Efficient Key Establishment for Group-Based Wireless Sensor Deployments

Li Zhou and Jinfeng Ni and Chinya V. Ravishankar
Department of Computer Science & Engineering
University of California, Riverside
Riverside, CA 92521, USA
{lzhou,jni,ravi}@cs.ucr.edu

ABSTRACT

Establishing pairwise keys for each pair of neighboring sensors is the first concern in securing communication in sensor networks. This task is challenging because resources are limited. Several random key predistribution schemes have been proposed, but they are appropriate only when sensors are uniformly distributed with high density. These schemes also suffer from a dramatic degradation of security when the number of compromised sensors exceeds a threshold. In this paper, we present a group-based key predistribution scheme, GKE, which enables any pair of neighboring sensors to establish a unique pairwise key, regardless of sensor density or distribution. Since pairwise keys are unique, security in GKE degrades gracefully as the number of compromised nodes increases. In addition, GKE is very efficient since it requires only localized communication to establish pairwise keys, thus significantly reducing the communication overhead. Our security analysis and performance evaluation illustrate the superiority of GKE in terms of resilience, connectivity, communication overhead and memory requirement.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: secuirty and protection; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Security, Design, Algorithms

Keywords

Wireless sensor networks, group-based key pre-distribution, security

1. INTRODUCTION

Sensors are typically small battery-equipped devices with very limited communication, computation, and memory capacity. They are becoming cheap enough for them to be deployed on demand, and in environments where access or maintenance is ruled out. Sensors typically need to communicate with their neighboring sensors [24, 21, 22], aggregating sensing data into a more compact report and then transmitting it to the base station over multiple hops. Messages sent between neighboring sensors may contain sensitive data or commands from the base station, so it is crucial to secure these communications. Unfortunately, resource limitations at sensor nodes rule out the use of expensive public key cryptosystems such as RSA [18] or Diffie-Hellman key agreement [5] for this purpose.

Key predistribution schemes are typically considered as an efficient way to establish pairwise keys between neighboring sensors [2, 1, 16, 24]. This task, however, is complicated by the adhoc and on-demand nature of sensor deployments. Since a sensor's neighbors are only known after deployment, it is not possible to preload these shared keys in any simple way.

The obvious approach of preloading each sensor with keys shared pairwise with every other sensor requires memory linear in the number of sensors in the network. For large networks, this is impractical. Recently, *Random key predistribution (RKP)* schemes [9, 4, 6] have been proposed for large scale sensor networks. The basic idea is to randomly preload each sensor before deployment with a set of keys from a global key pool. Since these subsets are chosen randomly, any pair of sensors will share a key with a certain probability. Two neighboring nodes can choose any element in the intersection of these subsets to be their pairwise key. When these subsets are disjoint, two neighboring nodes may establish a *path key* using intermediary nodes. These schemes are based on results from random graph theory [8], which guarantee that a random graph is connected with high probability if the number of edges in it exceeds a threshold. To further improve the network resilience of RKP schemes against node capture, [7, 14, 15] proposed *structured random key predistribution (SRKP)* schemes, which have a nice threshold property: when the number of compromised sensors is less than a threshold, any other keys shared between non-compromised sensors are affected with probability close to zero.

RKP and SRKP have attractive features. A small number of preloaded keys (say, 250) are able to almost ensure the connectivity of a large sensor network (say, 10,000 nodes) [9]. Also, each sensor incurs small communication overhead for key establishment, regardless of network size.

However, these random key predistribution schemes suffer from two major problems, which make them inappropriate for many applications. First, these schemes require that the deployment density is high enough to ensure connectivity (see Section 2.1). This requirement seriously hinders the use of RKP and SRKP when sensor networks are sparse, as is likely when sensors fail over time and new sensors are added, or when the deployments are themselves sparse.

Second, the idea of key sharing in RKP or key space sharing in SRKP, a requisite for high network connectivity, also degrades resilience against node capture. A node compromise also compromises the set of keys (or key spaces, respectively) in it, so that the security of all other sensors having these keys (or key spaces) will be weakened. Consequently, all the schemes based on key sharing or key space sharing show exponential degradation of security, when a small fraction of sensors are compromised [3]. This shortcoming seems unavoidable for RKP and SRKP schemes.

In their recent PIKE proposal [3], Chan et al. have addressed the problem of high density requirements in RKP and SRKP. In PIKE, each sensor is assigned an ID of the form (i, j), which corresponds to a location on a $\sqrt{n} \times \sqrt{n}$ grid, where n is the network size. Each sensor is also preloaded with pairwise keys, each of which is shared with a sensor that corresponds to a location on the same row or the same column of the grid. Now, any pair of sensors that do not share a preloaded pairwise key can use one or more peer sensors as trusted intermediaries to establish a path key. PIKE shows significantly improved security over SRKP due to the uniqueness of pairwise keys. However, since the intermediaries may not always be in the vicinity, PIKE requires network-wide communication to establish path keys, each of which requires $O(\sqrt{n})$ communication overhead [3]. In addition (see Section 7.2.2), a large fraction ($> 99\%$) of neighboring sensor pairs in PIKE do not share preloaded keys, and thus need to establish path keys. Consequently, the PIKE scheme involves relatively high communication overhead, making it unsuitable for large sensor networks.

1.1 Our Work

We present GKE, an efficient **G**roup-based **K**ey **E**stablishment scheme to establish pairwise keys between each pair of neighboring sensors in a large sensor network. As in previous work [15, 6, 11], we use the fact that sensors are often deployed in groups, so that sensors in the same group are more likely to be neighbors. In our scheme, each sensor will be preloaded with unique pairwise keys shared with all other sensors in the same group. This is feasible, with a reasonable group size (see Section 4.2.1). For every pair of neighboring sensors from different groups, we use a technique for path key establishment that requires only local communication. Each path key establishment involves at most two intermediate nodes, each of which is selected from *multiple candidates*.

The uniqueness of pairwise keys allows GKE security to degrade gracefully as the number of compromised sensor increases, hence significantly improving the resilience against node compromise over RKP or SRKP. Further, in GKE, the communication required for a path key establishment is localized to two adjacent groups. Therefore, as our analysis shows, GKE significantly reduces the communication overhead compared to PIKE, which requires network-wide communication for path key establishment.

Previous work [6, 11] typically assumes that the group adjacencies are known prior to sensor deployment. Our scheme is more flexible, and can be applied to deployment models that violate that assumption. Our scheme has the following salient features:

- *Density and distribution independence*: GKE establishes pairwise keys between any pair of neighboring sensors, regardless of sensor density or distribution. It is more general than RKP or SRKP, which require uniform sensor deployment with high density.
- *Graceful resilience degradation*: GKE degrades gracefully as the number of compromised sensors increases. Even with a large fraction of nodes compromised, only a small fraction of secure links are compromised in the rest of the sensor network. GKE provides stronger resilience against node capture than random key predistribution schemes.
- *Localized communication*: Pairwise key establishment between sensors pairs in GKE requires no communication when the sensors are from the same group, and only local communication when they are in neighboring groups.
- *Low memory requirements*: For a network of n sensors, with group size γ, GKE requires each sensor to be preloaded with $\lceil \frac{1}{2}(\gamma - 1) \rceil + \lceil \frac{(n-\gamma)t}{2\gamma^2} \rceil$ keys. If $n = 10,000$, $\gamma = 100$ and $t = 10$, we need preload each sensor with around 55 keys. If $n = 100,000$, we need only 75 keys, showing that our method scales very well to very large sensor networks.
- *Flexible deployment models*: GKE is applicable to more flexible group deployment models. (See Section 3)
- *Mobility support*: GKE is directly applicable both when sensor networks are static or when sensor groups move in swarm fashion.

The rest of this paper is organized as follows. Section 2 motivates our work. We present two flexible deployment models in Section 3 and present our solution in Section 4. In Section 5, we describe the metrics that we use to evaluate the security and performance. We analyze the security of our scheme in Section 6, and evaluate its performance in Section 7. Finally, we make a conclusion in Section 8.

2. MOTIVATION

We now motivate our approach more fully with a discussion of issues in sensor networks.

2.1 High Density Deployment with Uniform Distribution

RKP and SRKP require sensor deployments to form connected graphs, since they can not guarantee key establishment between neighbors without the use of path keys.

As indicated in [3, 12], both RKP and SRKP require high density deployment. Let the sensor density δ be the average number of sensors deployed in a sensor's transmission range. RKP and SRKP require high δ to ensure that the number of edges in the key sharing graph exceeds the Erdös-Renyi threshold for ensuring connectivity [8]. For the random key predistribution scheme in [9], Figure 1 uses the analysis in [9] to plot the sensor density against the memory requirement, when a $10,000$-sensor network is to be securely connected with a probability of 99%. For the configuration as in [9], where the key pool size ω is $100,000$, $\delta \geq 42$.

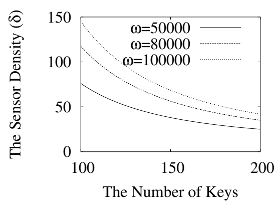
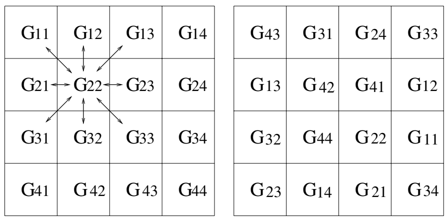


Figure 1: Density vs. memory

Further, as pointed in [3], in a network with non-uniform density, the RKP or SRKP could result in network partition, when a few critical pairs of nodes fail to perform key establishment.

High-density deployments are expensive. Consequently, current random key predistribution schemes may be unsuitable when cost



(a) Groups defined by Huang et al.

(b) RRGD

Figure 2: Group-based deployment methods

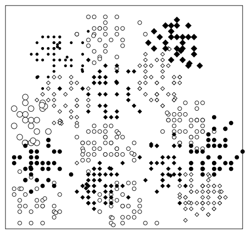


Figure 3: PRGD Model.

is a factor. In contrast, our GKE scheme ensures that every pair of neighboring sensors can establish a unique pairwise key, regardless of the density or the distribution of sensors, as long as the underlying network is physically connected.

2.2 Degradation of Resilience

As the work in [11] shows, the compromise of a mere 100 out of $10,000$ sensors in the original RKP scheme [9] leads to the compromise of around 30% of the secure links between uncompromised sensors. The SRKP scheme [7] does have the nice property that when the number of compromised sensors is below a threshold, the links between uncompromised sensors remain secure. However, the security of these other links decreases dramatically when the number of compromised links exceeds a threshold. This threshold can be as low as 140 for a network of $10,000$ sensors [11].

The resilience of RKP and SRKP degrades so dramatically because each key or key space is shared across a subset of sensors. Once a key or key space is compromised, the security of all other sensors which have this key or key space is weakened. We might expect clever adversaries to maximize their benefit by attacking sensors whose key subsets contain the fewest number of keys already compromised. RKP and SRKP face a dilemma in this respect. They require a high degree of sharing to ensure that the key sharing graph has a high enough edge density, but this requirement compromises resiliency.

3. FLEXIBLE GROUP-BASED DEPLOYMENT METHODS

Since sensors are deployed randomly, the *exact* location of a sensor, or the set of its neighbors will be unknown. However, the choice of a suitable deployment method may help in determining neighborhood relationships to some extent.

Group-based deployment is generally seen as reasonable in a large sensor network [6, 11]. In this approach, sensors are arranged into groups, and each group is deployed as a unit. As a result, sensors from the same group are more likely to be neighbors. A reasonable strategy is to allocate more resources for key establishment between sensors in the same group, and to provide an efficient mechanism for establishing pairwise keys between neighboring sensors from different groups.

Current group-based deployment models typically divide the target region into sub-regions that may [6] or may not [11] overlap, and require each group of sensors deployed into a *predetermined* sub-region.

Figure 2(a) illustrates the group-based deployment approach adopted by Huang et.al in [11]. The target deployment region is divided into 4×4 sub-regions R_{ij} $(1 \leq i, j \leq 4)$. The n sensors are accordingly arranged into 16 subgroups G_{ij} $(1 \leq i, j \leq 4)$. Sensors in Group G_{ij} are deployed into the sub-region R_{ij}. Sensors in group G_{22} have neighbors either in group G_{22} or in group $G_{11}, G_{21}, G_{31}, G_{32}, G_{33}, G_{23}, G_{13}, G_{12}$.

In this deployment method [6, 11], each group of sensors has eight predetermined adjacent groups. It is hence relatively easy for a sensor to establish a pairwise key with a neighboring node which is either in its own group, or in the eight predetermined adjacent groups. However, this deployment method is quite inflexible, and makes the following assumptions.

- Sub-regions are predefined. For example, each sub-region in [11] is a cell in a $k \times k$ grid.
- Each sensor group is deployed into a *specified* sub-region. For example, G_{ij} may have to be deployed into R_{ij}.

These assumptions will not hold in demanding or hostile environments, such as battlefields or emergencies. When sensors are scattered from an airplane in a battlefield, for example, it is unwise for the airplane fly in a predefined (and predictable) deployment pattern. Even if sensors are dropped from missiles, wind and weather can make it impossible for deployments to follow predetermined patterns.

We propose two more flexible deployment models. The first model, referred to as **R**andom **R**egion **G**roup **D**eployment (RRGD), only assumes that sub-regions are predefined, but allows a sub-region to receive any sensor group. RRGD provides more deployment flexibility, making it suitable for a wider variety of applications. Figure 2(b), illustrates one instance of sensor deployment under the RRGD model.

Our second model, which we call **P**ure **R**andom **G**roup **D**eployment (PRGD), relaxes both the assumptions made in [6, 11]. In PRGD, a sub-region could be *anywhere* in the target region. PRGD represents the most general group-based deployment model. Figure 3 shows one deployment of n sensors which are arranged in 16 groups, under PRGD.

Sensors might not be evenly deployed throughout the entire region under the PRGD model. However, this is an artifact of the deployment method, and is orthogonal to the performance of our key distribution scheme. Our work focuses on pairwise key establishment, and works under any group-based deployment method.

It is more challenging to establish pairwise keys for neighboring sensors from different groups under RRGD or PRGD, since it is difficult to predetermine group adjacencies. We next show how GKE addresses this problem.

4. THE GKE SCHEME

Let there be n sensors, and let sensor s_i have ID i, $1 \leq i \leq n$. We arrange these sensors into g groups $G_i, 1 \leq i \leq g$, each of

Notation	Description
s_i	a sensor with identity i
G_u	a group with identity u
n	the network size
γ	the group size
g	the number of groups
δ	the average number of sensors in a sensor's transmission range
K_{ij}	the unique pairwise key shared between s_i and s_j
m	the number of preloaded keys that a sensor shared with sensors that are in the different groups
t	the number of agents (see Definition 1 in Section 4.2.2) for a group in every other group

Table 1: Our Notation

which has $\gamma = n/g$ sensors. Group G_u will comprise sensors with IDs i such that $(u-1)\gamma < i \leq u\gamma$.

Let $\langle G_u, s_i \rangle$ denote sensor s_i from group G_u. We will replace $\langle G_u, s_i \rangle$ by s_i, when no confusion can arise.

4.1 Outline of GKE

The central idea in GKE is to preload each sensor with a carefully chosen set of keys, each shared pairwise with one other sensor. We exploit the fact that unique pairwise keys have perfect resilience against node captures.

We say that two sensors are *associated* if they share a *preloaded* pairwise key. We preconfigure each sensor so that it is associated with every other sensor in its own group. We also ensure that each sensor is associated with sensors from one or more other groups, in a pattern designed to ensure several sensor associations across each pair of groups.

A sensor s_i can now establish a unique pairwise key with any of its neighbors s_j. If s_i and s_j are from the same group, they start off associated. If they are from different groups, there will exist *multiple* associations between their groups, so they can establish a pairwise key using any pair of these associated nodes as intermediaries. This process involves only localized communication, which differentiates our scheme from PIKE [3].

4.2 Key predistribution

We now present the key predistribution schemes in GKE.

4.2.1 Intra-group Key predistribution

We preload each pair of sensors from the same group with a unique pairwise key. This strategy is feasible even with limited memory if we choose the group size γ appropriately. For example, let the group size γ be 100 as in [6, 11], so that each sensor must store 99 keys. If the key size is 64 bits, each sensor requires 792 bytes. This is doable for a Mica2 Mote sensor that has $4KB$ SRAM [19]. We can further halve this memory requirement buying using the techniques adopted in [3], so that allocating 396 bytes for keys suffices to ensure that any pair of neighboring sensors from the same group share a unique preloaded key.

4.2.2 Inter-group Key Predistribution

We begin with a few definitions.

DEFINITION 1 (AGENT). *$\langle G_u, s_i \rangle$ is called an agent for G_v in G_u, if $\langle G_u, s_i \rangle$ is associated with some $\langle G_v, s_j \rangle$ in G_v.*

DEFINITION 2 (t-ASSOCIATED GROUP). *Groups G_u and G_v are said to be t-associated if they have t agents for each other.*

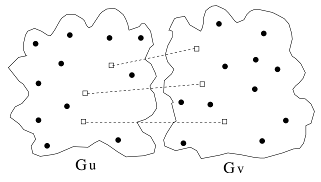


Figure 4: G_u and G_v are 3-associated.

Since each pair of agents share a pairwise key, neighboring sensors from groups G_u and G_v can establish path keys using any pair of agents as intermediaries. As group adjacencies are unknown prior to sensor deployment under RRGD and PRGD, the problem of key establishment between sensors in different groups reduces to that of creating group associations.

We will require each group to be associated with every other group. If there are g groups, and each sensor has enough memory to hold m inter-group pairwise keys, each group can have up to $t = \lceil \frac{m\gamma}{g-1} \rceil$ agents in each of the other groups. Algorithm 1 shows how to define group associations. We use functions $\mathcal{F}_i$ ($1 \leq i \leq t$) which uniformly map group pairs from $[1, g] \times [1, g]$ to $[1, n]$. $\mathcal{F}_i(G_u, G_v)$ selects the ith agent for G_v in G_u, and is defined as follows

$$\mathcal{F}_i(G_u, G_v) = \big(t(v-1) + i\big) \pmod{\gamma} + (u-1)\gamma.$$

G_u comprises the sensors s_i with $(u-1)\gamma < i \leq u\gamma$. Hence $\mathcal{F}_1(G_u, G_v), \cdots, \mathcal{F}_t(G_u, G_v)$ select t sensors, with IDs between $\big(t(v-1)+1\big) \mod \gamma + (u-1)\gamma$ and $tv \mod \gamma + (u-1)\gamma$ as agents for G_v.

Algorithm 1 Inter-group key predistribution

```
$t = \lceil \frac{m\gamma}{g-1} \rceil$
**for** each pair of groups $G_u, G_v$ **do**
  **for** $i = 1$ to $t$ **do**
    $s_x = \mathcal{F}_i(G_u, G_v)$
    $s_y = \mathcal{F}_i(G_v, G_u)$
    assign a unique pairwise key to $s_x$ and $s_y$
  **end for**
**end for**
```

Algorithm 1 has the following attractive features:

- *Uniformity*: Each sensor is agent for the same number of groups. This balances loads and creates no high-value targets, since no sensor holds more keys than any other.
- *Resilience*: Multiple agents improve resilience for establishing path keys.
- *Easy agent discovery*: Agents can be discovered using the functions $\mathcal{F}_1, \cdots, \mathcal{F}_t$, rather than by lookups.

Figure 5 shows the inter-group key predistribution for sensors in group G_{22}. For simplicity, we only show the scenario when each group pair has one agent pair. Accordingly, each sensor is required to be preloaded with $m = 2$ keys shared with sensors in distinct groups.

4.3 Pairwise Key Establishment

4.3.1 Intra-group Key Establishment

A unique pairwise key is preloaded for every intra-group sensor pair.

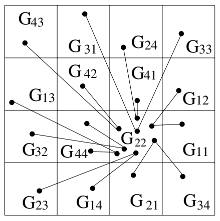


Figure 5: Inter-Group Key Predistribution for G_{22} ($t = 1$)

4.3.2 Inter-group Key Establishment

We adopt the Highest Random Weight technique [20] to choose agents for path key generation, using a hash function $\mathcal{H}$ to realize distributed agreement. The work in [20] discusses how to select $\mathcal{H}$. Sensors $\langle G_u, s_i \rangle$ and $\langle G_v, s_j \rangle$ generate a path key as follows.

1. One principal, say $\langle G_u, s_i \rangle$, first computes $\mathcal{H}(s_i, s_j, p)$ for $1 \le p \le t$, and selects the p that yields the biggest $\mathcal{H}$ value. It now uses the function $\mathcal{F}_p$ to pick an associated sensor pair $\langle G_u, s_x \rangle$ and $\langle G_v, s_y \rangle$ for path key generation. Now, s_i then randomly generates a key K_{ij} and sends it to agent s_x, encrypted with the association key K_{ix} it shares with s_x.
$$s_i \rightarrow s_x : (K_{ij}, G_v)_{K_{ix}}$$
2. Upon receipt, s_x decrypts this message and re-encrypts it with the association key K_{xy} it shares with s_y, and sends it to s_y.
$$s_x \rightarrow s_y : (K_{ij})_{K_{xy}}$$
3. s_y decrypts this packet, re-encrypts it with the key K_{jy} it shares with s_j, and sends it to s_j.
$$s_y \rightarrow s_j : (K_{ij})_{K_{jy}}$$
4. s_j first applies $\mathcal{H}$ to select the same associated pair $\langle G_u, s_x \rangle$ and $\langle G_v, s_y \rangle$ that s_i selected for path key establishment. It then recovers K_{ij} using K_{jy}, its preloaded association key with s_y.

Features of GKE

- *Resilience to impersonation:* All messages above are secured with the preloaded pairwise keys shared between the sender and the receiver. It is hence impossible for an attacker to impersonate the intermediaries, since it does not have the preloaded keys.
- *Failure resilience:* We can use the techniques in [20] to guarantee resilience. Each pair of groups has $t = \lceil \frac{m\gamma}{g-1} \rceil$ agent pairs. If there are $n = 10,000$ sensors arranged into $g = 100$ groups, and each sensor is preloaded with $m = 80$ pairwise keys shared with sensors in other groups, we will get $t = 80$ agent pairs for each pair of groups. Let a pair of intermediaries be selected for a path key request using function $\mathcal{H}$. If this agent pair fails, we simply select the pair corresponding to the index q that yields the second biggest $\mathcal{H}$ value, and use $\mathcal{F}_q$ to determine the new agent pair for path key generation. We can continue until we find an agent pair that is alive.
- *Routing protocol:* Routing is an issue orthogonal to our work. PIKE uses the geographic routing protocol GPSR [13] with a globally addressable infrastructure GHT [17] to find routes to the intermediate nodes. GKE can also use GPSR and GHT to find the route either from a sensor s_i to the agent s_x, or between the agents s_x and s_y.

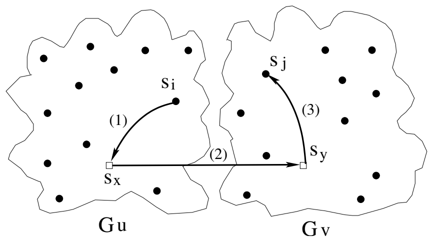


Figure 6: Inter-Group Key Establishment

However, there is a major difference between PIKE and GKE in this respect. Finding a route to the trusted intermediary nodes in PIKE involves network-wide routing discovery, since these intermediaries may not always be in the vicinity. In contrast, since sensor s_i and agent s_x are within the same group in GKE, discovering a route to the agent only involves route discovery within the group. Route discovery between s_x and s_y is also local since they are in adjacent groups. GKE can accomplish key establishment even without a globally addressable infrastructure. The overhead of routing in GKE is much smaller than that in PIKE.

4.4 Supporting Node Mobility

Consider a type of mobile sensor networks where groups of sensor move in a swarm fashion [23]. Previous group-based key predistribution schemes [6, 11] cannot adapt to group mobility, since they assume that group adjacencies are predetermined and preserved. When groups of sensors move, the neighborhood relationships between these groups cannot be maintained. In contrast, every pair of groups is associated in GKE, and sensors from associated groups are always able to establish a shared key. Group mobility is not a problem.

5. EVALUATION METRICS

We evaluate GKE in terms of security and performance. We measure security in terms of resilience against node capture and connectivity, and measure performance in terms of communication and memory overhead.

5.1 Security Metrics

5.1.1 Resilience

This metric measures how the capture of some sensors affects the security of the *rest* of the network. Let U be the set of uncompromised sensors. Let $L(U)$ and $\hat{L}(U)$, respectively, be the sets of total and compromised links between sensors in the set U. Resilience is defined as the ratio $\hat{L}(U)/L(U)$.

This definition of resilience is similar to those used in previous random key predistribution schemes [9, 4, 7, 14, 6, 11]. However, our meanings of *link* is different. In our definition, a link exists between every pair of neighboring sensors, and is secured either by a preloaded pairwise key or by a path key. In contrast, the previous schemes consider only the links between pairs of neighboring sensors secured by preloaded keys [9, 4, 6] or keys derived from preloaded key space [7, 14, 11].

As [7] points out, a path key is compromised if an attacker can decipher the messages during key establishment or compromise any of the intermediaries. It is hence important to consider the security of path keys to properly evaluate the effects of sensor compromise. In Section 6.1, we present some analytical and simulation results of resilience, considering the security of both preloaded keys and path keys.

5.1.2 *Connectivity*

Connectivity is defined as the probability that a sensor network is securely connected. In Section 6.2, we show that the GKE scheme can enable a sensor network securely connected with 100% probability, as long as the network is physically connected.

5.2 Performance Metrics

5.2.1 *Communication Overhead*

We compare the communication overhead of GKE with that of PIKE, since only GKE and PIKE show graceful security degradation as the number of compromised sensors increases (see Section6.1). For both GKE and PIKE, we only measure the communication overhead of transmitting the encrypted path keys among the sensors, neglecting the routing communication overhead. First, as indicated in [3], the routing communication highly depends on the underlying routing protocol, which is out of the scope of our paper. Second, as analyzed in Section 4.3.2, with the same routing protocol, GKE will introduce smaller routing communication overhead than PIKE. Therefore, neglecting the communication overhead of routing for both GKE and PIKE does not favour our scheme GKE in any aspect, when compared to PIKE. Rather, such processing will help us focus on the efficiency of key establishment technique.

More specifically, the communication overhead is measured as the average number of hops that a message has to be transmitted in order to establish a key between any pair of neighboring sensors.

5.2.2 *Memory Overhead*

As in earlier schemes, we quantify memory overhead in terms of the number of keys preloaded into each sensor. We do not count the temporary storage during the execution of our scheme, or the memory to store the newly established pairwise keys.

5.3 System Setting

We used the following configuration in our analysis and simulations. The size of the sensor network n varied between 10,000 and 50,000, with $10,000$ being the default value. The wireless communication range for each sensor was $40m$. The deployment density δ, the average number of sensors in a sensor's transmission range, varied from 20 to 100, to represent low- to high-density deployments. The deployment area A is determined by the network size n, sensor density δ, and the communication range. $A = \frac{n\pi r^2}{\delta}$. The group size γ was set to be 100 as previous group-based schemes [6, 11], and the number of groups varied from 100 to 500 accordingly. For simplicity, we assume that sensors in each group were uniformly distributed within a region of area A/g.

6. SECURITY ANALYSIS

In this section, we compare the security of GKE with SRKP [7], the group-based random key predistribution scheme in [6], and PIKE [3] in terms of resilience against node capture and connectivity.

6.1 Resilience

Let s_i and s_j be two uncompromised neighbors. Let L_{ij} be the communication link between them, and let K_{ij} be the key used to secure this link. Let $\mathbf{\Lambda(K_{ij})}$ be the event that K_{ij} is a preloaded key, and let $\mathbf{\Pi(K_{ij})}$ be the event that K_{ij} is a path key. Let $\mathbf{\bar{L}_{ij}}$ be the event that link L_{ij} is compromised, and $\mathbf{C(x)}$ be the event that x sensors have been compromised. The probability that $\mathbf{\bar{L}_{ij}}$ has occurred given that x sensors have been compromised is

$$\begin{aligned}\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)}] &= \Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Lambda(K_{ij})}] \times \Pr[\mathbf{\Lambda(K_{ij})}] \\ &+ \Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi(K_{ij})}] \times \Pr[\mathbf{\Pi(K_{ij})}].\end{aligned}$$

Schemes such as [9, 4, 7, 14, 6, 11] consider only the links secured by preloaded keys in evaluating resilience. Since pairwise keys are established by randomly selecting them from a global pool, the compromise of one sensor may compromise a number of pairwise keys for other sensors.

This is impossible in GKE since preloaded pairwise keys are unique. A link secured by a preloaded key can not be compromised unless one of its endpoints is compromised. Therefore, GKE achieves *perfect* resilience against node capture by their definition.

By our definition of resilience, for GKE,

$$\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)}] = \Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi(K_{ij})}] \times \Pr[\mathbf{\Pi(K_{ij})}].$$

Now, $\Pr[\mathbf{\Pi(K_{ij})}]$ is simply the ratio of the number of path keys to the total number of keys among all pairs of neighboring sensors. Let $\mathbf{\Pi_2(K_{ij})}$ be the event that the path key K_{ij} is generated using two agents, and $\mathbf{\Pi_1(K_{ij})}$ be the event that the path key K_{ij} is generated using a single agent, as in the case when s_i or s_j is itself the agent for the other's group. Now,

$$\begin{aligned}&\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)}] = \\ &\big(\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi_1(K_{ij})}] \times \Pr[\mathbf{\Pi_1(K_{ij})}] \\ &\;+ \Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi_2(K_{ij})}] \times \Pr[\mathbf{\Pi_2(K_{ij})}]\big) \times \Pr[\mathbf{\Pi(K_{ij})}].\end{aligned}$$

Let there be g groups each of size γ, and let each sensor hold m preloaded keys for sensors in other groups. As shown in Section 4.3.2, each group has $t = \frac{m\gamma}{g-1}$ agents in every other group. If α is the probability that either s_i or s_j is the agent of its neighbor's group, then

$$\alpha = \frac{\binom{\gamma-1}{t-1}}{\binom{\gamma}{t}} = \frac{t}{\gamma}$$

$\Pr[\mathbf{\Pi_1(K_{ij})}]$ is equivalent to the probability that one endpoint of the link L_{ij} is the agent for the other group, while the other endpoint is not. Thus we get

$$\Pr[\mathbf{\Pi_1(K_{ij})}] = 2\alpha(1-\alpha).$$

Similarly, $\Pr[\mathbf{\Pi_2(K_{ij})}]$ is equivalent to the probability that neither s_i nor s_j is an agent for the other group, and can be computed as

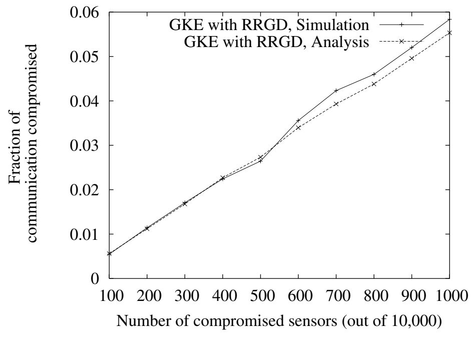
$$\Pr[\mathbf{\Pi_2(K_{ij})}] = (1-\alpha)^2.$$

Now, $\binom{n-3}{x}/\binom{n-2}{x}$ is the probability that the agent used to transmit the path key K_{ij} is not compromised, when s_i and s_j are uncompromised, but x other sensors are compromised. Thus $\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi_1(K_{ij})}]$ can be computed as

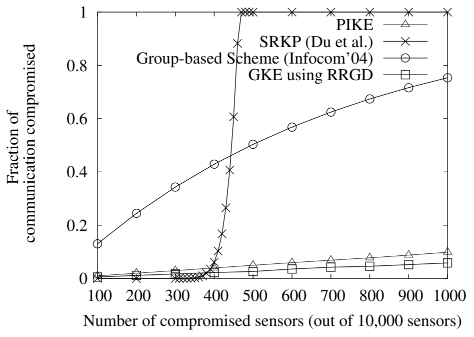
$$\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi_1(K_{ij})}] = 1 - \frac{\binom{n-3}{x}}{\binom{n-2}{x}} = \frac{x}{n-2}.$$

Similarly,

$$\begin{aligned}\Pr[\mathbf{\bar{L}_{ij}} |\, \mathbf{C(x)} \wedge \mathbf{\Pi_2(K_{ij})}] &= 1 - \frac{\binom{n-4}{x}}{\binom{n-2}{x}} \\ &= 1 - \frac{(n-2-x)(n-3-x)}{(n-2)(n-3)}.\end{aligned}$$



(a) GKE Resilience: Analysis vs. Simulation



(b) SRKP, a group-based scheme, PIKE and GKE Resilience

Figure 7: Fraction of Compromised Links among Uncompromised Sensors (n =10,000, $\delta = 50$)

Therefore,

$$\Pr[\bar{\mathbf{L}}_{\mathbf{ij}} \mid \mathbf{C}(\mathbf{x})] =$$
$$\big((1-\alpha)^2\big(1 - \frac{(n-2-x)(n-3-x)}{(n-2)(n-3)}\big) + 2\alpha(1-\alpha)\big(\frac{x}{n-2}\big)\big)$$
$$\times \Pr[\mathbf{\Pi}(\mathbf{K}_{\mathbf{ij}})].$$

$\Pr[\mathbf{\Pi}(\mathbf{K}_{\mathbf{ij}})]$, the ratio of the number of path keys to the total number of keys among all pairs of neighboring sensors, is dependent on sensor deployment. Based on our simulation results, Figure 13(b) plots $\Pr[\mathbf{\Pi}(\mathbf{K}_{\mathbf{ij}})]$ in GKE under the RRGD and PRGD models.

Figure 7(a) shows that our analytical and experimental results for the number of compromised links match each other closely.

Figure 7(b) compares the resilience of GKE with that of SRKP [7], the group-based scheme in [6], and PIKE [3]. The resilience of SRKP is derived based on the analysis in [7], and each sensor is preloaded with 200 keys drawn from 4 key spaces randomly chosen from 50 key spaces. The resilience of the group-based scheme in [6] is derived based on their analysis with a key space size of 100,000 and connectivity of 99.99% [6]. (The connectivity of GKE is 100%. See Section 6.2.) We note that their analysis only consider links secured by preloaded keys, and hence the fraction of compromised links should be higher if the security of path keys is also considered. For GKE and PIKE, we consider the security of both preloaded keys and path keys. If only the links secured by preloaded keys were considered, the resilience graphs of PIKE and GKE would both be lines of zero, representing perfect resilience against node capture.

Figure 7(b) shows that when around 350 of 10,000 sensors are compromised, the resilience of SRKP decreases dramatically. It also shows that when around 400 of 10,000 sensors are compromised in [6], 43% of the links among uncompromised sensors will be compromised. In contrast, both PIKE and GKE show graceful degradation of resilience with respect to the number of compromised nodes, so that attackers are unable to compromise a large fraction of other communication links by compromising a small number of sensors.

Although both GKE and PIKE shows graceful degradation of resilience, Figure 7(b) shows that the resilience of GKE is about twice as high as that of PIKE. In Section 7.2, we will show that this improvement of resilience is achieved with even significantly smaller communication overhead, compared to that of PIKE.

6.2 Connectivity

As shown in Section 2.1, RKP and SRKP require high density deployments to ensure the entire sensor network is securely connected with high probability. In contrast, GKE ensures that any two neighboring sensors are able to establish a path key, regardless of the sensor density or distribution, as long as the sensor network is physically connected. This guarantee is achieved since (1) any pair of sensors from the same groups have preloaded pairwise keys, (2) sensors from associated groups are able to establish path keys, and (3) the inter-group key predistribution scheme ensures that any two groups are t-associated (see Section 4.2.2).

Figure 8 compares the connectivity of SRKP, PIKE and GKE in a sensor network with 10,000 sensors. For SRKP, each sensor has 4 key spaces chosen from a pool of 50 key spaces, and preloaded with 200 keys. This is a typical configuration from [7].

As Figure 8 clearly shows, the connectivity of SRKP decreases dramatically when the sensor density is less than 50, and is almost surely disconnected when the density is around 25. In contrast, PIKE and GKE retains full connectivity regardless of sensor density. Remarkably, only 55 keys are required for the GKE scheme to achieve full connectivity, when any pair of groups are 10-associated (See Section 7.1).

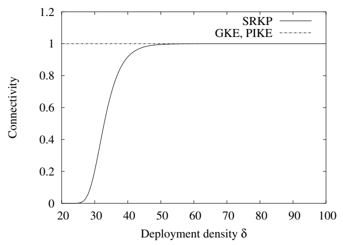


Figure 8: SRKP, PIKE and GKE Connectivity

7. PERFORMANCE EVALUATION

As shown in Section 6, PIKE and GKE have substantially better resilience against node compromises than random key predistribution schemes, and guarantee that any two neighbors can establish a path key if needed. We now compare PIKE and GKE in terms of memory and communication overhead.

7.1 Memory Overhead

The GKE scheme imposes low memory requirements. For a sensor network of n sensors, with group size γ, GKE requires each

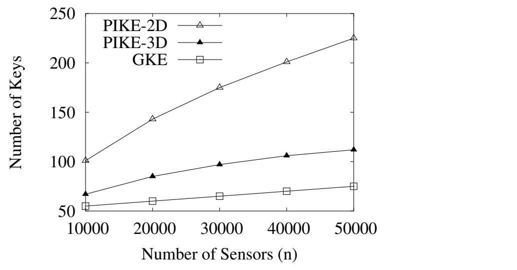


Figure 9: Memory Requirements in PIKE-2D, PIKE-3D and GKE (t=10).

sensor to be preloaded with $\gamma - 1$ pairwise keys shared with sensors from the same group and $t(g-1)/\gamma$ pairwise keys shared with sensors that are in the different groups. Further, we use the method in [3] to reduce the memory requirement by a factor of two. Therefore, the total memory overhead per sensor is $\lceil \frac{1}{2}(\gamma - 1)\rceil + \lceil \frac{(n-\gamma)t}{2\gamma^2} \rceil$ keys.

In contrast, the memory overheads of PIKE-2D and PIKE-3D are $\lceil \sqrt{n} \rceil + 1$ and $3\lceil \sqrt[3]{n} \rceil + 1$ separately [3]. Figure 9 shows the memory requirements of PIKE-2D, PIKE-3D and GKE separately.

7.2 Communication Overhead

Messages are transmitted in GKE only for path key establishment. Let H denote the the average number of hops that a message traverses when any path key K_{ij} is established. Therefore, the average communication overhead is simply $H \times \Pr[\mathbf{\Pi}(\mathbf{K_{ij}})]$.

Two major differences between PIKE and GKE result in a big difference in their communication overheads. First, nodes use local intermediaries when establishing path keys in GKE, so only local communication is needed to transmit key establishment messages. In contrast, intermediaries in PIKE could be *anywhere* in the entire target region, so that network-wide communication is required.

Second, the fractions of keys that are path keys is much higher in PIKE than in GKE. Sensors are deployed in groups in GKE, so that sensors from the same group are more likely to be neighbors. In GKE, all pairs of sensors from the same group are preloaded with pairwise keys. In PIKE, only sensors on the same grid column or row share preloaded pairwise keys. No deployment knowledge can be predetermined on constructing the grid makes the fraction of path keys in PIKE significantly higher than that of GKE.

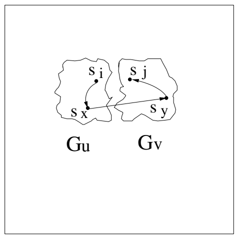
7.2.1 Communication Overhead for Path Key Establishment

Establishing a path key between s_i and s_j in GKE (see Figure 10(a)) requires messages from s_i to s_x, from s_x to s_y, and from s_y to s_j. If $h(s_p, s_q)$ denotes the hop distance between s_p and s_q, the number of hops required for path key establishment is $H(s_i, s_j) = h(s_i, s_x) + h(s_x, s_y) + h(s_y, s_j)$. If H_{GKE} is the expected number of hops for path key establishment in GKE, linearity of expectation leads to

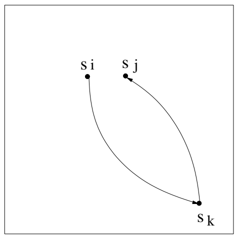
$$H_{GKE} = 2 * \bar{h}_{GKE} + \bar{h'}_{GKE},$$

where $\bar{h}_{GKE}$ is the expected hop distance between any two nodes in a group, and $\bar{h'}_{GKE}$ is the expected hop distance between any two nodes from adjacent groups.

Establishing a path key between neighboring sensors s_i and s_j in PIKE (Figure 10(b)) includes the round trip from the neighboring sensors to the intermediary s_k, who may be anywhere in the region. The number of hops required is $h(s_i, s_k) + h(s_k, s_j)$. If $\bar{h}_{PIKE}$ is the average hop distance between any two nodes in the entire region, the expected communication overhead in PIKE is



(a) Path Keys in GKE



(b) Path Keys in PIKE

Figure 10: Path Key Establishment in GKE and PIKE

$$H_{PIKE} = 2 * \bar{h}_{PIKE}$$

Next, we give lower bounds for H_{PIKE} and for H_{GKE} under the RRGD deployment model. If two nodes are separated by physical distance $\bar{\lambda}$, we will need at least $\bar{\lambda}/r$ hops, where r is the transmission radius. Therefore, we can use $\bar{\lambda}/r$ as a lower bound for the average hop distance.

In PIKE, the expected physical distance between s_i and s_k is the expected distance between two randomly picked points in a square of area A, and is known [10] to be $\bar{\lambda}_{PIKE} = 0.52\sqrt{A}$.

In GKE, each group of sensors is in a square of size of $a \times a$, where $a = \sqrt{A/g}$. Since s_i and s_x are in the same square, the expected physical distance between them [10] is $\bar{\lambda}_{GKE} = 0.52a$. To get $\bar{\lambda'}_{GKE}$, the expected distance between s_x and s_y, we note that two adjacent squares may touch (see Figure 11) along an edge or at a corner. Let $\bar{\lambda'}_{\oplus}$ be the expected distance between two random points picked randomly from neighboring squares that are vertically (or horizontally) disposed. Let $\bar{\lambda'}_{\otimes}$ be the expected distance between two random points picked from neighbors that are diagonally disposed. Clearly, we now have $\bar{\lambda'}_{GKE} = \Pr[\bigoplus]\bar{\lambda'}_{\oplus} + \Pr[\bigotimes]\bar{\lambda'}_{\otimes}$, where $\Pr[\bigoplus]$ (or $\Pr[\bigotimes]$) is the probability that neighboring squares are horizontally or diagonally disposed, respectively.

It is known [10] that the expected distance between two random points in an $a \times 2a$ rectangle is $0.804a$. Since these two points are from the same square half with probability 0.5 and from different square halves with probability 0.5, we can use linearity of expectation to get $0.804a = 0.5\bar{\lambda'}_{\oplus} + 0.5\bar{\lambda}_{GKE}$. That is, $\bar{\lambda'}_{\oplus} = 1.088a$.

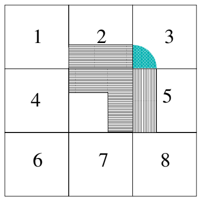


Figure 11: Adjacencies

To get $\bar{\lambda'}_{\otimes}$, consider two random points in a $2a \times 2a$ square, which consists of four $a \times a$ squares. Clearly, the expected distance between two random points in a $2a \times 2a$ square is $0.52 \times 2a = 1.04a$. Since these two points are from the same $a \times a$ square with prob-

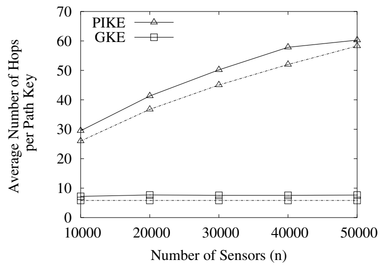
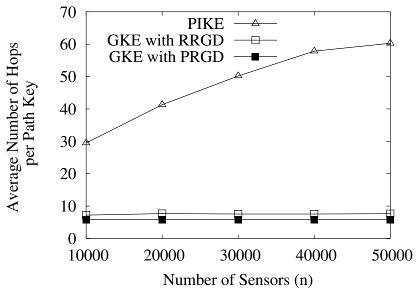
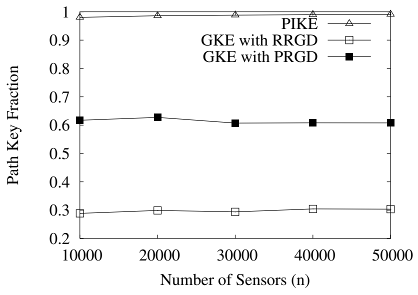


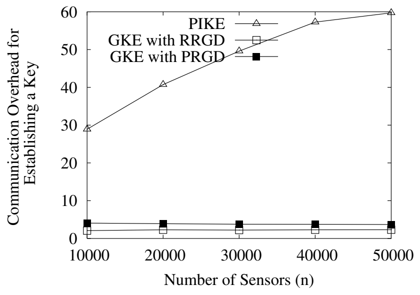
Figure 12: H_{GKE} and H_{PIKE}: Analysis vs. Simulation



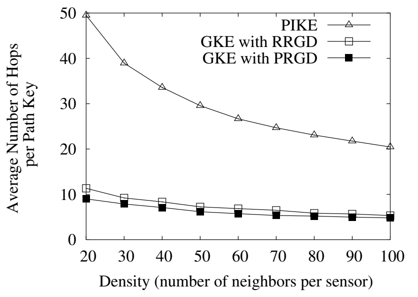
(a) Avg Number of Hops vs. Network Size ($\delta = 50$)



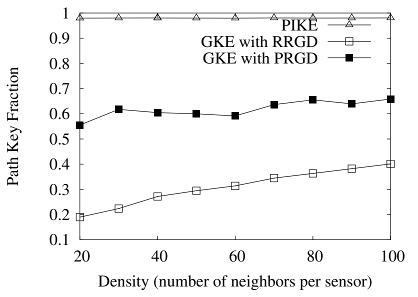
(b) Path Key Fraction vs. Network Size ($\delta = 50$)



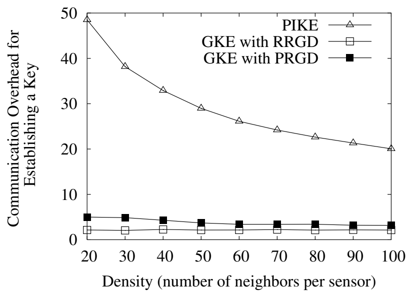
(c) Communication Overhead vs. Network Size ($\delta = 50$)



(d) Avg Number of Hops vs. Density (n =10,000)



(e) Path Key Fraction vs. Density (n =10,000)



(f) Communication Overhead vs. Density (n =10,000)

Figure 13: Average Number of Hops, Path Key Fraction and Communication Overhead.

ability 0.25, from two horizontally or vertically adjacent $a \times a$ squares with probability 0.5, and from two diagonally disposed $a \times a$ squares with probability 0.25, we can use linearity of expectation to write $1.04a = 0.25\bar{\lambda}_{GKE} + 0.5\bar{\lambda'}_{\oplus} + 0.25\bar{\lambda'}_{\otimes}$. Since we know $\bar{\lambda}_{GKE} = 0.52a$, $\bar{\lambda'}_{\oplus} = 1.088a$, we get $\bar{\lambda'}_{\otimes} = 1.464a$.

We estimate Pr[$\bigoplus$] and Pr[$\bigotimes$] as follows. s_i will need to communicate with a neighbor s_j from a vertically adjacent square (see Figure 11) only when s_i is within distance r from the top edge of its cell. Similarly, s_i will need to communicate with a neighbor from a diagonally disposed square only when s_i is within distance r from the corner, that is, inside the quarter circle shown. Hence, we will have Pr[$\bigoplus$] = $\frac{ar}{ar+0.25\pi r^2}$, and Pr[$\bigotimes$] = $\frac{0.25\pi r^2}{ar+0.25\pi r^2}$.

Now, $\bar{\lambda'}_{GKE} = \frac{ar}{ar+0.25\pi r^2}\bar{\lambda'}_{\bigoplus} + \frac{0.25\pi r^2}{ar+0.25\pi r^2}\bar{\lambda'}_{\bigotimes}$. This yields $\bar{\lambda'}_{GKE} = \frac{4.35a^2+1.46\pi ra}{4a+\pi r}$. Now,

$$\begin{aligned} H_{GKE} &= 2 * \bar{h}_{GKE} + \bar{h'}_{GKE} \\ &\geq \frac{1.04\sqrt{A/g}}{r} + \frac{4.35A/g + 1.46\pi r\sqrt{A/g}}{(4\sqrt{A/g}+\pi r)r}, \end{aligned}$$

and

$$H_{PIKE} = 2 * \bar{h}_{PIKE} \geq \frac{1.04\sqrt{A}}{r}.$$

In Figure 12, the solid line shows the experimental results and the dashed line shows theoretical lower bound H for PIKE and GKE, using a density $\delta = 50$. For both schemes, the experimental results match the lower bound quite closely. Therefore, we may use this lower bound to approximate H.

Figure 13(a) plots simulation results for the average number of hops H to establish a path key in PIKE and GKE, varying the network size from $10,000$ to $50,000$, for a density of 50. For a fixed group size, H_{GKE} remains constant as the network grows, indicating that network size has no impact on the expected communication overhead. This is because the communication of establishing a path key in GKE is localized to two adjacent groups. In contrast, establishing a path key in PIKE requires network-wide communication, and thus H_{PIKE} increases as the network size increases.

7.2.2 Fraction of Keys Which are Path Keys

Let the path key fraction be the fraction of keys which are path keys. Figure 13(b) shows the path key fraction in PIKE and GKE vs. network size, respectively. Almost all (about 99%) of the links in PIKE are secured by path keys. This is expected, since only sensors at the same column or row of the logical grid have preloaded keys. This logical grid used to predistribute pairwise keys, includes no deployment information, so that sensors sharing preloaded keys are rarely neighbors. In contrast, although deployment information is not available in GKE, sensors in the same group, which are preloaded with pairwise keys shared with one another, are more likely to be neighbors. As a result, GKE has a much smaller path key fraction, around 30% under RRGD and 60% under PRGD. PRGD will have higher path key fraction than RRGD, since the coverage areas of two groups might overlap, resulting more neighboring sensors that are in different groups.

7.2.3 *Communication Overhead*

We plot the communication overhead, which is $H \times \Pr[\mathbf{\Pi}(\mathbf{K_{ij}})]$, in Figure 13(c), for network size from $10,000$ to $50,000$. Clearly, GKE reduces the communication overhead by a factor of about 6 for a network of size $10,000$, with the improvement proportional to the network size. This demonstrates that GKE is especially suitable for very large sensor networks.

7.2.4 *Comparison for Low Density Deployments*

We also compare the communication overhead of GKE with PIKE for low density deployments. Figure 13(d) plots the average number of hops for establishing a path key in PIKE and GKE, for network densities from 20 to 100, for a network size of 10,000. The number of neighbors increases with network density, so the average number of hops decreases in both PIKE and GKE. Again, GKE requires much lower communication overhead to establish path keys than PIKE. Figure 13(e) shows the ratio of path keys in PIKE and GKE, and Figure 13(f) plots the communication overhead in PIKE and GKE, varying the network density from 20 to 100. Clearly, GKE has much lower communication overheads than PIKE even when the network density is low, demonstrating pretty good efficiency.

8. CONCLUSIONS

We have presented GKE, a new group-based key predistribution scheme for large sensor networks. GKE has a number of advantages over current methods. First, it accommodates two very flexible deployment models, RRGD and PRGD. Second, it enables any pair of neighboring sensors to establish a unique pairwise key, regardless of sensor density or distribution, making our scheme suitable for a wide range of applications. Third, GKE is nearly perfectly resilient against node capture attacks, due to the uniqueness of pairwise keys. Unlike SRKP, which also establishes unique pairwise keys, system security in GKE does not degrade dramatically beyond a certain threshold. Instead, GKE is remarkably resilient, and degrades gracefully. Finally, GKE involves only local communication to establish pairwise keys, and has very low communication overhead.

Acknowledgment

This work is supported in part by grants from Tata Consultancy Services, Inc., and the Fault-Tolerant Networks program of Defense Advanced Research Projects Agency, under contract F30602-01-2-0536.

9. REFERENCES

[1] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi. Secure pebblenets. In *MobiHoc*, 2001.

[2] D. Carman, P. Kruus, and B. Matt. Constraints and approaches for distributed sensor network security. In *NAI Labs Technical Report*, 2000.

[3] H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *INFOCOM*, 2005.

[4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, 2003.

[5] W. Diffie and M. Hellman. Multiuser cryptographic techniques. In *AFIPS Conference*, 1976.

[6] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM*, 2004.

[7] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In *ACM Conference on Computer and Communications security (CCS)*, 2003.

[8] Erdos and Renyi. On random graphs. In *I. Publ. Math. Debrecen*, 1959.

[9] L. Eschenaer and V.D.Gligor. A key-management scheme for distributed sensor networks. In *ACM Conference on Computer and Communications security (CCS)*, 2002.

[10] B. Ghosh. Random Distances within a rectangle and between two rectangles. *Bull. Calcutta Math. Soc.*, 43:17–24, 1951.

[11] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2004.

[12] J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2004.

[13] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MOBICOM*, pages 243–254, 2000.

[14] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM Conference on Computer and Communications security (CCS)*, 2003.

[15] D. Liu and P. Ning. Location-based pairwise key establishments of static sensor networks. In *ACM Workshop in Security in Ad Hoc and Sensor Networks*, 2003.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: security protocols for sensor networks. In *ACM MOBICOM*, 2001.

[17] S. Ratnasamy, B. Karp, L. Yin, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *ACM Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[18] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. In *Communications of the ACM*, 1978.

[19] I. C. Technology. Mica2: Wireless measurement system, http://www.xbow.com/product_pdf_files/wireless_pdf/6020-0042-0%4_a_mica2.pdf.

[20] D. Thaler and C. V. Ravishankar. Using name-based mappings to increase hit rates. *IEEE/ACM Transactions on Networking.*, 6(1):1–14, 1998.

[21] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route detection and filtering of injected false data in sensor networks. In *INFOCOM*, 2004.

[22] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *IEEE INFOCOM*, 2004.

[23] B. Zhou, K. Xu, and M. Gerla. Group and Swarm mobility models for AD HOC network scenarios using virtual tracks. In *Military Communications Conference (MILCOM)*, 2004.

[24] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In *ACM Conference on Computer and Communications Security (CCS)*, 2003.