CAGE: Clique-based Assignment of Group kEy

Avinash Srinivasan, Feng Li, and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: {asriniva@, fli4@, jie@cse.}fau.edu

Minglu Li
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, P. R. China
Email: mlli@sjtu.edu.cn

***Abstract*—Due to the unique operational environment of WSNs combined with their communication medium that is open to eavesdroppers, the traditional wireless network threats manifest in new forms. Consequently, key management protocols have become paramount in mitigating the damage caused. Numerous group-key protocols have been proposed in an effort to restrict access to only legitimate users. However, the group-key assignment protocols in literature neither address the issue of the size of a group nor its geographic boundaries. Consequently, in applications like watchdog-based reputation monitoring systems, malicious users are encouraged to pollute the reputation values by bad-mouthing benign nodes. On the otherhand, pairwise-key protocols are very restrictive and impose substantial storage overhead on resource constrained sensors. They do not suit the reputation monitoring systems either, since messages encrypted with pairwise keys cannot be monitored by watchdogs. In this paper, we propose CAGE, a novel, distributed, clique-based group-key assignment protocol, which distinctly addresses the size and geographic restrictions on groups. Our protocol is a simple distributed method, yet effective in securing the neighborhood communication. We prove that CAGE is an optimal solution through simulations and analysis.**

***Index Terms*—Clique, clustering, group-key, reputation, security, wireless sensor networks, trust.**

I. Introduction

In WSNs, one key requirement is that all communications be protected since the transmission medium is open and vulnerable to interception and overhearing. Prototypically, encryption has been used to overcome this problem by rendering the intercepted message gibberish. Encryption allows any node to intercept the message but the intercepting node can decrypt the message only if it is authorized. This authorization is usually provided by means of keys with which the message is encrypted. How keys are generated and assigned is a well-researched area and will not be further discussed here. Our primary focus in this paper is to propose a novel, clique-based group-key assignment protocol to restrict group membership to legitimate members.

Numerous group-key assignment protocols have been developed to permit only legitimate group members to take part in any group communication. However, one common drawback in all these protocols is that none of them give insight as to how the group size is determined. They also do not impose any spatial restrictions on the group membership. This makes the group membership and its geographical boundary very fuzzy. The above drawback can be overcome by using a pairwise-key protocol, in which each message is encrypted with a key shared strictly between two nodes. Although the pairwise-key protocol is more secure, it increases the storage and communication overhead significantly. It necessitates a node to retransmit the message multiple times since each time the transmitted message is encrypted with a key that it shares with only one node. In particular, pairwise-key protocols don't suit watchdog-driven reputation and trust-based monitoring systems, which is a potential application for our model. Reputation and trust-based systems have been used to compliment the security loopholes in cryptographic systems, such as insider attacks in which the adversary is a legitimate member of the network. We will not be discussing this further due to paper size limitations. However, we will provide a brief overview of reputation and trust-based systems in Section II-A and refer interested readers to [15] for a detailed discussion on reputation and trust-based systems.

In light of the above discussion, we draw our motivation and propose CAGE, a novel, distributed, clique-based group-key assignment protocol. In CAGE, group membership is restricted to a one-hop neighborhood and each group is assigned a single key. An immediate observation is that CAGE reduces the number of keys a node stores by a significant amount compared to the pairwise-key protocol. Also, the number of retransmissions is substantially lower in CAGE compared to the pairwise-key protocol. This conservation is critical in resource-constrained sensors. CAGE is quite similar to the work in [1]. However, the main deviation of our work comes from the fact that in CAGE, each node is required to share at least one clique with each of its neighbors. In [1], this is not a requirement, and therefore their model generates fewer cliques compared to CAGE. CAGE is also similar to the NP-Complete minimum clique cover problem [14]. But, the main deviation in our work is that in CAGE, the main objective is to find locally maximum cliques as opposed to generating a minimum number of cliques.

Our contributions in this paper can be summarized as follows:

- Clique-based group-key assignment has been considered for the first time.
- CAGE is the first group-key protocol to clearly lay down the size and spatial restrictions on group membership.

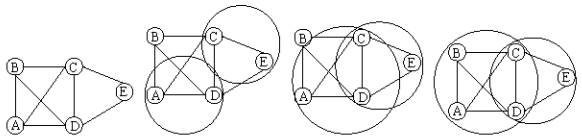


Fig. 1. (a) Network with 5 nodes (b) - (d) Cliques formed by A and C

- CAGE is the first key assignment protocol that can be applied exclusively for securing secondhand information sharing in reputation and trust-based systems.
- The proposed protocol is a distributed approach for securing neighborhood communication in WSNs. Nonetheless, CAGE can be easily extended to other networks like MANETs.
- We confirm the optimality and robustness of CAGE through simulation and analysis.

II. Related Work

Since this paper is an attempt to bring two disconnected areas together, we present related work from both these areas. First, we review existing reputation and trust-based systems. Then we briefly discuss the existing pairwise and group-key protocols.

A. Reputation and Trust-based Systems

Michiardi and Molva [13] proposed CORE, which has a watchdog along with a reputation mechanism to distinguish between subjective, functional, and indirect reputation, all of which are weighted to get the combined reputation of a node. Here, nodes exchange only positive reputation information. The authors argue that this prevents badmouthing attacks. However, they do not address the issue of collusion of malicious nodes to create false praise. Buchegger and Boudec [11] have presented CONFIDANT with predetermined trust, and later improved it with the Bayesian trust system and a passive acknowledge mechanism (PACK) respectively. This model makes misbehavior unattractive in MANETs based on selective altruism and utilitarianism. CONFIDANT is a distributed, symmetric reputation model which uses both first-hand and second-hand information for updating reputation values. Mundinger and Boudec [12] have presented a two-dimensional reputation system for protecting the system from liars to ensure cooperation and fairness in mobile ad-hoc networks.

B. Pairwise and Group-key Protocols

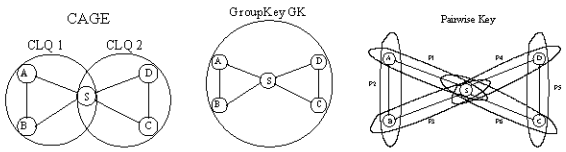
Numerous pairwise-key protocols [2], [3], [4], [5] have been presented. [2] is polynomial-based key pre-distribution protocol while [3] makes use of sensors' location information to establish pairwise keys. [4] is a probabilistic key pre-distribution method for establishing pairwise keys. In [5], Chan et al. extended the idea presented in [4] and developed two key pre-distribution techniques: q-composite key pre-distribution and random pairwise keys scheme. On the otherhand, several group key protocols have been proposed [6], [7], [8]. Group key protocols can be broadly classified into two groups: centralized and distributed. In centralized group key management protocols, there is a central authority that is trusted by everyone in the network. The central authority generates keys and distributes them. However, this approach has traditionally suffered from two weaknesses. First, there is a single point of failure. When the central authority fails or malfunctions, the security of the entire system is jeopardized. Second, the centralized system does not scale well as the number of members increases. Distributed group key management protocols overcome the above two drawbacks effectively. In this approach, the key is generated either collaboratively by the members themselves or by the leader the members elect for the group. The drawback with this kind of approach is that when members join or leave the group, rekeying is necessary to ensure forward and backward secrecy, which is a computationally-expensive task.

III. Reputation and trust-based system: Overview

In a reputation and trust-based system, each node monitors the behavior of nodes in its neighborhood using a *watchdog* mechanism and has two types of information available: *first-hand* and *secondhand*. The firsthand information is gathered by virtue of direct observation. To a node, this is the most reliable piece of information since it is observed directly. The observations are recorded in two parameters α and β, denoting good and bad behavior respectively, which is then converted into a reputation value using the Beta distribution function $Beta(\alpha, \beta)$ [9]. However, if nodes are allowed to build reputation values based solely on firsthand information, it could take a substantial amount of time before the system is bootstrapped to a stable state. Hence, nodes are encouraged to publish their findings in their neighborhood. This is known as secondhand information. Nodes usually perform a simple deviation test before accepting the secondhand information of other nodes in an effort to filter out false information published by malicious nodes [13]. If a publishing node passes the deviation test, then its secondhand information is considered compatible and is accepted. Otherwise, the secondhand information is considered incompatible and is discarded. Later on, when a decision has to be made for choosing a neighbor for any network activity like routing, a node uses the accumulated reputation values to choose the most trustworthy neighbor.

IV. CAGE

We consider a network with N homogeneous sensors. Each sensor is randomly assigned a unique ID prior to deployment. After deployment, every node i broadcasts its ID and degree information in its neighborhood $N(i)$. The neighborhood of a node is divided into two groups:- $N(i) = N(i)^{\{A\}} \bigcup N(i)^{\{I\}}$ where $N(i)^{\{A\}}$ consists of nodes that are active and contest for *Initiator*, and $N(i)^{\{I\}}$ consists of inactive nodes that cannot contest for *Initiator*. Note that an *Initiator* is the node that initiates the clique formation process. The operation of CAGE is formally presented in Algorithm 1. To assume the role of an *Initiator*, nodes compete with other active nodes in



(a) (b) (c)

Fig. 2. (a) - (c) Comparison of number of keys in CAGE, Group-Key, and Pairwise-Key.

Algorithm 1 Clique Formation

```
1: init in each active neighborhood initiate clique formation;
2: for each node i in N(init) do
3:    if i is not in any clique that init belongs to then
4:       init starts a new clique with {init, i};
5:       for each node k in N(init) do
6:          if k is a common neighbor of all nodes in the current
            clique then
7:             Update the clique to include k;
8:          end if
9:       end for
10:   end if
11: end for
```

their neighborhood. The nodes with the highest ID in their respective neighborhood win the contest. *Initiator* nodes will be referred to as $init$ in the rest of this paper. $init$ start the process of clique formation by inviting nodes in their neighborhood. The order in which the nodes are invited to join the clique has no impact on the number or size of the cliques generated. This is because Algorithm 1 ensures the generation of locally maximum cliques at all times and this property shall be confirmed by Theorem 1. Each node i maintains a list, C^i_{list}, to keep track of the cliques it belongs to, and each entry in the list will be of the form C_j, which indicates that i belongs to the j^{th} clique. Once $init$ has shared at least one clique with all its neighbors, it is marked as inactive and rendered ineligible to be an *Initiator* henceforth.

Now, once again, all active nodes compete to be the *Initiator* and the whole process, as discussed above, repeats. Algorithm 1 terminates when all nodes have been marked as inactive exhausting their turn as an *Initiator*. We do acknowledge that all the cliques can be generated in a single round if Algorithm 1 is run in parallel on all the nodes and we refer to it as the *One Round* method. Nodes can then exchange information and eliminate redundant cliques. However, in the *One Round* method, every node in a single neighborhood will generate the same cliques. Under this scenario, information exchange incurs a lot of traffic and bandwidth and the number of redundant cliques generated is very large for resource-constrained WSNs. This is in confirmation with the simulation results presented in Figure 4 (e). Hence, we opt for CAGE, the method as presented in Algorithm 1, as a better alternative. Further, a more in-depth comparative study between the two methods is on our agenda for future work.

For discussion, consider Figure 1. Let A be the highest ID node, followed by B, C, D, and E. All these nodes contest to assume the role of an *Initiator*. However, since A is the highest ID node, it wins the contest and becomes the *Initiator*. A now starts the clique formation process. Note that nodes B, C, and D cannot assume the *Initiator* role simultaneously with A since they all belong to the same neighborhood. Also, node E cannot assume the *Initiator* role simultaneously with A since C is the highest ID node in that neighborhood. Initially, node A invites node B to form the clique C_1. To begin with, C_1 has only two members $\{A, B\}$. Then A checks to see if any of its neighbors are also neighbors of B. A finds that C is a common neighbor of both A and B. Hence, C is included in C_1. A continues this process, checking at each stage if it has a neighbor that is a common neighbor of all the nodes currently in C_1 and if so, it adds that node to C_1. The process terminates when node A cannot find any more nodes to add to C_1. In the above example, the algorithm terminates with $C_1 = \{A, B, C, D\}$. Now, A sends a copy of C_1 to all the members of C_1 and the members update their clique list. In this scenario, since A shares a clique with all its neighbors, A is marked as inactive and cannot be an *Initiator* henceforth.

Now, once again nodes B, C, D, and E compete for the *Initiator* role. This time B wins the contest. However, since B already shares a clique with all its neighbors, it does not generate any new clique and is marked as inactive. This process continues until all nodes in the network are marked as inactive. In the above example the only two cliques generated are $C_1 = \{A, B, C, D\}$ and $C_2 = \{C, D, E\}$. Irrespective of which node initiates the clique formation process and in which order it induces its neighbors, only C_1 and C_2 are generated for the network setting presented in Figure 1. This property will be discussed in detail in Section V. Following this, nodes are assigned keys based on their clique membership. The members of C_1 are assigned the clique key K_{C_1} and members of C_2 are assigned the clique key K_{C_2}. Any message published in clique C_1 is always encrypted with K_{C_1} and those published in clique C_2 are always encrypted with K_{C_2}. The keys can be generated centrally by the base station, distributively by chosen clique heads, or contributively by clique members. Due to space limitations, we will not go into the details in this paper.

V. Analysis

In CAGE, note that $N(i) = N(i)^{\{A\}} \bigcup N(i)^{\{I\}}$. It follows that at any point in time $N(i)^{\{A\}} \bigcap N(i)^{\{I\}} = \emptyset$. To begin with, $N(i)^{\{I\}} = \emptyset$ and $N(i) = N(i)^{\{A\}}$. But with time, the size of $N(i)^{\{A\}}$ decreases and that of $N(i)^{\{I\}}$ increases. Finally, when all nodes have been marked as inactive, then $N(i)^{\{A\}} = \emptyset$ and $N(i) = N(i)^{\{I\}}$.

Theorem 1: CAGE always generates locally maximum cliques.

Proof: Consider a node set of k nodes denoted as $1, 2, ..., k$. Assume $C = \{1, 2, ..., k-1\}$ is a clique generated by Algorithm 1. Now, consider a node $i \notin C$ that is connected to all nodes in C. If i has the highest ID, then using CAGE i will be the *Initiator* and forms a clique $C' \leftarrow C \bigcup i$ since i is connected to all nodes in C. Else, if i is not the highest

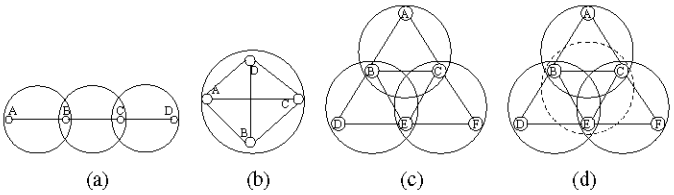


Fig. 3. a) CAGE performance same as Pairwise-key; (b) CAGE performance same as group-key; (c) Optimum- Minimum clique number; (d) CAGE-Minimum clique number.

ID node, then $init$, the highest ID node in C, which is the *Initiator* of C, will include i in $C \leftarrow C \bigcup i$. ■

As a direct implication of Theorem 1, we observe that CAGE always generates locally maximum cliques with the exact same members for a given network setting. Note that the formation of a clique C is independent of the order in which nodes are induced.

CAGE is useful only if the number of cliques generated is substantially smaller than the number of edges. Figure 3(a) depicts that even in the worst case scenario, the performance of CAGE is on par with the best case scenario of pairwise-key. In this worst case scenario, no clique containing more than two members is generated. Consequently, the number of cliques equals the number of edges. Figure 3(b) depicts the scenario wherein the average case performance of CAGE is on par with the best case performance of group-key. Here, CAGE performs as well as any other group-key protocol with no additional overhead. Note that the number of cliques generated by CAGE need not necessarily be a minimum. For instance, consider Figure 3(c), which shows the optimum results for the minimum clique cover. The number of cliques generated by CAGE is presented in Figure 3(d). It is clear that the minimum clique cover generated by CAGE is not optimum. With the *One Round* method, the redundancy is very high. For a network, if CAGE generates m cliques, then the redundancy introduced by the *One Round* method can be expressed as $\sum_{i=1}^{m}(|C_i|-1)$.

The example scenario considered for the following discussions is the publishing of secondhand information in a reputation and trust-based system. While pairwise-key is very restrictive and group-key is highly open, CAGE ensures the appropriate group size. In pairwise-key, nodes fail to detect badmouthing of malicious nodes since the message is encrypted with a key that is shared with only one node. Exploiting this situation, a node can publish different information to different nodes. On the otherhand, in a group-key, the publishing range of secondhand information is too broad. Nodes may receive information about other nodes for which they don't have any direct observation. In this scenario nodes cannot perform any deviation test before accepting the information. They have to either blindly reject it or accept it. In the former case they lose valuable information if the publishing node is benign and in the latter case they are vulnerable to brainwashing if the publishing node is malicious. CAGE ensures the right group size and range for publishing secondhand information. Malicious nodes cannot publish different information to different nodes since every member in a group is pairwise connected. Also, nodes cannot receive information about nodes that are not in their range for whom they have no direct observation unlike group-key. Hence, CAGE strikes the right balance between pairwise-key and group-key. CAGE is more robust to three different types of attacks compared to group-key and pairwise-key protocols. Please refer to Figure 2(a) - (c).

Attack Scenario 1: Attacker and attacked node belong to the same clique. Attacker badmouths in the same clique.

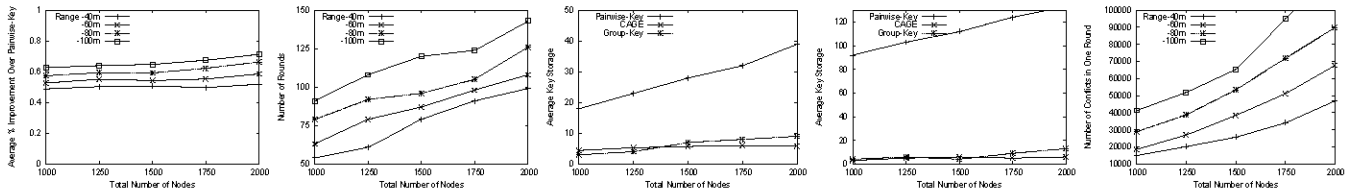
Let node A be the attacker and node B be the attacked node. With CAGE, if A badmouths B, then the message is published in C_1 encrypted with the key K_{C_1} (Figure 2(a)). S can verify A's findings in light of its own observations using a simple deviation test [13]. If the deviation test fails, then S will accordingly punish A. B will punish A irrespective of any test. Now consider the group-key protocol as depicted in Figure 2(b). Here, node A's published message is received by B, S, C and D since they all belong to the same group. As such, nodes C and D cannot verify if A is lying since they have no direct observations on B. This gives A some latitude to play foul. Finally, let us consider the pairwise-key protocol as depicted in Figure 2(c). Here, if A sends a message to S encrypting it with a pairwise-key, node S could still detect that node A is lying. However, B will be kept in the dark since the key used by A to encrypt the message is shared only between A and S. Therefore, A can get away with the misbehavior. This is not desirable in a reputation monitoring system.

Attack Scenario 2: Attacker and attacked node belong to different cliques. Attacker badmouths the attacked node in the attacker's clique.

Let node A be the attacker and node D be the attacked node. When A badmouths D in C_1, S and B can either discard the message since D is not part of C_1 or punish A for badmouthing a node that is not part of the group. However, in the above scenario, S shares C_2 with D. Even though the message is encrypted with K_{C_1}, S can punish A more severely than B, since S knows that A does not share a clique with D. If, indeed, A shared a clique with D, then S would be part of that clique since S is a neighbor of both A and D. This follows directly from Theorem 1. Now, in the same scenario, consider a group-key protocol. Here, B has no way of verifying A's claim since B has no direct observation on D. Hence, B has to take a chance in either accepting or rejecting it. With pairwise encryption, similar arguments as presented in *Attack Scenario 1* apply.

Attack Scenario 3: Attacker and attacked node belong to different cliques. Attacker badmouths the attacked node in the attacked node's clique.

Let node A be the attacker and node D be the attacked node. In this scenario, A will never be able to badmouth D in D's neighborhood since D is part of C_2 and messages in C_2 are encrypted using K_{C_2}. Since A is not part of C_2, A has no way of injecting false information into C_2. However, this is possible with the group-key protocol and pairwise encryption using similar lines of argument as presented in *Attack Scenario 1*.



(a) (b) (c) (d) (e)

Fig. 4. (a) Performance of CAGE vs pairwise-key; (b) Number of rounds in CAGE; (c) - (d) Key storage overhead pairwise-key vs CAGE; (e) Conflicts in *One Round* method

VI. Results

Our simulations were carried out on a custom Java simulator. For each trial, a field of $500m \times 500m$ was randomly seeded with arbitrarily deployed sensors and results were averaged for 1000 iterations. We have considered the number of nodes and the transmission range as tunable parameters. Figure 4(a) depicts the mitigation in storage overhead that CAGE achieves over pairwise-key protocol. It is clear that even in the worst case scenario, with 1000 nodes and a transmission range of $40m$, the total number of cliques is about 48% lower than the total number of edges. In the best case scenario with 2000 nodes and a transmission range of 100m, the total number of cliques is about 68% lower than the number of edges. CAGE, on average, generates about 58% fewer cliques compared to the number of edges. In Figure 4(b) we have presented the number of rounds required to generate all the maximum cliques. The results were averaged for 1000 different network settings. It is clear that the number of rounds is sensitive to both the number of nodes as well as their transmission range. Note that, as the transmission range increases, fewer *Initiator* nodes will be chosen in each round which subsequently increases the total number of rounds required. This is due to the fact that the size of the independent set decreases with an increase in the transmission range. Figure 4(c) presents the results comparing the key storage overhead for pairwise-key, CAGE, and group-key mechanism for a transmission range of $40m$. It is clear that group-key has the best performance up to $n = 1400$ beyond which CAGE outperforms group-key. On the otherhand, CAGE consistently outperforms pairwise-key significantly. CAGE is neither as sensitive to changes in the number of nodes nor to the transmission range as pairwise-key protocols are. Results were observed for transmission ranges of $60m$, $80m$, and $100m$ respectively. We have presented the results only for $40m$ and $100m$ (Figure 4(d)), due to paper length constraints, as they represent extreme cases. Finally, in Figure 4(e), we have shown the redundancy introduced by the *One Round* method. It is clear that even in the best case the number of cliques generated by this method is over 15000.

VII. Conclusion

In this paper, we have proposed CAGE, a novel, distributed, clique-based group-key assignment protocol. CAGE is the first distributed protocol that can be used exclusively for reputation and trust-based systems. It overcomes the communication and storage overhead of pairwise-key protocols and the spatial fuzziness of group-key protocols. We have presented a formal algorithm for CAGE and discussed it in detail. We have also presented a detailed analysis of CAGE, highlighting its strengths and confirmed through simulation that CAGE achieves optimal results. In our future work, we will do a more in-depth simulation of CAGE to further confirm its applicability.

References

[1] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, April 1997.

[2] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. *In Proceedings of SASN '03*, Fairfax, VA, October 2003.

[3] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. *In 10th ACM Conference on Computer and Communications Security*, October 2003.

[4] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. *In Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41 to 47, November 2002.

[5] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. *In 2003 IEEE Symposium on Research in Security and Privacy*, pp 197-213.

[6] A. Perrig, D. Song, and J. D. Tygar. ELK, A new protocol for efficient large-group key distribution. *In Proceedings of the IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA.

[7] S. Rafaeli, L. Mathy, and D. Hutchison. EHBT: An efficient protocol for group key management. *In Proceedings of the 3rd International Workshop on Networked Group Communications*. (London, U.K., Nov).

[8] A. C-F. Chan. Distributed Symmetric Key Management for Mobile Ad Hoc Networks. *IEEE INFOCOM*, 2004.

[9] A. Josang and R. Ismail. The beta reputation system. *In Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia, June 2002.

[10] R. Gupta and J. Walrand. Approximating maximal cliques in ad-hoc networks. *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2004*, Barcelona, September 2004

[11] S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes- Fairness In Dynamic Ad-hoc NeTworks). *Proceedings of MobiHoc 2002*, Lausanne, CH, June 2002.

[12] J. Mundinger, J.-Y. Le Boudec. Analysis of a Reputation System for Mobile Ad-Hoc Networks with Liars. *In Proceedings of The 3rd International Symposium on Modeling and Optimization*, Trento, Italy, April 2005.

[13] P. Michiardi and R. Molva. CORE: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. *Communication and Multimedia Security*, September, 2002.

[14] J. Gramm, J. Guo, F. Hffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *Accepted for publication in ACM Journal of Experimental Algorithmics*, in press.

[15] A. Srinivasan, J. Teitelbaum, H. Liang, J. Wu, and M. Cardei. Reputation and Trust based System for Ad Hoc and Sensor Networks. *In Algorithms and Protocols for Wireless Ad Hoc and Sensor Networks*, A. Boukerche (ed), Wiley&Sons, 2006.