node is not being watched by the current node, the packet is forwarded without modification or authentication. Although this node performed actions on the packet (received and forwarded the packet), its local virtual perceived energy value is not updated. If the node is being watched by the current node, the forwarding node checks the associated current virtual energy record (Algorithm2) stored for the sending node and extracts the energy value to derive the key. It then authenticates the message by decoding the message and comparing the plaintext node ID with the encoded node ID. If the packet is authentic, an updated virtual energy value is stored in the record associated with the sending node. If the packet is not authentic it is discarded. Again, the virtual energy value associated with the current sending node is only updated if this node has performed encoding on the packet.

Algorithm 2. Forwarding Node Algorithm with Communication Error Handling

1: Forwarder($currentNode, WatchedNode, UpstreamNode$)
2: **begin**
3: $i \leftarrow currentNode; enc \leftarrow 0; WL_i \leftarrow WatchList$
4: $k \leftarrow WatchedNode; src \leftarrow 0; j \leftarrow 0$
5: $E_{rx_i}, \langle ID_{clr}, \{msg\}_K \rangle \leftarrow ReceivePacket()$
6: **if** $ID_{clr} \in WL_i$ **then**
7: **while** $(keyFound = 0) and (j <= thresHold)$ **do**
8: $E^k_{p_i} \leftarrow FetchVirtualEnergy(i, ID_{clr}, enc, src)$
9: $K \leftarrow ComputeDynamicKey(E^k_{p_i}, ID_{clr})$
10: $Pc \leftarrow RC4(K, ID_{clr})$
11: $E_{dec_i}, Msg_{ID} \leftarrow decode(Pc, \{msg\}_K)$
12: **if** $ID_{clr} = Msg_{ID}$ **then**
13: $keyFound \leftarrow true$
14: **else**
15: $j++$
16: $E^k_{p_i} \leftarrow E^k_{p_i} - E_{tx_i} - E_{enc_i} - E_{rx_i} - E_{dec_i} - 2 * E_{a_i}$
17: **end if**
18: **end while**
19: **if** $keyFound = true$ **then**
20: **if** $j > 1$ **then**

```
21:          reEncode ← true
22:       else
23:          if E_{b_i} > 0 then
24:             reEncode ← true
25:          else
26:             reEncode ← false
27:          end if
28:       end if
29:       if reEncode = true then
30:          enc ← 1
31:          E_{b_i} ← FetchVirtualEnergy(i, ID_clr, enc, src)
32:          K ← ComputeDynamicKey(E_{b_i}, ID_clr)
33:          Pc ← RC4(K, ID_clr)
34:          E_{enc_i}, {msg}_Pc ← encode(Pc, msg)
35:          packet ← ⟨ID_clr, {msg}_Pc⟩
36:          E_{tx_i} ← ForwardPacket()
37:          E_{b_i} ← E_{b_i} − E_{tx_i} − E_{enc_i} − E_{rx_i} − E_{dec_i} − 2 * E_{a_i}
38:       else
39:          ForwardPacket() //Without any modification
40:       end if
41:    else
42:       DropPacket() //Packet not valid
43:    end if
44: else

45:    ForwardPacket() //Without any modification
46: end if
47: end
```

4. Operational Modes of VEEEK

The VEEEK protocol provides three security services Authentication, integrity, and non-repudiation. The fundamental notion behind providing these services is the watching mechanism. The watching mechanism requires nodes to store one or more records(i.e., current virtual energy level, virtual bridge energy values, and Node-Id) to be able to compute the dynamic keys used by the source sensor nodes, to decode packets and to catch erroneous packets either due to communication problems or potential attacks. However, there are costs (communication, computation, and storage) associated with providing these services.

PHASE-I:

In the first phase operational mode, all nodes watch their neighbors; whenever a packet is received from a neighbor sensor node, it is decoded using the and its authenticity and integrity are verified. Only legitimate packets are forwarded toward the sink. In this mode, there exists a short window of time, because it takes time for an attacker to capture a node or get keys. After the packet is decoded successfully, the plaintext ID is compared with the decoded ID. In this process, if the forwarding node is not able to extract the key successfully, it will decrement the predefined virtual energy value from the current perceived energy (line 16 in Algorithm 2) and tries another key before classifying the packet as malicious. If the packet is authentic, and this hop is not the final hop, the packet is re-encoded by the forwarding node with its own key derived from its current virtual bridge energy level. If the packet is illegitimate, the packet is discarded. This process continues until the packet reaches the sink.

Accordingly, illegitimate traffic is filtered before it enters the network. Re-encoding at every hop refreshes the strength of the encoding. The phase reduces the transmission overhead as it will be able to catch malicious packets in the next hop, but increases processing overhead because of the decode.

PHASE-II:

In the phase-II operational mode, nodes in the network are configured to only watch some of the nodes in the network. Each node randomly picks r nodes to monitor and stores the corresponding state before deployment. As a packet leaves the source node (originating node or forwarding node) it passes through node(s) that watch it probabilistically. If the current node is not watching the node that generated the packet, the packet is forwarded. If the node that generated the packet is being watched by the current node, the packet is decoded and the plaintext ID is compared with the decoded ID. Similar to phase-I, if the watcher-forwarder node cannot find the key successfully, it will try as many keys as the value of virtual Key Search Threshold before actually classifying the packet as malicious. If the packet is authentic, and this hop is not the final destination, the

original packet is forwarded unless the node is currently bridging the network. In the bridging case, the original packet is re-encoded with the virtual bridge energy and forwarded. This operational mode has more transmission overhead because packets from a malicious node may or may not be caught by a watcher node and they may reach the sink.

However, in contrast to the phase-I mode, it reduces the processing overhead (because less re-encoding is performed and decoding is not performed at every hop). The trade-off is that an illegitimate packet may traverse several hops before being dropped.

5. Performance Analysis

In this section, we evaluate the effectiveness of the VEEEK framework using analysis with Programming Language Java.

5.1 Assumptions

Due to the broadcast nature of the wireless medium used in sensor networks, attackers may try to eavesdrop, intercept, or inject false messages. In this paper, we mainly consider the false injection and eavesdropping of messages from an outside malicious node; hence, similar to [12], insider attacks are outside the scope of this paper. This attacker is thought to have the correct frequency, protocol, and possibly a spoofed valid node ID. Throughout this work, the following assumptions are also made:

- Directed Diffusion [7] routing protocol is used, but others such as [8] can also be used. According to specifics of Directed Diffusion, after the sink asks for data via interest messages, a routing path is established from the sources in the event region to the sink. We assume that the path is fixed during the delivery of the data and the route setup is secure.
- The routing algorithm is deployed on an unreliable medium access control protocol. The network may experience ACK or data packet drops.
- The sensor network is densely populated such that multiple sensors observe and generate reports for the same event.

5.2 Experimental Analysis

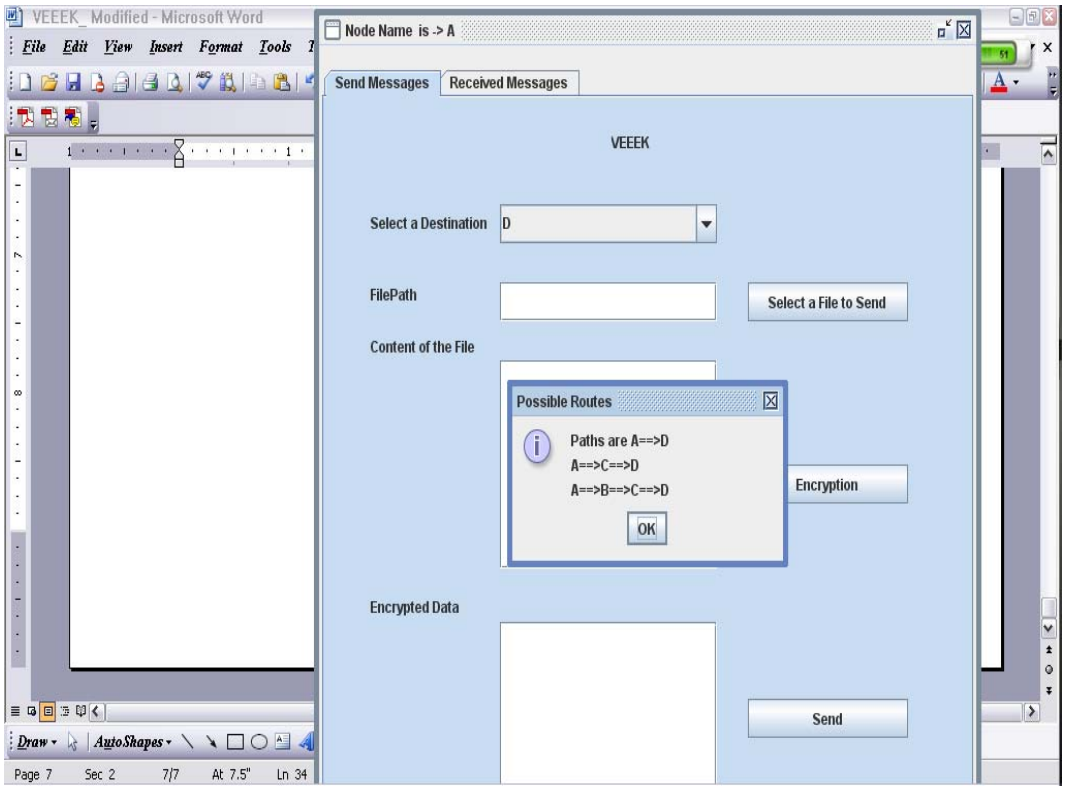

Fig 2: Possible Paths from Source A to Destination D

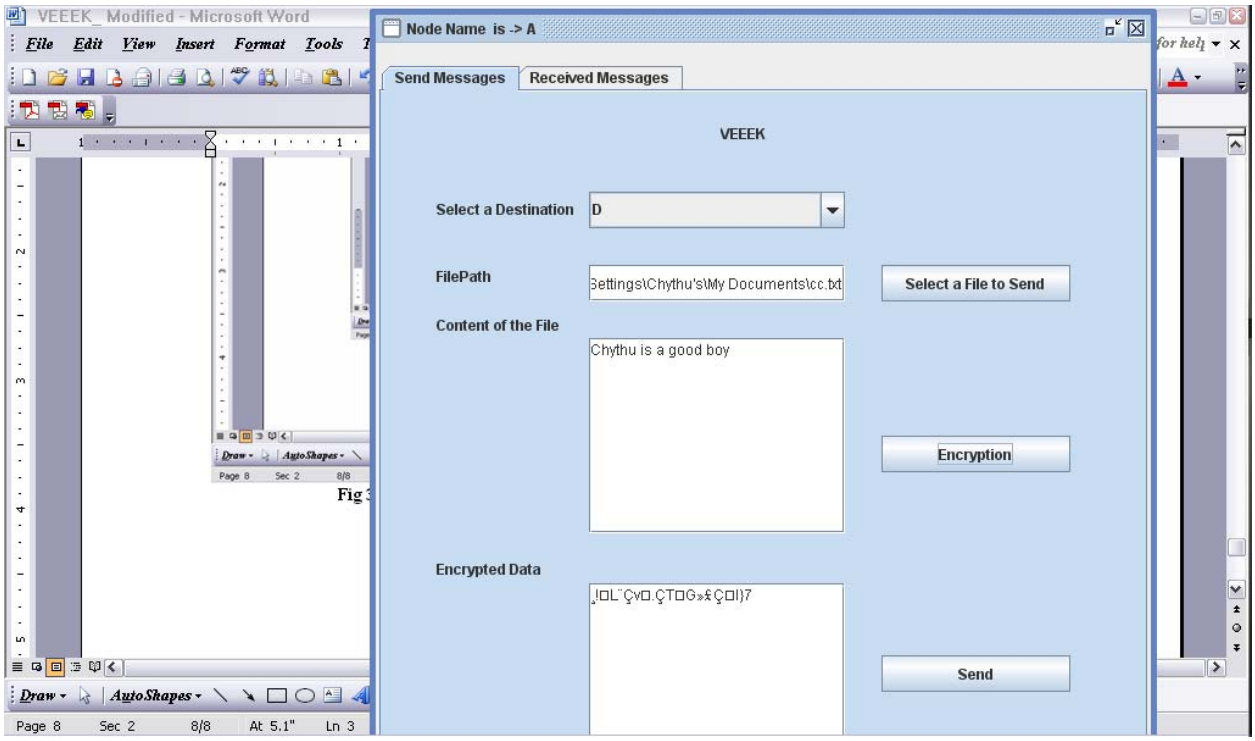

Fig 3: Showing Data after Encryption using RC5 with key value of 1800

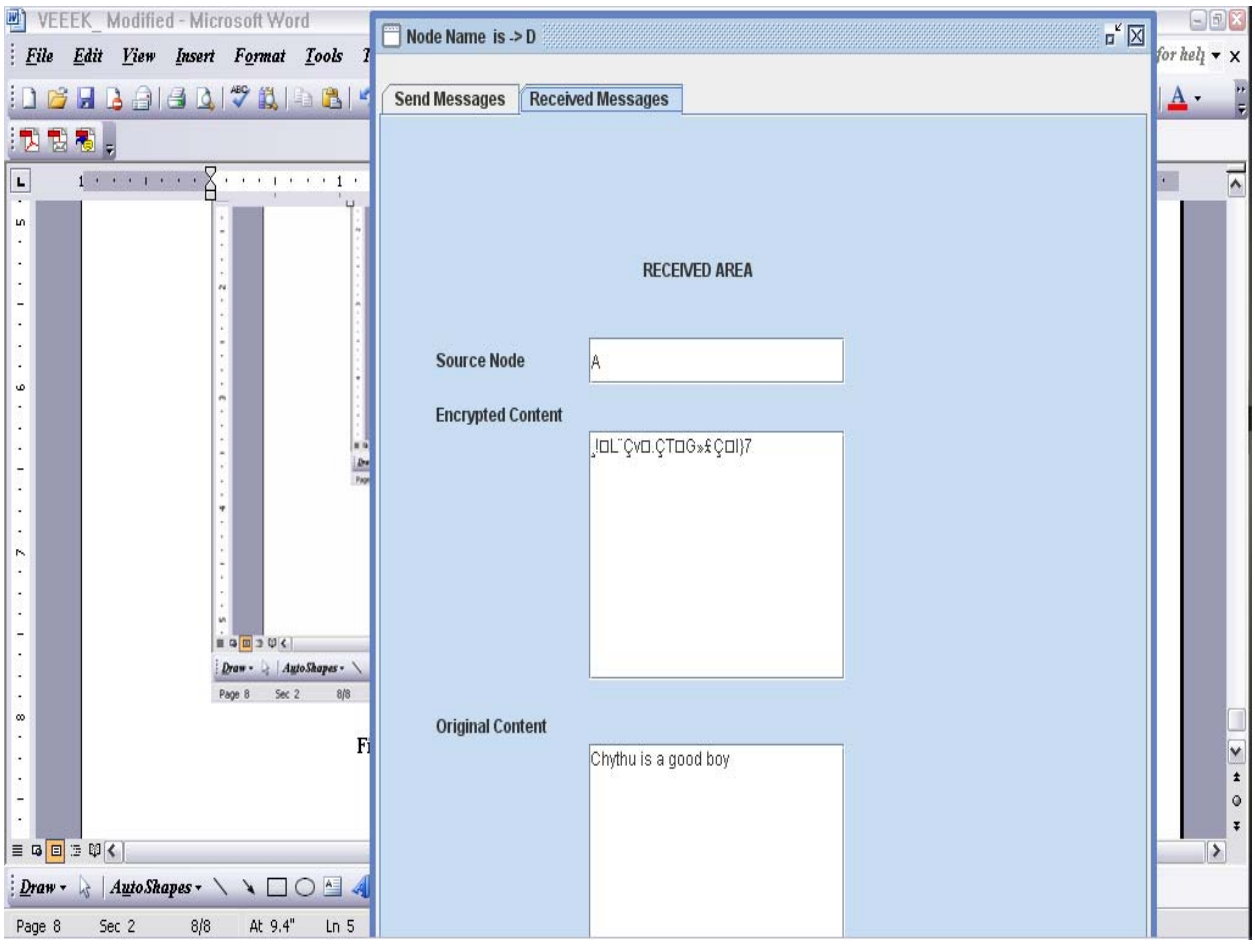

Fig 4: Showing Decrypted Data at Destination D.

6. Conclusion

Communication is very costly for wireless sensor networks (WSNs) and for certain WSN applications. Independent of the goal of saving energy, it may be very important to minimize the exchange of messages (e.g., military scenarios). To address these concerns, we presented a secure communication framework for WSNs called Virtual Energy-Efficient Encryption and Keying. VEEEK uses RC5 algorithm to perform encryption and keying. It is very easy to implement because it uses only primitive arithmetic operations on the blocks of plaintext. And it provides one of the major security services namely confidentiality by the help of RC5 Encryption scheme which is based on the permutation codes on the blocks of data.

7. References

[1] S. Uluagac, R. A. Beyah, Yingshu Li, A. Copeland "VEBEK: Virtual Energy Based Encryption and Keying for wireless sensor networks " IEEE Transaction on Mobile Computing vol. 9 No 7 pp.994-1007 July 2010.
[2] S. Uluagac, C. Lee, R. Beyah, and J. Copeland, "Designing Secure Protocols for Wireless Sensor Networks," Wireless Algorithms, Systems, and Applications, vol. 5258, pp. 503-514, Springer, 2008.
[3] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors," Comm. ACM, vol. 43, no. 5, pp. 51-58, 2000.
[4] R. Roman, C. Alcaraz, and J. Lopez, "A Survey of Cryptographic Primitives and Implementations for Hardware-Constrained Sensor Network Nodes," Mobile Networks and Applications, vol. 12, no. 4, pp. 231-244, Aug. 2007.
[5] H. Hou, C. Corbett, Y. Li, and R. Beyah, "Dynamic Energy-Based Encoding and Filtering in Sensor Networks," Proc. IEEE Military Comm. Conf. (MILCOM '07), Oct. 2007.
[6] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," Proc. Ninth ACM Conf. Computer and Comm. Security, pp. 41-4, 2002.
[7] Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," Proc. ACM MobiCom, pp. 56-67, Aug. 2002.
[8] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," Ad Hoc Networks, vol. 3, pp. 325-349,May 2005.

K. Ravi Chythanya Received B. Tech degree in Computer science and Engineering from Ramappa Engineering College (affiliated to JNTU Hyderabad), Warangal. He is pursuing M.Tech in Computer Science and Engineering in S. R. Engineering College (affiliated JNTU Hyderabad), Warangal. His research interests are in the areas of Wireless Sensor Networks, Mobile Adhoc Networks, Network Security and Wireless Networks.

S.P. Anandaraj Received B.E in Computer Science and Engineering Specialization in April 2004. He was awarded Honor in M.Tech(CSE) in the April 2007. Currently, he is pursuing Doctoral in Computer Science and Engineering Discipline. He is a Life Member in professional Bodies like ISTE, CSI, IEEE. His research spanned a large number of disciplines, emphasizing on wireless sensor networks, Network Security. He has authored, edited and coauthored several Conference Publications. In the course of his research and teaching, Mr. S.P. Anandaraj has mentored over several B.Tech and M.Tech students and authored for about 10 Journals.

Mrs. S. Padmaja received MCA degree from University PG College (affiliated to KU), Karimnagar. She is working as sr. Asst. Professor in Sree Chaitanya college of Engineering, Karimnagar. She has authored an IEEE publication in the year 2011. Her research interests are in the areas of Wireless Adhoc Networks, MANETs, Wireless Sensor Networks.