IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 6, Ver. VI (Nov – Dec. 2014), PP 01-07
www.iosrjournals.org

An Approach to Identify Dishonest Nodes Using MAC Protocol in Multihop Cellular Networks

Mr. P.Vijaya Babu Varma[1], Mrs. Lavanya Pamulaparty[2], Mr. T. Praveen Kumar[3]

[1]*(CSE, Methodist College of Engineering & Technology/Osmania University, India)*
[2]*(CSE, Methodist College of Engineering & Technology/Osmania University, India)*
[3] *(CSE, Methodist College of Engineering & Technology/Osmania University, India)*

Abstract: *In mobility based networks, the mobile nodes usually relay on other node packets rather than their own rightful packet for enhancing network performance. It happens due to the availability of some selfish or malicious nodes which do not cooperate in transmission of received packets to honest nodes, which totally reduces the network performance. A fairness issue arises when a selfish/malicious node access the advantage of the honest nodes without providing their contributions to honest nodes which significantly degrades the performance of network and May results in failure of entire communication. In this paper, we are analyzing network performance by using Message Authentication Code (MAC) Based protocol which can identify misbehaving/malicious nodes.MAC uses only fast symmetric cryptographic operations, making it suitable for multicast networks, where latency is an important factor. For its construction, we used an efficient key assignment based on Blom's scheme, and a Merkle tree to provide authenticity during our identification routine. Further, this scheme support addition of new nodes in to the existing network only after proper authentication is done.*

Keyword: *Mobile networks, honest nodes, Symmetric cryptography, node authentication.*

I. Introduction

Mobile cellular network (MCN) [1], [2], [3], [4] follows the architecture which incorporates the ad hoc characteristics into the cellular system. A node's traffic is usually relayed through other nodes to the destination. The mobile nodes commit bandwidth, data storage, CPU cycles, battery power, etc., forming as group of resources that can be shared among them. A privilege that all nodes can obtain from the group of resources is more than they can obtain on their own. The considered MCN is used for many common life applications where the network has long life and the mobile nodes are supposed to have long-term relations with the network. Multihop packet relay can reduce the dead areas by extending the communication range of the base stations without extra costs. It can also reduce the energy consumption because packets are transmitted over shorter distances, and improve the area spectral efficiency and the network throughput and capacity [5], [6], [7]. However, due to involvement of autonomous devices in packet relay, the packet routing process suffers from new security challenges that endanger the practical implementation of MCN. The assumption that the network nodes are willing to relay other nodes' packets may not hold for common life applications where the nodes are autonomous and selfish or malicious in the sense that they aim to maximal own benefits and shows minimal contributions towards network. Although the proper network operation requires the nodes to collaborate, collaboration consumes the nodes' resources such as energy and computing power, and does not provide direct benefits. Selfish nodes are not interested in cooperation without incentive and make use of the cooperative nodes to relay their packets, which has a negative effect on the network fairness and performance. A fairness issue arises when selfish nodes take advantage of the cooperative nodes without any contribution to them. The selfish behavior also significantly degrades the network performance, which may result in failure of the total communications [8], [9]. In this article we present a mechanism that allows honest nodes to exclude misbehaving nodes, by not forwarding packets coming from these nodes in the future. To achieve detection, we base our scheme on the security properties of systems aimed at detecting pollution in network coding. To provide authentication, we present a fast authentication scheme based on Blom's key distribution scheme [10]. Our construction guarantees either that the sender of a packet is identified or an honest node does not waste significant resources checking a packet whose sender cannot be identified.

Unlike other constructions in the literature [11], [12], where a central authority is needed to exclude a misbehaving node; our construction allows any honest node to take proper action against attackers, without contacting the network's central authority; since it is based on cryptographic primitives with provable security properties, its security is high. Our proposal is also reasonable in terms of assumptions; we do not require knowledge of the topology beyond neighboring nodes, making it suitable for dynamic networks. The price we pay is: linear transmission overhead in the number of neighbors a given node has another not so significant drawback is that, the proposal becomes vulnerable when more than c attackers collude; however, this does not represent a significant challenge, since the collusion bound can be set arbitrarily large without significant impact in performance.

II. Related Work

In tamper-proof device (TPD)-based incentive mechanisms [17], [18], [19], a TPD is installed in each node to manage its credit account and secure its operation. In Nuglets [17], [18], the self-generated and forwarding packets are passed to the TPD to decrease and increase the node's credit account, respectively. The packet purse and the packet trade models have been proposed. In the packet purse model, only the source node pays by loading some credits in each packet before sending it. Each intermediate node acquires the amount of credits that cover the packet's relaying cost. In the packet trade model, each intermediate node runs an auction to sell the packets to the following node in the route. In this way, each intermediate node earns some credits and the destination node pays the total packet relaying cost. In SIP [19], after receiving a data packet, the destination node sends a payment RECEIPT packet to the source node to issue a REWARD packet to increment the credit accounts of the intermediate nodes. In CASHnet [20], [21], the source node's traffic credit account is charged and a signature is attached for each data packet. Upon receiving the packet, the destination node's traffic credit account is also charged and a digitally signed ACK packet is sent back to increase the helper credit accounts of the intermediate nodes. Users regularly visit service points to buy traffic credits with real money and/or transfer helper credits to traffic credits. The TPD-based incentive mechanisms suffer from the following problems. First, the assumption that the TPD cannot be tampered is neither secure nor practical for MCNs. That is because the nodes are autonomous and self-interested and the attackers can communicate freely in an undetectable way if they could compromise the TPDs [22]. Moreover, since the security protection of these mechanisms completely fails if the TPDs are tampered, only a small number of manufacturers can be trusted to make the network nodes, which is too restrictive for common life networks. Second, a node cannot communicate if it does not have sufficient credits at the communication time. The nodes at the network edge cannot earn as many credits as the nodes at other locations because they are less frequently selected by the routing protocol. Furthermore, the credit distribution has direct impact on the network performance, e.g., if a small number of nodes have a large ratio of the network credits, the network performance significantly degrades because the rich nodes are not motivated to cooperate and the poor nodes cannot initiate communications. Finally, since credits are cleared in real time, the network performance degrades if the network does not have enough credits circulating around. In [23], it is shown that the overall credits in the network decline gradually because the total charges are not necessarily equal to the total rewards. That is because the source node is fully charged after sending a packet but some intermediate nodes may not be rewarded when the route is broken. Although CASHnet can alleviate this problem by buying credits with real money, it is shown in [23] that in spite of having helper credits, some nodes starve because they cannot find a service point to convert them to traffic credits. In order to eliminate the need for TPDs, a central bank called the AC (Accounting Center) can be used to store and manage the nodes' accounts. In [24], the source node appends a payment token to each transmitted packet, and each intermediate node uses its secret key to check whether the token corresponds to a winning ticket. Winning tickets are submitted to the AC to reward the winning nodes. The source and destination nodes are charged per packet but the intermediate nodes are rewarded per winning ticket. In a security flaw, the colluding nodes can exchange tokens to be checked in each node to steal credits. However, due to the nature of the reputation systems, some honest nodes may be falsely identified as cheaters and the colluding nodes may manage to steal credits. Each node in a route buys packets from the previous node and sells them to the next node. The packets' buyers contact the AC to get deposited coins and the packets' sellers submit the coins to the AC to claim their payment. However, the interactive involvement of the AC in each communication session is not efficient and creates a bottleneck at the AC. So AC has to properly authenticate every node which is joining to the network.

III. Identifying Malicious Nodes

In this section we will present some existing authentication mechanisms, along with a key assignment needed for our proposal.

3.1 Digital Signatures

Digital signatures allow any node in possession of a public key P, check that a node in possession of the corresponding private key S generated a message m; however, the knowledge of P does not allow nodes to produce valid digital signatures. To achieve this property, some kind of asymmetry is needed. In the RSA signature scheme [13], asymmetry comes from number theory. Compute $n = pq$ (public key), where p, q are prime numbers of a suitable size; then, compute $\varphi(n) = (p-1)(q-1)$. Select a number e relatively prime to $\varphi(n)$ and find its inverse modulo $\varphi(n)$. To sign $0 < m < n$, compute $\sigma = md \bmod n$; to verify the signature, check if $m = (\sigma e \bmod n)$. Another source of asymmetry to create digital signatures comes from time. This idea is exploited by the Tesla [14] protocol. The construction assumes nodes are loosely synchronized; using this; nodes can generate signatures that can be checked at a future time:

- Assume there is a one way function *h*; this is given a value *s*, computing *h*(*s*) is easy; but given *h*(*s*) is not possible for an adversary to find *s*.
- Invoke *h*, *t* times using the output of the previous invocation as input for the next one; the value for the first invocation will be *s*. In mathematical terms, this is: *h*0 = *h*(*s*), *hi* = *h* (*hi*−1). Publish *ht* as the "public key".
- During the first period of time, compute a MAC involving the message *m* using *ht*−1 as a key to the fast symmetric cryptographic function (e.g. HMAC). Finally, publish the result of the MAC along with *ht*. In general, use *hi*−1 as a secret input to a function and publish *hi*.

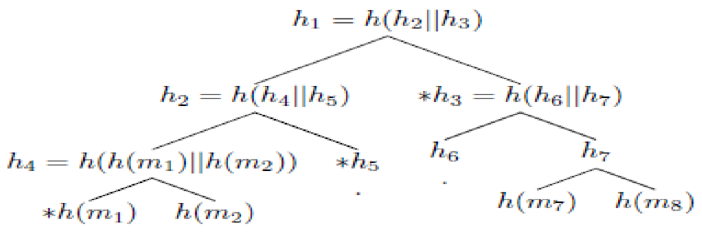

Figure 1. Example of a signature for *m*2 using a Merkle tree, appended nodes are marked with (∗).

- Nodes cannot check a signature for the current period, but once the period expires, the information is revealed. Security comes from the fact that it is computationally impossible for an adversary to simulate the chain, unless they actually know *s*. Since no expensive number theory functions are used, this scheme is very efficient, but incurs in an initial delay. A real time authentication function can be achieved if the source can perform initial packet buffering [15].

In the multicast authentication literature, the Wong and Lam scheme [16] provides fast stream authentication using a binary tree. In this scheme, the root of the tree is delivered reliably. To verify a particular message *mi*; *h* (*mi*) and the siblings of its ancestors are appended to the message (*h* is a cryptographic hash function). Figure 1 contains an example of signature for message *m*2, where the nodes of the tree appended to the message have an asterisk (∗). To verify the signature, the verifier simply recomputes the tree using the given information; then if the root matches the digitally signed data, received at a previous stage or from a trustworthy entity, the node can conclude the message is authentic.

3.2. Blom's Scheme

Blom's scheme [3] allows every user in a network, to share a secret with any other user in an efficient way. Let *D* be a secret random *c*×*c* symmetric matrix and *G* be a public *c*×*n* generator matrix of a Maximum Distance Separable (MDS) code; this is a code that meets the Singleton bound: for a (*n*, *k*, *d*)*q* then *k* = *n*−*d*+1; Reed-Solomon codes have this property. All elements from both matrices and operations are carried in the field F*q*. The set of secrets of the system is given by *A* = (*DG*) *TG*, where (*DG*) *T* denotes the transpose of (*DG*). The following derivation shows that *A* is symmetric:

A = (*DG*) *TG* = *GTDTG* = *GT* (*DG*) = *AT*

The shared secret between users *i* and *j* is *Aij* = *Aji*; to compute this value, the *i*-th user receives the *i*-th row from the matrix *Ki* = (*DG*) *T*. To get the shared secret *Aij* ; *i* computes *KiG*∗*j* , where *G*∗*j* is the *j*−th column of *G*; similarly *j* computes *Aji* = *KjG*∗*i*. If *x* ≤ *c* attackers collude, they get no information about secrets not in the collusion.

3.3 HMAC

HMAC [25] is a cryptographic MAC, based on a hash function *h* and a random key *k* (|| denotes the concatenation operator). Given information *m* to be authenticated as

HMAC (*k*, *m*) = *h* ((*k* ⊕ opad) ||*h* ((*k* ⊕ ipad) ||*m*))

Where opad=0*x*36|| . . . ||0*x*36 and ipad=0*x*5*c*|| . . . ||0*x*5*c* are fixed values padded to match the key length. If a node wishes to verify that a node in possession of *k* created a message, it must recompute the function from the received message and then check if the output matches what was received. Proofs of securities for HMAC are found in [26].

IV. Work Model

As our proposal relies on an existing selfish/dishonest node prevention scheme, which we will denote as a Source Message Authentication Codes (SMAC). MACs introduced by our proposal will be denoted as Relay MACs (RMACs). The key idea of the protocol is that it guarantees authentication for all messages using a fast function, before the slow SMAC verifying routine is applied.

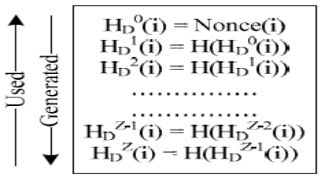

Figure 2. Hash Chain generation by receiving node

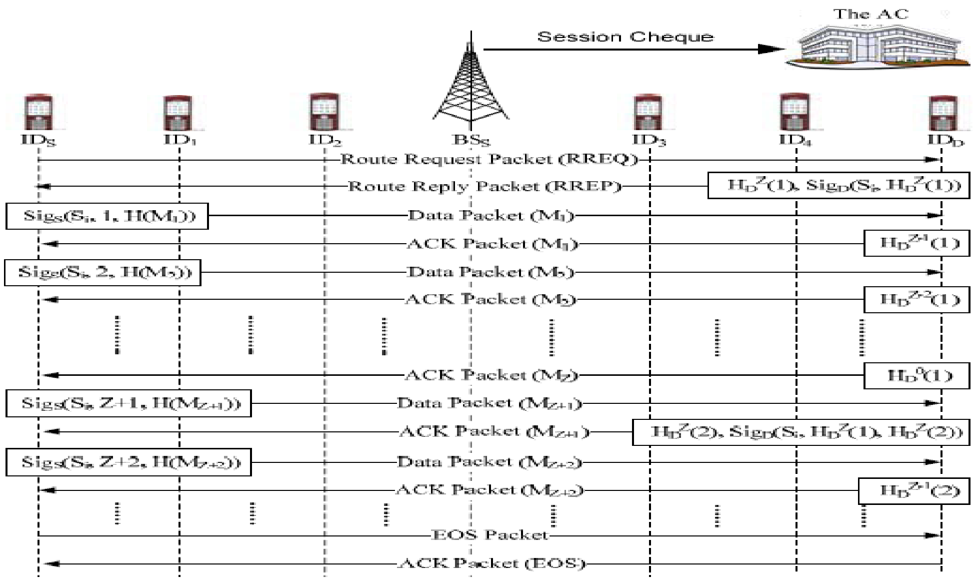

Figure 3. The exchange of keys between nodes.

4.1 Initialization

The sources initialize every node in the network and themselves, with two sets of secret values *Si* and *Ri*. The purpose of *Si* is to give the node the ability to authenticate information sent by the sources; this information will be modified using network coding operations. *Ri* is given according to Blom's scheme; its purpose is to generate shared keys *kij* to authenticate messages among neighbors. Recall from Blom's scheme that these values correspond to the *i*−th row of the matrix (*DG*) *T;* in addition, the identifier *i* is also provided to the node. A digital signature of all the keys in the system is given to each node, along with information to produce a proof that one key was assigned by the source to a particular node. We will now explain how these are generated. To avoid the use of digital signatures based on number theory at the nodes; we will use the tree construction to sign the whole square matrix of mutual secrets *A*. The idea is to embed *A* in the tree, as a set of row vectors appended one after another. Recall from Blom's scheme that the secrets given to node *i*, are represented by the *i*−th row of matrix *A*, that will be denoted by *Ai*∗. The first step taken by the source is to produce a hash of each row in *A* independently, by using each coordinate of *Ai*∗ as a separate message for a Merkle tree, whose root for each row will be denoted by *h*1,*i*.

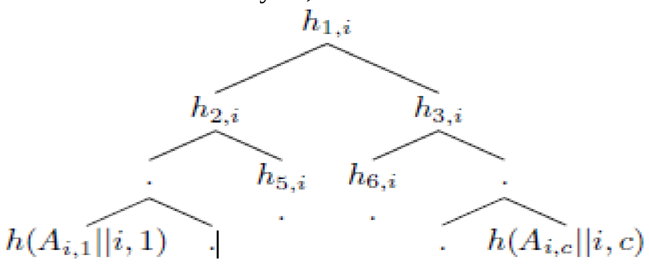

Figure 4. Computing the hash of i-th row in A.

The above figure shows this procedure; the reason we concatenate the actual coordinates of the secret to the invocation of the hash function is to guarantee the secret was in that given position in matrix *A*. Next, create a vector *v* = (*h*1, 1, *h*1, 2 . . . *h*1, *c*−1, *h*1, *c*) where each component is the root of the tree computed from each row. Now we use this vector as an input to another tree whose root will be named *σA*; *σA* is the signature for the whole matrix *A*. In this computation, the additional string is not needed.

id	*r*	*m*	SMACS	RMACS

Figure 5. Packet structure

The only remaining thing to give the nodes, is information to prove other nodes, one secret was assigned to them by the sources. To accomplish this, nodes receive the siblings of their ancestors in the Merkle tree created from *v*. The reasoning behind this, is that only node *i* is able to replicate the path for any of its secrets until *h*1, *i*; then, by using the information from the siblings from his ancestors, any node can verify the result without contacting the source.

4.2 Transmission

When any node *i* (including the source) want to transmit a message, it creates a packet with the following format:

- id: Id of the the last relay that processed the message; for this case *i*.
- *r*: Increasing value used to derive a different key for computing RMACs.
- *m*: Data to be sent.
- SMACs: A suitable pollution prevention scheme.
- RMACs: One RMAC for each one of its neighbors *j*; first, a master key is derived for each neighbor *κij* = HMAC (salt, *kij*), where salt is a random public system parameter; *kij* is derived using Blom's scheme. The key used to generate a RMAC for node *j* is:
- *k_ij* = HMAC (*κij, r*). Next, let *m_* = (id, *r, m, SMACS*); the output of one RMAC function is (id*j* ||HMAC (*k_ij, m_*||id*j*)).

4.3 Relay Processing

When node *j* receives a packet, it verifies the RMAC intended for it. If the RMAC is not authentic, the packet is discarded immediately; otherwise, the node stores the packet until the SMAC in the packet can be verified (buffering is common in Tesla-based protocols).If the verification for SMACs is successful, the packet is coded with other packets by generating a linear combination of them; otherwise, the node increases a "bad events counter" for the sender and calls the decision routine (section-D); this counter is set to 0 periodically. If the decision routine returns "true", packets from sender *i* are not relayed anymore.

To inform other nodes of the discovery of a misbehaving node *i*, entry *Aji* = *Aij* is sent to the neighborhood, along with information from the internal part of the Merkle tree only known to node *i*. We want to point out that the whole internal tree needs to be computed only once, so it can be queried later to accelerate the procedure, the total amount of information stored is twice the number of nodes in the network, times the size of the output of the hash function As it was stated in the initialization stage, nodes complement their part of the tree with the message received, if the result of the tree computation equals *σA*, the secret is considered legitimate. Note that the number of hash invocations is 2 log2 (*n*), where *n* is the number of nodes in the network. Consider now node *l* who receives the new secret and who still considers *i* as honest; then, *l* uses that key to check the RMAC authenticity of packets coming from *i*, whose SMACs have not been authenticated yet. Every time a RMAC is verified successfully and the corresponding SMAC is invalid, we increment the counter for "Bad events" proportional to the number of valid RMACs we can check. If the number of bad events from node *i* exceeds threshold *x, l* labels *i as* compromised and reveals *Ail* = *Ali* to his neighbors.

4.4 Decision Routine

This routine is only called when either:

1. A node presents a valid RMAC for another node; 2. The message authenticated by the SMAC is not authentic. Assuming there are *c* (or less) compromised nodes, an attacker can cause the first event by either, forging the SMACs for one node and letting an honest node forward the packet (attack I); or forging the RMAC himself (attack II). In attack I, we assume the attacker can forge a valid SMAC with probability 1/*p*; thus, the probability in succeeding in this attack *v* times is:

$$(1/p)^{v}.1 \qquad (1)$$

Here, the first term is the probability of an attacker forging the SMAC *v* times. The second term is the probability of the second event happening; the value is 1 because an honest node will always forward packets that pass the verification routine for SMACs.

In attack II, the first event occurs with probability 1/|RMAC|; hence the probability of the first event happening *v* times is given by 1/|RMAC|*v*. If both events happen *v* times, the probability of labeling an honest node as compromised is given by:

$$1/|RMAC|v.\ (1-1/p)^{v} \qquad (2)$$

Which is the probability of submitting a valid RMAC, times the probability of the validated message to be wrong by chance. Both events are considered independent because the RMAC and SMAC routines are completely unrelated due to their construction. The system administrator can determine a threshold that is reasonable for false positives depending on the system parameters and equations (1) and (2).

V. Conclusion

Our Analysis work shows a fair and secure cooperation mechanism for MCN, In order to fairly and efficiently charge the source and destination nodes, the lightweight hashing operations are used to reduce the number of public-key-cryptography operations. Our extensive analyses have demonstrated that our incentive mechanism can secure the payment and significantly reduce the overhead of storing, submitting, and processing the checks. Node non repudiation can be achieved using a hash chain at the source node side to efficiently verify the message integrity at each intermediate node. By using our analytical procedure AC can process the identity of selfish/dishonest node that involves themselves in sessions with the intention of dropping the data packets to launch Denial of- Service attacks. By using our method the AC can process the checks to identify the irrational nodes which improves reliable, efficient and secure transfer of packet and which enhance the network performance.

References

[1] Y. Lin and Y. Hsu, “Multihop Cellular: A New Architecture for Wireless Communications,” Proc. IEEE INFOCOM, vol. 3, pp. 1273-1282, Mar. 2000.

[2] X. Li, B. Seet, and P. Chong, “Multihop Cellular Networks: Technology and Economics,” Computer Networks, vol. 52, no. 9, pp. 1825-1837, June 2008.

[3] C. Gomes and J. Galtier, “Optimal and Fair Transmission Rate Allocation Problem in Multi-Hop Cellular Networks,” Proc. Int’l Conf. Ad-Hoc, Mobile and Wireless Networks, Aug. 2009.

[4] H. Wu, C. Qios, S. De, and O. Tonguz, “Integrated Cellular and Ad Hoc Relaying Systems: iCAR,” IEEE J. Selected Areas in Comm., vol. 19, no. 10, pp. 2105-2115, Oct. 2001.

[5] G. Shen, J. Liu, D. Wang, J. Wang, and S. Jin, “Multi-Hop Relay for Next-Generation Wireless Access Networks,” Bell Labs Technical J., vol. 13, no. 4, pp. 175-193, 2009.

[6] R. Schoenen, R. Halfmann, and B. Walke, “MAC Performance of a 3GPP-LTE Multihop Cellular Network,” Proc. IEEE Int’l Conf. Comm. (ICC), pp. 4819-4824, May 2008.

[7] Third Generation Partnership Project, Technical Specification Group Radio Access Network, “Opportunity Driven Multiple Access,” 3G Technical Report 25.924, Version 1.0.0, Dec. 1999.

[8] S. Marti, T. Giuli, K. Lai, and M. Baker, “Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,” Proc. ACM MobiCom, pp. 255-265, Aug. 2000.

[9] P. Michiardi and R. Molva, “Simulation-Based Analysis of Security Exposures in Mobile Ad Hoc Networks,” Proc. European Wireless Conf., Feb. 2002.

[10] R. Blom, “Non-public key distribution.” in Advances in Cryptology: Proceedings of CRYPTO ’82, 1982, pp. 231–236.

[11] A. Le and A. Markopoulou, “Cooperative defense against pollution attacks in network coding using spacemac,” CoRR, vol. abs/1102.3504, 2011.

[12] Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, “Mis: Malicious nodes identification scheme in network-coding-based peer-to-peer streaming,” in INFOCOM, 2010 Proceedings IEEE, 2010.

[13] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Common. ACM, vol. 26, pp. 96–99, January 1983. 35:50–52.

[14] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “The tesla broadcast authentication protocol,” RSA CryptoBytes, pp. 2–13, 2002.

[15] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, “Efficient and securesource authentication for multicast,” in In Network and Distributed System Security Symposium, NDSS ’01, 2001, pp. 35–46.

[16] C. K. Wong and S. Lam, “Digital signatures for flows and multicasts,” in Network Protocols. Proceedings. Sixth International Conference on, Oct 1998, pp. 198 –209.

[17] L. Buttyan and J. Hubaux, “Enforcing Service Availability in Mobile Ad-Hoc WANs,” Proc. ACM MobiHoc, pp. 87-96, Aug. 2000.

[18] L. Buttyan and J. Hubaux, “Stimulating Cooperation in Self- Organizing Mobile Ad Hoc Networks,” Mobile Networks and Applications, vol. 8, no. 5, pp. 579-592, Oct. 2004.

[19] [19] Y. Zhang, W. Lou, and Y. Fang, “A Secure Incentive Protocol for Mobile Ad Hoc Networks,” ACM Wireless Networks, vol. 13, no. 5, pp. 569-582, Oct. 2007.

[20] A. Weyland and T. Braun, “Cooperation and Accounting Strategy for Multi-Hop Cellular Networks,” Proc. IEEE Local and Metropolitan Area Networks (LANMAN ’04), pp. 193-198.

[21] A. Weyland, “Cooperation and Accounting in Multi-Hop Cellular Network,” PhD thesis, Univ. of Bern, Nov. 2005.

[22] J. Hubaux, L. Buttya´n, and S. Capkun, “The Quest for Security in Mobile Ad Hoc Networks,” Proc. ACM Symp. Mobile Ad Hoc Networking and Computing, Oct. 2001.

[23] A. Weyland, T. Staub, and T. Braun, “Comparison of Motivation- Based Cooperation Mechanisms for Hybrid Wireless Networks,”J. Computer Comm., vol. 29, pp. 2661-2670, 2006.

[24] M. Jakobsson, J. Hubaux, and L. Buttyan, “A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks,” Proc. Seventh Financial Cryptography (FC ’03), pp. 15-33.

[25] M. Bellare, R. Canetti, and H. Krawczyk, “Keying hash functions for message authentication,” in Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO ’96. London, UK: Springer-Verlag, 1996, pp. 1–15.

[26] New proofs for NMAC and HMAC: Security without collision-resistance, Lecture Notes in Computer Science, vol. 4117. Springer, 2006.

P Vijaya Babu Varma received his Bachelor’s degree in computer science from and received Master’s degree in Information Technology from Jawaharlal Nehru Technological University of Hyderabad, India. Presently, he is a professor in the Department of Computer Science and Engineering at Methodist College of Engineering and Technology, Osmania University, Hyderabad. His research interests include Adhoc and mobile networks, information storage and retrieval, Web Mining, Clustering technology and computing, and information security. He is a senior member of Computer Society of India.

Lavanya Pamulaparty, Associate Professor and Head of the Dept. of CSE, MCET, Osmania University, Hyderabad, obtained her Bachelor’s degree in computer science from Nagpur University of KITS, Nagpur, India, and Masters Degree in Software Engineering from School of Informatics from JNT University Hyderabad, India, and Pursuing the PhD degree in computer science and engineering from JNT University Hyderabad. Her research interests include information storage and retrieval, Web Mining, Clustering technology and computing, performance evaluation and information security. She is a senior member of the ACM, IEEE and Computer Society of India.

T. Praveen Kumar received his Bachelor’s Degree in Computer Science and Engineering from JNTUH and Master’s Degree in Software Engineering from JNTUH. He is presently working as Assistant Professor in MCET, Osmania University. His research interests include Data Mining, Information Security and Databases. He is the life member of Indian Society for Technical Education
.