Clique-based Group Key Assignment in Wireless Sensor Networks

Avinash Srinivasan*
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: asriniva@fau.edu
*Corresponding author

Feng Li
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: fli4@fau.edu

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: jie@cse.fau.edu

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

Abstract: Security has become the corner stone of research in wireless sensor networks (WSNs). Due to the unique operational environment of WSNs, where the communication medium is open to eavesdroppers, the threats manifest in new forms necessitating the safeguarding of communication. Consequently, key management protocols have become paramount in mitigating the damage caused. Numerous group-key protocols have been proposed in an effort to secure both inter and intra network communication. However, the group-key protocols in literature neither address the issue of the size of a group nor its geographic boundaries. Therefore, in applications like watchdog-based reputation monitoring systems, malicious users are encouraged to pollute the reputation values by bad-mouthing benign nodes and false-praising other collaborating malicious nodes. On the otherhand, though pairwise-key protocols can overcome the above drawback of group-key protocols, they are highly 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 render promiscuous watchdog monitoring systems useless. 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 and ideally suits the promiscuous monitoring requirements of reputation and trust-based systems. CAGE ensures pairwise connectivity of all group members which thwarts information asymmetry (bad-mouthing) attacks. We confirm through simulations and analysis that CAGE strikes an optimal balance between pairwise-key and group-key protocols and that it achieves an optimal solution. We then present the improvised CAGE, which we refer to as the Extended CAGE (E-CAGE). E-CAGE achieves the same results as CAGE with mitigated computation time compared to CAGE. We confirm this through simulation results.

Keywords: clique, clustering, group-key, reputation, security, wireless sensor networks, trust.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) do not command the luxury of extensive computing and battery power or the typical operational environment like their counterparts. 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 which renders 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 on how to divide a network into meaningful groups, which is an important step before establishing keys. Once the network has been divided into groups, we can use any of the existing key establishment protocols such as centralized, decentralized, contributive, etc. Therefore, this paper proposes CAGE - a novel, clique-based group-key management protocol to restrict group membership to legitimate members. CAGE ensures pairwise connectivity of all group members which thwarts information asymmetry (bad-mouthing) attacks.

Numerous group-key protocols have been developed to permit only legitimate group members to take part in any group communication. However, one common drawback in 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 (Du et al., 2003; Liu and Ning, 2003a,b), in which each message is encrypted with a key shared strictly between two nodes. Although a 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 are potential applications for our model. Reputation and trust-based systems have been used to compliment the security loopholes in cryptographic systems, such as insider attacks[1] in which the adversary is a legitimate member of the network. We will provide a brief overview of reputation and trust-based systems in Section 2.1 and refer interested readers to Srinivasan et al. (2007a) 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 also draws its motivation from Krishna et al. (1997). 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 Krishna et al. (1997), this is not a requirement, and therefore their model can generate fewer cliques compared to CAGE. CAGE is also similar to the NP-Complete minimum clique cover problem (Gramm et al., In Press). However, 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. CAGE, in principle, is equivalent to finding all locally maximum cliques of a graph G.

We realize that CAGE indeed requires considerable computation time for generating all locally maximum cliques. We thus improvise CAGE and present a new model which we refer to as the Extended CAGE (E-CAGE), which mitigates the computation time of CAGE to a large extent. E-CAGE is a variation of CAGE and technically the only difference between the two models lies in the method of choosing the *Initiator* nodes. *Initiator* nodes are the nodes that initiate the clique formation process simultaneously. In E-CAGE, adjacent nodes can be chosen as *Initiator* simultaneously unlike CAGE. This significantly reduces the number of rounds needed for the entire process of generating locally maximum cliques. Other than that, the two methods are exactly the same and they both generate the exact same cliques. Note that in E-CAGE, since adjacent nodes are chosen as Initiator nodes concurrently, there is a generation of redundant cliques. We will discuss this further when we present the model in Section 4.2.

We also compare the above two models with a brute force method of generating locally maximum cliques that we refer to as the *One-Round* method. We compare the three models for their performance in terms of their computation time and the redundancy they introduce. The objective is to reduce both time complexity and redundancy. 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.
- CAGE is the first key assignment protocol that can be applied exclusively for securing secondhand information sharing in reputation and trust-based systems.

[1]By physically capturing a node, the adversary can extract all the cryptographic material held by that node. Consequently, the adversary can get through authentication, validation, etc. Hence, the adversary is now part of the network and can launch attacks from within.

- 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.
- We present Extended CAGE (E-CAGE), an improvement of CAGE that further mitigates the computation time for generating all locally maximum cliques.
- We compare the performance of our models with a brute force method of generating cliques, which we call *One-Round*.

2 RELATED WORK

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

2.1 Reputation and Trust-based Systems

Michiardi and Molva (Michiardi and Molva, 2002) 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. Another interesting feature of CORE is that its members have to contribute on a continual basis to remain trusted. Otherwise, their reputation will deteriorate until they are excluded.

Buchegger and Boudec (Buchegger and Boudec, 2004) 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 (Mundinger and Boudec, 2005) have presented a two-dimensional reputation system for protecting the system from liars to ensure cooperation and fairness in mobile adhoc networks.

Ganeriwal and Srivastava (Ganeriwal and Srivastava, 2004) proposed a reputation-based framework for sensor networks where nodes maintain reputation for other nodes and use it to evaluate their trustworthiness. They show that their framework provides a scalable, diverse and a generalized approach for countering all types of misbehavior resulting from malicious and faulty nodes.

Algorithm 1 CAGE: Initiator Selection

```
1: while there are nodes that are active and have not served as $Initiator$ do
2:   for each node $i$ in non-decreasing order of its $ID$ that has not served as $Initiator$ do
3:     if $i$ is not PRUNED and $\forall j \in N(i)^{\{A\}}, ID_i > ID_j$ then
4:       Choose $i$ as $Initiator$;
5:     end if
6:   end for
7: end while
```

2.2 Pairwise and Group-key Protocols

Numerous pairwise-key protocols (Liu and Ning, 2003a,b; Eschenauer and Gligor, 2002; Chan, 2004) have been presented. A polynomial-based key pre-distribution protocol is presented in Liu and Ning (2003a) while Liu and Ning (2003b) makes use of sensors' location information to establish pairwise keys. A probabilistic key pre-distribution method for establishing pairwise keys is proposed in Eschenauer and Gligor (2002). Chan et al. (Chan et al., 2003b) extended the idea presented in Eschenauer and Gligor (2002) 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 (Perrig et al.; Rafaeli et al.; Chan, 2004). 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, in which case each member contributes a piece of information to the final key or by the leader the members elect for the group who is assigned the responsibility of generating the key for that 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.

3 OVERVIEW- Reputation and Trust-based Systems

In a reputation and trust-based system, each node monitors the behavior of nodes in its neighborhood using a *watchdog* mechanism. Nodes operate in a promiscuous mode to facilitate the watchdog to observe and gather in-

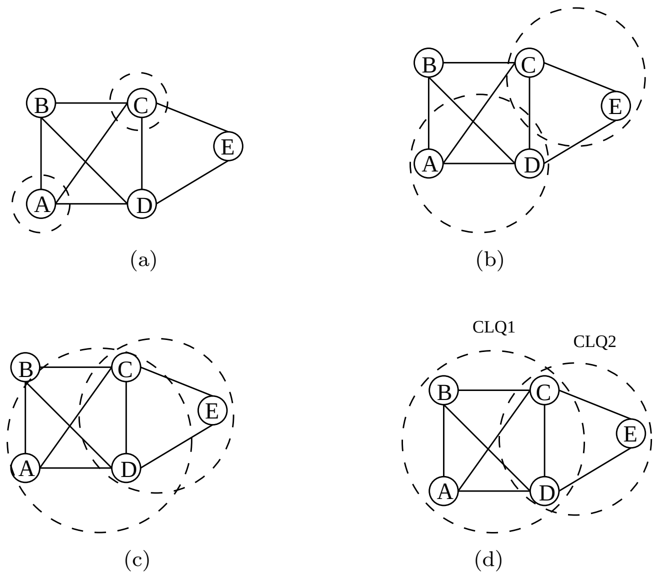


Figure 1: (a) Network with 5 nodes, (b) - (d) Cliques formed by A and C.

Algorithm 2 E-CAGE: Initiator Selection

```
1: while there are nodes that are active and have not served as $Initiator$ do
2:     for each node $i$ in non-decreasing order of $ID$ that has not served as $Initiator$ do
3:         Choose $i$ as $Initiator$ if $i$ is not PRUNED and $\forall j \in N(i)^{\{A\}}, ID_i > ID_j$;
4:         Else
5:         Choose $k \in N(i)^{\{A\}}$ as $Initiator$ if $k$ is not PRUNED and $ID_k > ID_j$, $\forall j \in N(i)^{\{A\}}$;
6:     end for
7: end while
```

formation about node misbehavior in the neighborhood. In a reputation and trust-based system, two types of information are available to each node: *firsthand* 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 the tuple (α, β), where α denotes a positive interaction or good behavior while β denotes a negative interaction or bad behavior. The information captured by the tuple (α, β) is then converted into a reputation value using the Beta distribution function $Beta(\alpha, \beta)$ (Josang and Ismail, 2002). However, if nodes are allowed to build reputation values based solely on firsthand information, it can 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. However, there can be malicious nodes in the network that intentionally publish incorrect information to pollute reputation values and mislead benign nodes. Hence, nodes usually perform a simple deviation test before accepting the secondhand information of the publishing nodes in an effort to filter out false information published by malicious nodes (Michiardi and Molva, 2002). 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. Accordingly, the tuple (α, β) is updated to reflect the collective view of the neighborhood. 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.

4 MODEL

We consider a network with N homogeneous sensors. Each sensor is randomly assigned a unique ID prior to deployment. After deployment, we assume that nodes are benign for a time period δ during which 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*. Here, the terms active and inactive are not used in scheduling context and merely refer to nodes' capability to contest in *Initiator* selection process. Note that an *Initiator* is the node that initiates the clique formation process. Therefore, $N(i) = N(i)^{\{A\}} \bigcup N(i)^{\{I\}}$. The operation of CAGE is formally presented in Algorithm 1. To assume the role of an *Initiator*, nodes compete with other active nodes in their neighborhood. The three models that we present in this paper differ in the way the *Initiator* nodes are chosen. *Initiator* nodes will be referred to as *init* in the rest of this paper.

All three models generate the same number of cliques with the same members since the primary criteria in all these methods are locally maximum cliques. Hence we use two parameters to measure the performance of the three methods. The first parameter is *Rounds*. This parameter measures the number of times the Algorithm is executed before all cliques are generated. The second parameter is called *Conflicts*. Since two neighboring nodes, when chosen as *init* simultaneously, generate two copies of one or more cliques, there will be redundancy in the cliques generated. For illustration, consider Figure 2 (a). In this network setting, nodes 3, 4, 5, 6, and 8 are chosen as *init* simultaneously, the details of which will be discussed in Section 4.2. Consequently, nodes 3 and 4 both generate a copy of the clique $\{2, 3, 4\}$, and nodes 5 and 8 both generate a copy of the clique $\{5, 7, 8\}$. Now there are two redundant cliques, which are one copy of $\{2, 3, 4\}$ and one copy of $\{5, 7, 8\}$. Therefore, $Conflicts = 2$. There are more conflicts in this scenario, but we shall not discuss them all. The *Conflicts* parameter records the number of such redundant cliques generated, by summing the number of conflicts at the end of each round.

4.1 CAGE

In CAGE, nodes compete with other active nodes in their neighborhood to assume the role of *init* and nodes with the highest ID in their respective neighborhood win the contest. *init* start the clique formation process and generate all the cliques in their neighborhood that they possibly can. The order in which the nodes are included into the clique has no impact on the number or size of the cliques generated. This is because Algorithm 1 ensures the generation of all locally maximum cliques at all times and this property shall be confirmed by Theorem 5.1 in Section 5. In CAGE, a node i is chosen as *init* for a given round if the following conditions are satisfied:

Condition 1. *i is the highest ID node in the neighborhood.*

Condition 2. *i has not been chosen as init in one of the previous rounds.*

Condition 3. *i can generate at least one clique that has not been generated in the previous rounds.*

Each node i maintains a list, C^i_{list}, to keep track of the cliques it belongs to. In this list, each entry will be of the form C_j, which indicates that i belongs to the j^{th} clique. Once *init* has generated all the cliques that it possibly can, it is marked as inactive and rendered ineligible to assume the role of *init* henceforth. Now, once again, all active nodes compete to assume the *init* role and the entire process, as discussed above, repeats until the stopping condition is reached. The stopping condition in our models is the marking of every single node as inactive. Condition 3 above is checked using the following check and prune rules.

Rule 1. *u has an outgoing edge that is not covered by any clique generated so far.*

Algorithm 3 Clique Formation

```
1: init determines node(s) to generate new clique(s) using
   $C^{init}_{list}$ and Rules 1 and 2
2: init generates new clique(s);
3: update $C^{init}_{list}$ with newly generated clique(s);
4: send out copies of new clique(s) to nodes in Step 1;
```

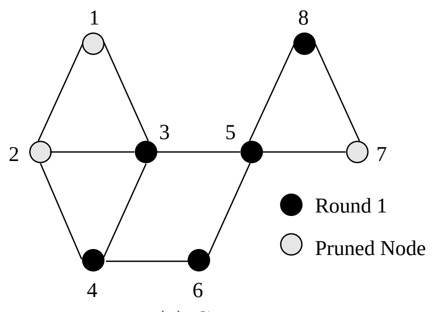
Rule 2. *u has pair(s) of connected neighbors such that*

- *neither of them have been an Initiator in previous rounds.*

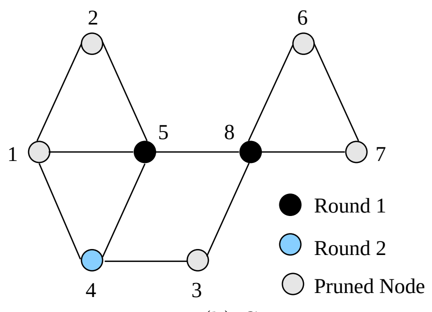
- *they have not been included, along with u, in any clique generated so far.*

If either or both of these rules are satisfied, then a node is not pruned. The check and prune phase of the algorithm ensures that only those nodes that can generate new cliques are chosen as initiators for each new round. Algorithm 1 terminates when all nodes have been marked as inactive by either exhausting their turn as *init* or being pruned in accordance with Rules 1 and 2. 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 *init*. However, since A is the highest ID node, it wins the contest and becomes the *init*. Note that nodes B, C, and D cannot assume the *init* role simultaneously with A since they all belong to the same neighborhood as A. Also, node E cannot assume the *init* role simultaneously with A since C is the highest ID node in E's neighborhood. A now starts the clique formation process and initially inducts D to form the clique C_1. To begin with, C_1 has only two members $\{A, D\}$. Then A checks to see if any of its neighbors are also neighbors of D. A finds that C is a common neighbor of both A and D. 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 A cannot find any more nodes to add to C_1. In the above example, the algorithm terminates with $C_1 = \{A, D, C, B\}$. 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 has generated all possible cliques that it can, A is marked as inactive and cannot be an *init* henceforth.

Now, once again nodes B, C, D, and E are active and compete for the *init* role. However, node B gets pruned and is marked as inactive since it already shares a clique with all its neighbors and neither Rule 1 nor Rule 2 hold true. But for nodes C, D, and E, Rule-2 holds true and hence they cannot be pruned. Finally, C, D, and E compete in the second round for the *init* role and C wins the contest and generates the clique $C_2 = \{C, E, D\}$. This process of check and prune, and clique generation continues until all nodes in the network are marked as inactive. In the above example the only two cliques generated are



(a) Case-1



(b) Case-2

Figure 2: Impact of node ID ordering on number of rounds in E-CAGE.

$C_1 = \{A, D, C, B\}$ and $C_2 = \{C, E, D\}$ requiring only two rounds effectively. 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 further discussed in detail in Section 5. 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}.

The number of keys a node stores in CAGE is equal to the number of distinct cliques it belongs to. In our example, nodes C and D belong to both cliques C_1 and C_2, and consequently store both keys K_{C_1} and K_{C_2}. Messages published in clique C_1 are always encrypted with the clique key K_{C_1} and those published in clique C_2 are always encrypted with clique key K_{C_2}. The clique keys in CAGE can be generated in any one of the following three ways: (1) centrally by the base station; (2) distributively by electing clique heads; or (3) contributively by all the clique members. Since a detailed discussion on different key generation methodologies is already available in literature (Rafaeli and Hutchison, 2003), it would be both redundant and beyond the scope of this paper. Note that in CAGE there can never be a conflict[2] since no two neighboring nodes are chosen as initiators simultaneously. This is because, in CAGE, each node competes with all the active nodes in its neighborhood and finally the highest ID node in each neighborhood is chosen. Also, since each initiator node shares the cliques it has generated with its neighbors, the neighboring nodes will not generate the same redundant cliques.

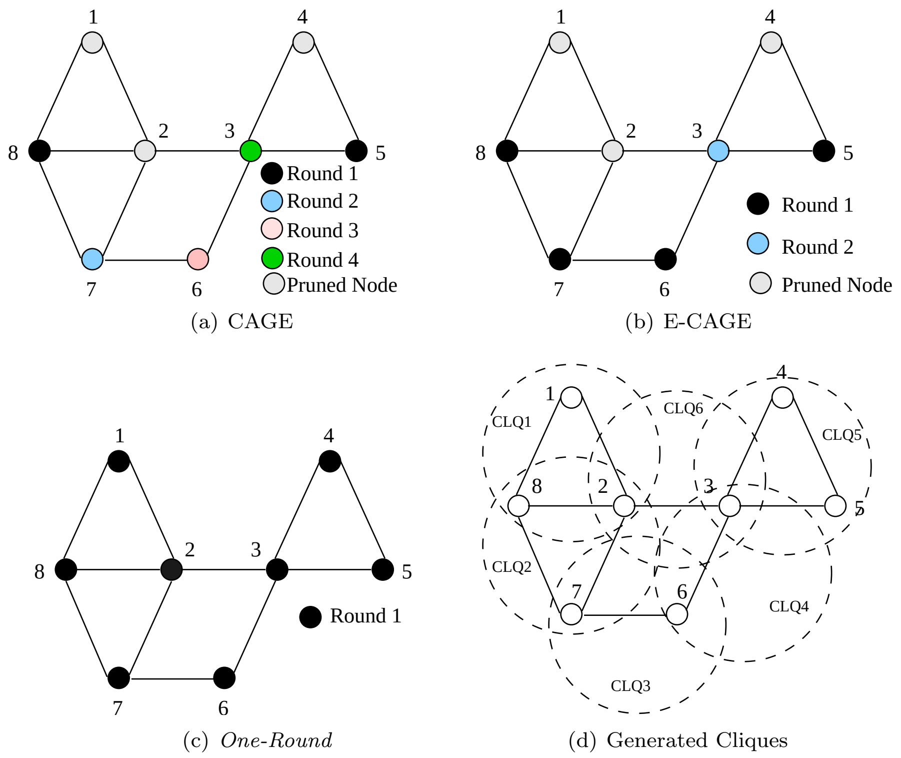
4.2 Extended CAGE (E-CAGE)

In this method too, nodes compete with other active nodes in their neighborhood to assume the role of an *init* just as in CAGE. The deviation of this model from CAGE comes from the fact that, in E-CAGE, each node chooses the node with the highest ID in its neighborhood as the *init*. If the node itself happens to be the highest ID node, then it chooses itself as *init*. Consequently, neighboring nodes may be chosen to be the *init* simultaneously which generates redundant cliques. However, this redundancy is the inevitable trade-off for generating the cliques in fewer rounds. This method makes the process of clique generation highly parallel, thereby cutting down on the number of rounds necessary. A node i is selected as an *init* for a given round if and only if Conditions 1, 2, and 3 are satisfied. If Condition 1 is not satisfied, but Condition 4 stated below holds true along with Conditions 2 and 3, then node i can still be chosen as an *init*.

Condition 4. *i has at least one neighboring node to whom i is the highest ID neighbor.*

This idea has been captured well in Figures 2 (a) and (b). It is very clear from the figures that node ID ordering has impact on the performance of E-CAGE in terms of number of rounds. We shall discuss the working of E-CAGE using the network setup in Figure 2 (a). Note that, in E-CAGE, selection of *init* nodes in both decreasing and non-decreasing order of node ID produces the same results for a given network setting. In the following discussions we always consider non-decreasing order of node ID unless otherwise stated. We begin with node 1, which has two neighbors, node 2 and node 3 respectively. Using Algorithm 2, 1 chooses 3 as the *init* in its neighborhood. Similarly, node 2, which has nodes 3 and 4 as its neighbors, chooses 4 as the *init* in its neighborhood. Node 3 which has nodes 1, 2, 4 and 5 as its neighbors, chooses 5 as the *init* in its neighborhood. Node 4 with neighbors 2, 3, and 6 chooses 6 as the *init* in its neighborhood, and node 5 with neighbors 3, 7, and 8 chooses 8 as the *init*. Node 6 with neighbors 4 and 5 chooses itself as the initiator in its neighborhood, node 7 chooses 8 as the *init* in its neighborhood, and node 8 chooses itself as the *init*. The five *init* nodes $\{3, 4, 5, 6, 8\}$, shaded in black, are shown in Figure 2 (a). These *init* nodes use Algorithm 3 to generate all the cliques they can, at the end of which they are marked as inactive. For the next round, nodes $\{3, 4, 5, 6, 8\}$ are excluded from the *init* selection process. Along with these nodes, nodes 1, 2, and 7 are pruned and marked as inactive in compliance with Rules 1 and 2 since they cannot generate any new cliques. So, for the given network settings,

[2]When two nodes generate the same clique simultaneously, the redundancy is called conflict.



(a) CAGE (b) E-CAGE (c) *One-Round* (d) Generated Cliques

Figure 3: (a) - (c) Comparison of number of rounds in CAGE, E-CAGE, and *One-Round*; (c) Cliques generated by all three methods.

all the cliques are generated in a single round with 5 redundant cliques, when compared to the one round method which would generate 9 redundant cliques for the same network setting. We will discuss the detail in Section 4.3. Figures 2 (b) show different node ID ordering for the same network setting. For the network setting in Figure 2, two rounds are needed to generate all the cliques and the the number of conflicts is only 1. In summary, for this network and any node ID ordering, a minimum of one round and a maximum of two rounds are necessary to generate all the cliques.

Figures 3 (a) and (b) intuitively capture the number of rounds required by CAGE and E-CAGE respectively to generate all the cliques for a given network setting. In this example, E-CAGE requires only 2 rounds to generate all the locally maximum cliques whereas CAGE requires 4 rounds to generate the same.

4.3 ***One-Round***

We do acknowledge that all the cliques can be generated in a single round by a brute force method which we call the *One-Round* method. In *One-Round*, Algorithm 3 is run in parallel on all the nodes in the network, i.e., all N nodes assume the role of the *init* simultaneously. Consequently, every node will initiate the clique formation process concurrently. Once all nodes terminate the clique formation process, nodes can exchange information with other nodes in their respective neighborhood. After this information exchange phase, redundant cliques can be eliminated. In Figure 3 (c), we see that a total of six locally maximum cliques can be generated. Note that both *One-Round* and E-CAGE, irrespective of the number of conflicts they generate and the number of rounds E-CAGE takes, always generate only the following six cliques: $C_1 = \{1, 2, 8\}$, $C_2 = \{2, 7, 8\}$, $C_3 = \{6, 7\}$, $C_4 = \{3, 6\}$, $C_5 = \{3, 4, 5\}$and $C_6 = \{2, 3\}$. In Figures 3 (a) - (c), all the nodes, shaded in the same color, assume the role of the initiator simultaneously.

When neighboring nodes assume the role of *init* and execute Algorithm 3 concurrently, then they generate redundant clique copies. Particularly, in *One-Round*, since every node executes Algorithm 3 simultaneously, every member of a clique will generate a copy of the clique.

To begin with, let the number of conflicts for *One-Round*, denoted by C_{OR} be zero, i.e., $C_{OR} = 0$. Similarly, let $C_{EC} = 0$, denote the number of conflicts in E-CAGE. We see from Figure 3 (d) that C_1 has three members, and therefore, 3 copies of C_1 will be generated in *One-Round* since all three members initiate the clique formation simultaneously. Consequently, $C_{OR} = C_{OR} + 2 = 2$, since two of the three copies of C_1 are redundant. The same clique, C_1, is generated using E-CAGE with no conflicts since only node 8 forms the clique, and therefore $C_{EC} = 0$. Similarly, in CLQ2, there are three members, namely 2, 7, and 8. Consequently, three copies of CLQ2 will be generated out of which two copies are redundant, and therefore

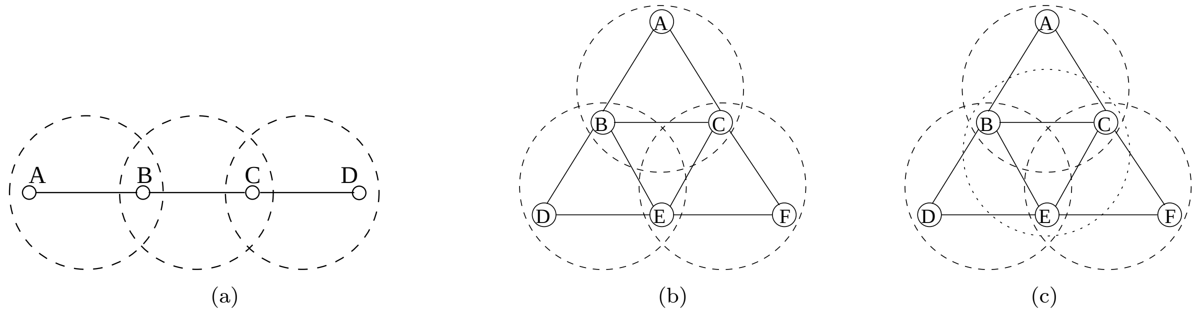


Figure 4: (a) CAGE performance same as Pairwise-key; (b) Minimum clique number for clique cover algorithm; (c) CAGE - Minimum clique number.

$C_{OR} = C_{OR} + 2 = 4$. On the otherhand, in E-CAGE, two nodes, 7 and 8, initiate the formation of C_2 simultaneously, and therefore one redundant copy of C_2 is generated making $C_{EC} = C_{EC} + 1 = 1$. Continuing our argument on these lines we have the following: after C_3 is generated, $C_{OR} = C_{OR} + 1 = 5$ and $C_{EC} = C_{EC} + 0 = 1$, after C_4 is generated $C_{OR} = C_{OR} + 1 = 6$ and $C_{EC} = C_{EC} + 0 = 1$, after C_5 is generated $C_{OR} = C_{OR} + 2 = 8$ and $C_{EC} = C_{EC} + 0 = 1$, and after C_6 is generated $C_{OR} = C_{OR} + 1 = 9$ and $C_{EC} = C_{EC} + 0 = 1$. We see that for a small network with 8 nodes and 6 cliques, *One-Round* generates 9 conflicts where as E-CAGE generates only 1 conflict. Also, note that in this particular example, E-CAGE generates all six possible cliques in a just two rounds with the check and prune rules. However, CAGE generates all six cliques in four rounds with zero conflicts. This example serves to intuitively capture the tradeoff between the three methods.

In general, a clique of size s will have $s - 1$ redundant copies. So, for a network, if Algorithm 3 generates m cliques, then the redundancy introduced by *One-Round* can be expressed as

$$\sum_{i=1}^{m}(|C_i| - 1)$$

where $|C_i|$ is the number of elements in the clique C_i. Under this scenario, information exchange incurs a lot of traffic and wasted bandwidth, and the number of redundant cliques generated is mammoth considering the resource-constraints in WSNs. This is in confirmation with the simulation results presented in Figure 8 (c).

5 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 assumed the role of an *init*, then $N(i)^{\{A\}} = \emptyset$ and $N(i) = N(i)^{\{I\}}$.

Theorem 5.1. *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 3. 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 *init* 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 ID node, then *init*, the highest ID node in C, which is the *init* of C, will include i in $C \leftarrow C \bigcup i$. $\square$

As a direct implication of Theorem 5.1, 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 into C.

CAGE is useful only if the number of cliques generated is substantially smaller than the number of edges. In the worst case scenario, the performance of CAGE will still be on par with the performance of pairwise-key. In this scenario, no clique containing more than two members is generated. Consequently, the number of cliques equals the number of edges, as shown in Figure 4 (a). Also, Note that the number of cliques generated by CAGE need not be a minimum as generated by other clique cover algorithms. For instance, consider Figure 4(b), which shows the results for the minimum clique cover. The number of cliques generated by CAGE is presented in Figure 4(c).

In the following discussion, the example scenario considered 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 bad-mouthing and false-praise 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

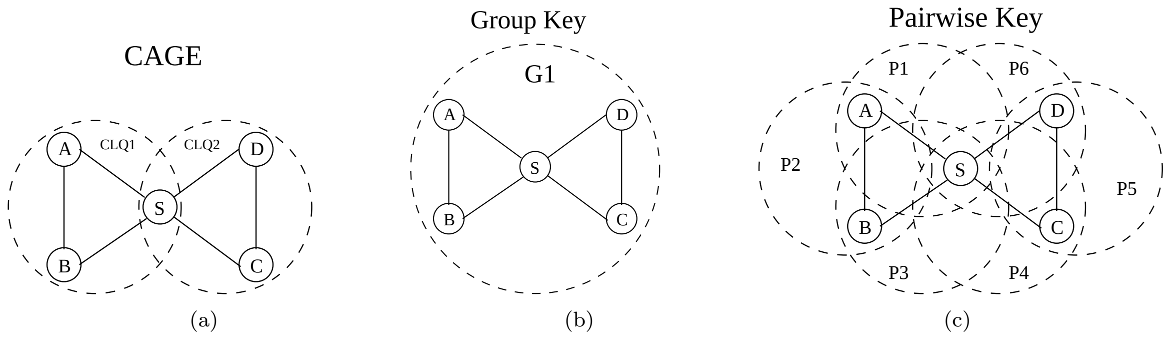


Figure 5: (a) - (c) Comparison of number of keys in CAGE, Group-Key, and Pairwise-Key.

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 brain-washing 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.

Another interesting observation is that E-CAGE is better than CAGE in certain aspects only. CAGE does generate all the cliques without generating even a single redundant clique. However, this comes at the cost of an increased number of rounds. On the otherhand, E-CAGE generates all the cliques in fewer rounds, but in each round several redundant cliques might be generated. Hence, an inevitable tradeoff between redundancy and the number of rounds exists between CAGE and E-CAGE. Please refer to Figure 5 for the following discussion.

Attack Scenario 1. *Attacker and attackee*[3] *belong to the same clique. Attacker badmouths the attackee in the same clique.*

Let node A be the attacker and node B be the attackee. With CAGE (see Figure 5 (a)), if A badmouths B, then the message is published in clique C_1 encrypted with the key K_{C_1}. Now, S can verify A's findings in light of its own observations using a simple deviation test similar to the one proposed by Michiardi and Molva (2002). If the deviation test fails, then S will accordingly punish A for badmouthing. Though B should punish A only if A is badmouthing, B is allowed to punish A regardless. However, this does not compromise the system's performance, since when B publishes its findings on A, nodes accept it only if it qualifies the deviation test. If B has punished A using a tit-for-tat[4] strategy, then it is bound to fail the deviation test and have its own reputation tarnished by other nodes in the neighborhood.

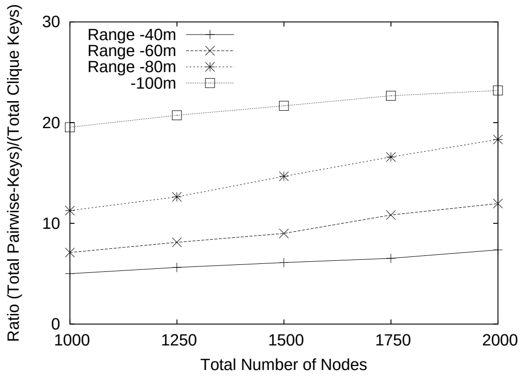
Now consider the group-key protocol as depicted in Figure 5(b). Here, node A's published message is received by B, S, C and D since they all belong to the same group. As such, nodes C and D cannot verify if A's findings are consistent with their's using a deviation test since they have no direct observations on B. Hence, they have to either accept it with a possible chance of being misinformed or reject it assuming that they did not lose any valuable information. This gives node A some latitude to play foul. Finally, let us consider the pairwise-key protocol as depicted in Figure 5(c). Here, if A sends a message to S encrypting it with a pairwise-key, node S can 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 a desirable property in a reputation monitoring system.

Attack Scenario 2. *Attacker and attackee belong to the same clique. Attacker badmouths the attackee in another clique.*

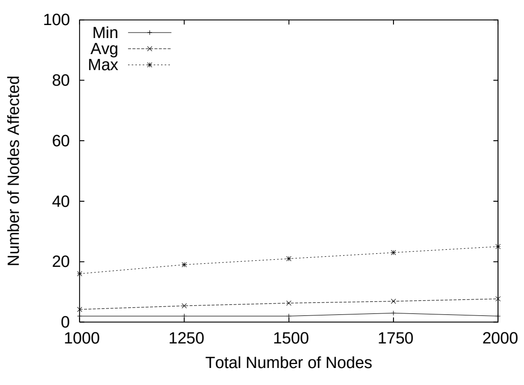
Let node S be the attacker and node A be the attackee. With CAGE (see Figure 5 (a)), if S badmouths A in clique C_2, then nodes C and D discard the message right away since node A does not belong to the clique. Alternatively, they can also punish S for badmouthing and decrease its reputation. Therefore, it's in S's best interest not to do so. However, the same network settings with a group-key protocol (see Figure 5 (b)) will allow nodes C and D to accept S's opinion on A since they belong to the same group. However, in this scenario, A will punish S since it also receives the message sent to C and D encrypted with key $G1$. With pairwise-key encryption, similar arguments as presented in *Attack Scenario 1* apply.

[3]Attackee is a node that is attacked by the attacker.

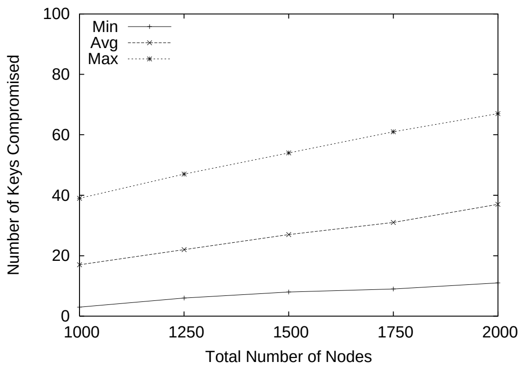
[4]Node A, irrespective of its behavior, punishes node B whenever it sees that node B has given it a bad rating.



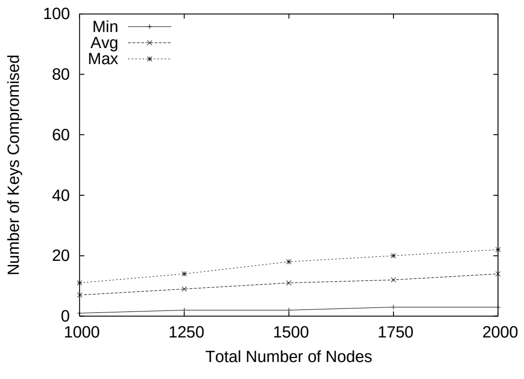
(a) Performance Improvement



(b) CAGE-1 Key compr.



(c) Pairwise-1 Node compr.



(d) CAGE-1 Node compr.

Figure 6: (a) Performance of CAGE vs pairwise-key; (b) Number of nodes affected in CAGE with one key compromise; (c) Number of keys compromised in pairwise-key protocol with one node compromise;(d) Number of keys compromised in E-CAGE with one node compromise.

Attack Scenario 3. *Attacker and attackee belong to two different cliques and have no clique in common. Attacker badmouths the attackee in the attacker's clique.*

Let node A be the attacker and node D be the attackee. When A badmouths D in C_1 (see Figure 5 (a)), S and B can either discard the message since D is not part of clique C_1 or punish A for badmouthing about a node that is not part of the group. However, in the above scenario, since S shares the clique C_2 with D, it can punish A more severely than B, as 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 5.1. Now, in the same scenario, consider a group-key protocol (see Figure 5 (b)). 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-key encryption, similar arguments as presented in *Attack Scenario 1* apply.

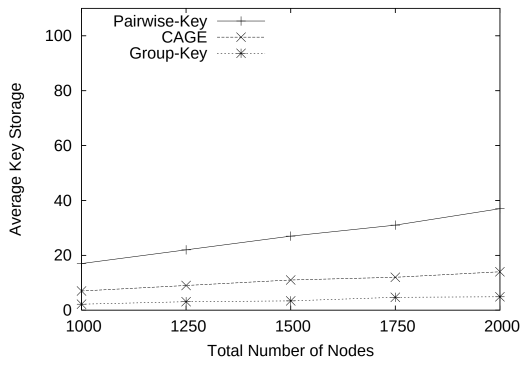
Attack Scenario 4. *Attacker and attackee belong to two different cliques and have no clique in common. Attacker badmouths the attackee in the attackee's clique.*

Let node A be the attacker and node D be the attackee. 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 information into C_2. However, this is possible with the group-key and pairwise-key protocols using similar lines of argument as presented in *Attack Scenario 1.*

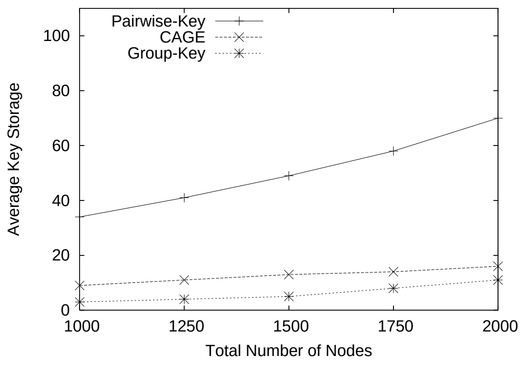
6 SIMULATION

6.1 Environment

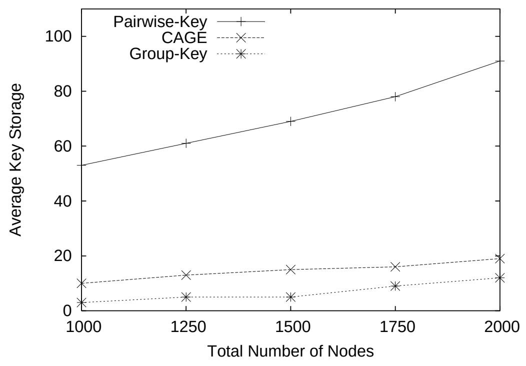
Our simulations were carried out on a custom Java simulator. For each trial, a $500m \times 500m$ field was randomly seeded with arbitrarily deployed sensors and results were averaged for $1,000$ iterations. In our simulations, we have considered the number of nodes N and the transmission range R as the tunable parameters. The number of nodes were varied from 1,000 to 2,000 in steps of 250 while transmission range was varied from 40m to 100m in steps of 20m. With the above variations, we could generate 20 different network settings. In our simulations, we consider N homogeneous sensors and model the network as an undirected graph $G = (V, E)$. Here V is the set of vertices and



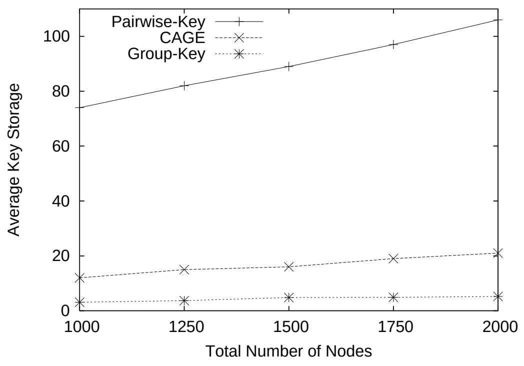
(a) Range $40m$



(b) Range $60m$



(c) Range $80m$



(d) Range $100m$

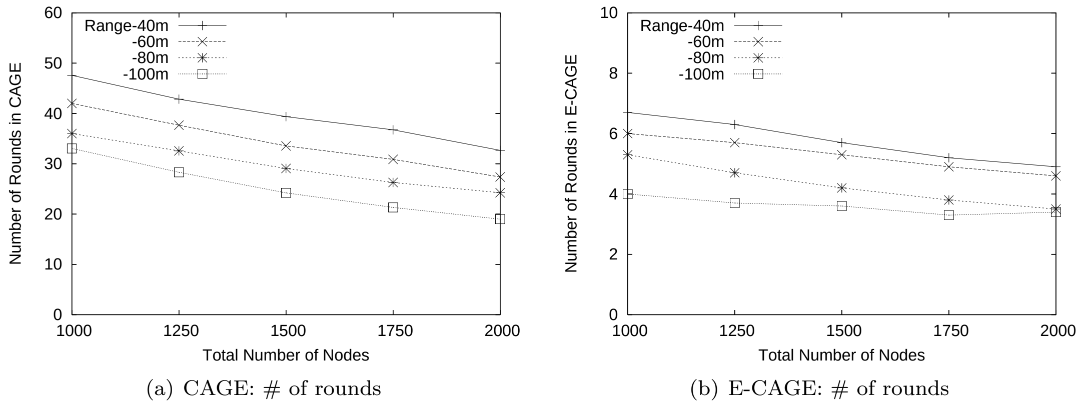
Figure 7: Comparison of key storage overhead in Pairwise-key, CAGE, and group-key for different transmission ranges.

E is is the set of edges. An edge exists between two vertices if they lie in each other's communication range. Each edge is treated as a bidirectional link, i.e., if node i can communicate with node j on edge (i, j), then node j can communicate with node i on edge (j, i).

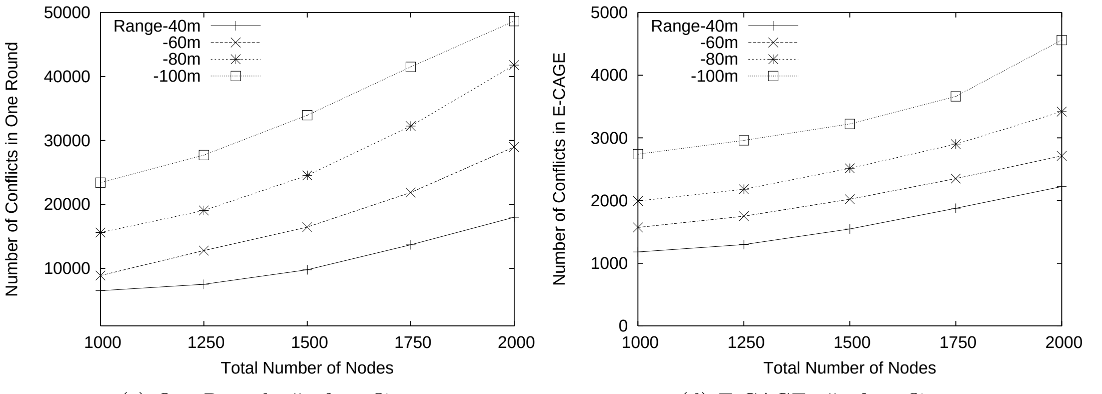
6.2 Results

In Figure 6(a), we have presented the results that depict the improvement of CAGE over pairwise-key protocols. Note that, any simulation scenario, in which there is no comparison between CAGE and E-CAGE indicates that they have the same performance. The graph plotted is the ratio of the total number of pairwise keys to the total number of clique keys in the network. It is clear that even with $1,000$ nodes and a transmission range of $40m$, which represents a relatively sparse network, the total number of clique keys is about 5 folds lower than the total number of pairwise-keys. With $2,000$ nodes and a transmission range of 100m, which represents a very dense network, the total number of clique keys is about 23 folds lower than the number of pairwise-keys. This is a significant reduction in the key storage overhead. CAGE, on average, generates about 13 folds fewer clique keys compared to the total number of pairwise-keys.

We have also studied the extent of damage caused by limiting the adversary's capacity from two perspectives: (1) compromise one key- the adversary can compromise only one key in the entire network, (2) compromise one node- the adversary can compromise only one node in the entire network. In pairwise-key protocols, if the adversary is allowed to compromise a single key, then it affects only two nodes since a key is shared only between two node. On the otherhand, in CAGE, if the adversary is allowed to compromise a single clique key, then the number of nodes that get affected depends on the size of the clique whose key is compromised. In the best case, only two nodes get affected and in the worst case, there is no limit on the number of nodes that get affected, since the compromised clique can be of any arbitrary size. We have presented the results plotting the extent of damage caused, in terms of number of nodes affected, along the y-axis in Figure 6 (b). In this graph, we have plotted three scenarios: (1) Minimum number of nodes affected, (2) Maximum number of nodes affected, and (3) Average number of nodes affected. From the "Min" curve, it is evident that the number of nodes affected is equal to that in pairwise-key protocols. However, the "Max" and the "Avg" indicate that the amount of damage induced is much higher compared to pairwise-key protocols. Also, for the network in general, as the density increases, more and more nodes tend to get affected with a single key compromise in CAGE. Although we have not plotted it in our graph, it should be noted that the num-



(a) CAGE: # of rounds (b) E-CAGE: # of rounds



(c) One-Round: # of conflicts (d) E-CAGE: # of conflicts

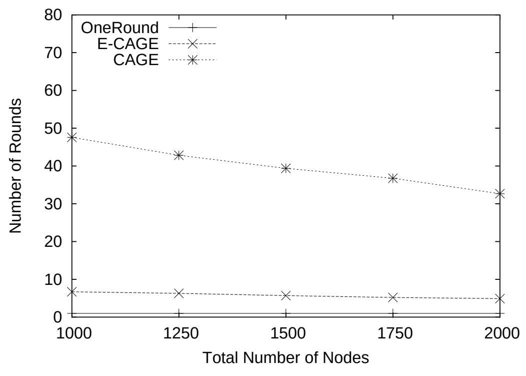
Figure 8: (a) Number of rounds in CAGE; (b) Number of rounds in E-CAGE; (c) Conflicts in *One-Round*; (d) Conflicts in E-CAGE.

ber of affected nodes with the compromise of a single key in group-key protocols will be much higher than that in CAGE, since groups tend to be considerably larger than cliques.

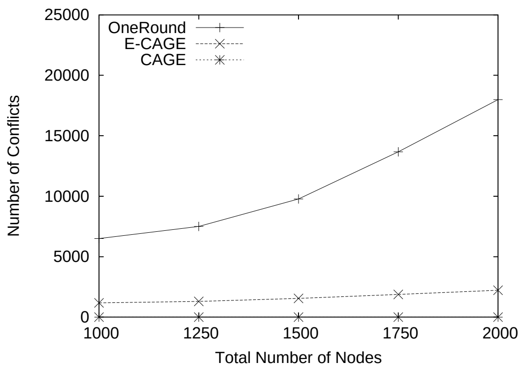
Nodes in both pairwise-key protocols and CAGE hold multiple keys unless a node has only one neighbor. In Figures 6 (c) and (d), we have presented the results simulating the situation in which the adversary is allowed to compromise a single node in pairwise-key protocols and CAGE respectively. Here, the induced damage is measured in terms of total number of keys compromised with the compromise of a single node. We have plotted three curves representing the minimum, maximum, and average number of keys compromised by compromising a single node. It is evident from the results that CAGE successfully curtails the induced damage compared to pairwise-key protocols. For sparse networks, the gain in CAGE is not significant for the "Min" curve. However, as the network gets denser, the amount of damage caused in pairwise-key protocols is many folds higher when compared to CAGE. The "Max" and "Avg" curves consistently outperform the pairwise-key protocols significantly.

In Figure 7(a), we have presented the results comparing the key storage overhead for pairwise-key, CAGE, and group-key protocols for a transmission range of $40m$. It is evident from the graph that CAGE consistently outperforms pairwise-key protocol significantly. It can also be inferred that CAGE is neither as sensitive to changes in the number of nodes nor to the transmission range as pairwise-key protocols are. However, the group-key protocol marginally outperforms CAGE. Simulations were also carried out for transmission ranges of $60m$, $80m$, and $100m$, which have been presented in Figure 7 (b), (c), and (d) respectively.

In Figure 8(a), we have presented the number of rounds required by CAGE to generate all the locally maximum cliques. The results were averaged for 1,000 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. The number of rounds decreases with increasing transmission range as well as with increasing number of nodes. This follows from a simple argument: as the transmission range increases, more nodes belong to a single neighborhood. Consequently, the size of a clique increases. As a result, smaller cliques get eliminated and more nodes tend to prune themselves during the *init* selection process. The number of rounds required by CAGE varies approximately between 47 rounds for the most scarce network with 1,000 nodes and 40m transmission range and 20 for the most dense network setting with 2,000 nodes and 100m



(a) Range=40m



(b) Range=40m

Figure 9: (a) Comparison of number of rounds required for the three methods; (b) Comparison of conflicts in the three methods.

transmission range. Similarly, in Figure 8 (b), we have presented the results for the number of rounds required by E-CAGE to generate all the maximum cliques. Here again the results were averaged for 1,000 different network settings for statistical stability. We can see that E-CAGE is extremely efficient and generates all the locally maximum cliques in fewer rounds compared to cage. The number of rounds required by E-CAGE varies approximately between 7 rounds for the most scarce network with 1,000 nodes and 40m transmission range and 4 rounds for the most dense network setting with 2,000 nodes and 100m transmission range. In Figure 8 (c), we have presented the the number of conflicts generated by the *One-Round* method. It is clear that even in a sparse network with 1,000 nodes and 40m transmission range, the number of redundant cliques generated by this method is extremely large and over 6,000. It should be noted that CAGE does not generate any conflicts, since in each round only the highest ID node is chosen from a neighborhood, i.e., neighboring nodes are never chosen as *init* simultaneously. In Figure 8 (d), we have presented the conflicts resulting in E-CAGE. It is evident that E-CAGE is very efficient compared to *One-Round* in generating all the cliques with fewer conflicts and curtails the redundancy, on average, at least by a factor of 10.

In Figure 9 (a), we have presented the results comparing the number of rounds required by the three methods to generate all the cliques with transmission range of 40m. The performance improvement of E-CAGE over CAGE is significantly high. The results comparing the number of conflicts generated in the three methods for a transmission range of 40m have been presented in Figure 9 (b). These two graphs are plotted for a comparative look at the performance of the three methods.

7 CONCLUSION

In this paper, we have proposed CAGE, a novel, distributed, clique-based group-key assignment protocol for WSNs. 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 *Initiator* selection in CAGE and discussed it in detail. We have also presented a formal algorithm for generating all locally maximum cliques. We then propose the Extended CAGE (E-CAGE) method to further mitigate the computation time of CAGE in generating all locally maximum cliques. An algorithm has been presented delineating the *Initiator* selection process in E-CAGE. We have conducted simulation studies comparing CAGE with pairwise-key and group-key protocols. The results confirmed that CAGE strikes an optimum balance between these two extremes. We have also compared E-CAGE with CAGE and the brute force *One-Round* method. From simulation results it is very clear that E-CAGE strikes a balance between CAGE and *One-Round* method in both computation time and redundancy. We have also presented a detailed analysis of CAGE (E-CAGE), highlighting its novelty, strength, and applicability.

REFERENCES

Krishna, P., Vaidya, N. H., Chatterjee, M., and Pradhan, D. K. (1997) 'A cluster-based approach for routing in dynamic networks', *In ACM SIGCOMM Computer Communication Review*, 1997.

Marti, S., Giuli, T. J., Lai, K., and Baker, M. (2000) 'Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks', *In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

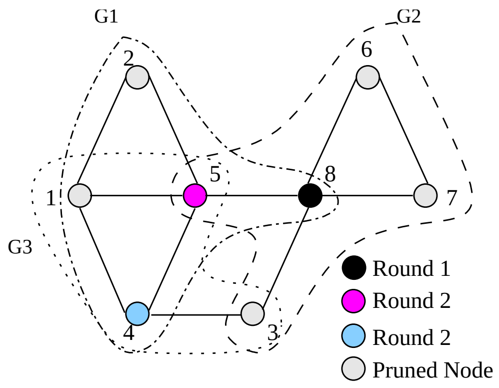


Figure 10: Group-Key Implementation Example.

Perrig, A., Song, D., and Tygar, J. D. () ‘ELK, A new protocol for efficient large-group key distribution’, *In Proceedings of IEEE Symposium on Security and Privacy*, 2001.

Rafaeli, S., Mathy, L., and Hutchison, D. () ‘An efficient protocol for group key management’, *In Proceedings of the 3rd International Workshop on Networked Group Communications*, 2001.

Josang, A. and Ismail, R. (2002) ‘The beta reputation system’, *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

Buchegger, S. and Le Boudec, J.-Y. (2002) ‘Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes- Fairness In Dynamic Ad-hoc NeTworks)’, *In Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

Michiardi, P. and Molva, R. (2002) ‘CORE: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks’, *Communication and Multimedia Security*, 2002.

Eschenauer, L. and Gligor, V. D. (2002) ‘A key-management scheme for distributed sensor networks’, *In Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp 41-47, 2002.

Du, W.,Deng, J., Han, Y., and Varshney, P. (2003) ‘A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks’, *In Proceedings of 10th ACM Conference on Computer and Communications Security*, 2003.

Rafaeli, S. and Hutchison, D. (2003) ‘A survey of key management for secure group communication’, *ACM Computer Survey*, 2003.

Liu, D. and Ning, P. (2003-A) ‘Location-based pairwise key establishments for static sensor networks’, *In Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003.

Liu, D. and Ning, P. (2003-B) ‘Establishing pairwise keys in distributed sensor networks’, *In 10th ACM Conference on Computer and Communications Security*, 2003.

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

Buchegger, S., Tissieres, C., and Le Boudec, J.-Y. (2004) ‘A Test-Bed for Misbehavior Detection in Mobile Ad-hoc Networks - How Much Can Watchdogs Really Do?’, *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2004.

Buchegger, S. and Le Boudec, J.-Y. (2004) ‘A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks’, *In Proceedings of Third Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2004.

Ganeriwal, S. and Srivastava, M. (2004) ‘Reputation-based framework for high integrity sensor networks’, *In Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, pp 66-77, 2004.

Chan, A. C-F. (2004) ‘Distributed Symmetric Key Management for Mobile Ad Hoc Networks’, *In Proceedings of IEEE Conference on Communications (INFOCOM)*, 2004.

Chan, H. and Perrig, A. (2005) ‘PIKE: Peer Intermediaries for Key Establishment in Sensor Network’, *In Proceedings of IEEE Conference on Communications (INFOCOM)*, 2005.

Mundinger, J. and Le Boudec, J.-Y. (2005) ‘Analysis of a Reputation System for Mobile Ad-Hoc Networks with Liars’, *In Proceedings of The 3rd International Symposium on Modeling and Optimization*, 2005.

Gramm, J., Guo, J., Hffner, F., and Niedermeier, R. (2006) ‘Data reduction and exact algorithms for clique cover’, *In Proceedings of 8th Workshop on Algorithm Engineering and Experiments (SIAM ALENEX)*, pp 86-94, 2006.

Srinivasan, A., Teitelbaum, J., Liang, H., Wu, J., and Cardei, M. (2006) '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.

Srinivasan, A., Wu, J., and Teitelbaum, J. (2007) 'Distributed Reputation-based Secure Localization in Sensor Networks', accepted to appear *In Special issue on Journal of Autonomic and Trusted Computing (JoATC)*, 2007.

Srinivasan, A., Li, F., Wu, J., and Li. M (2007) 'CAGE: Clique-based Assignment of Group kEy', *In Proceedings of the Second International Conference on Communications and Networking (ChinaCom)*, 2007.

8 APPENDIX

In this, section, for the sake of completeness, we describe the group-key protocol implemented if our simulations. In our group-key protocol, we use the same initiator selection process used in E-CAGE. However, here, instead of node ID, node degree is used as the *init* selection criteria. A *init* node forms its group by inducting all its 1 hop neighbors into a single group. Consequently, no two nodes in a single group are further than 2 hops. For illustration, consider Figure 10. In this network setting, we see that nodes 5 and 8 have the highest node degree. Therefore, 5 and 8 start the formation of their groups and node 5 generates $G1 = \{5, 4, 1, 2, 8\}$ and node 8 generates $G2 = \{8, 5, 3, 7, 6\}$ which completes round one. The process of selecting the highest degree node and completing the group formation process once is referred to as a round. Now, before the start of the next round, all the nodes that have already been included in a group and have no outgoing edges that are uncovered prune themselves. Returning to our example, nodes 1 and 2 prune themselves since they are already covered by $G1$ and additionally have no out going edges that are uncovered. Similarly, nodes 6 and 7 prune themselves since they are already covered by $G2$ and have no outgoing edges that are uncovered. Consequently, only nodes 3 and 4 contest for the *init* position in round two since edge $(3, 4)$ is still uncovered. However, node 4 gets chosen as *init* and forms the group $G3 = \{4, 1, 5, 3\}$. With this, every edge in the network is covered, which is the stopping condition for our group formation. For the given network setting as depicted in Figure 10, three groups are formed in two rounds.