Location-aware Key Predistribution Scheme for Wide Area Wireless Sensor Networks

Katerina Simonova
University of Vermont
33 Colchester Ave.
Burlington, VT 05401
ekaterina.simonova@uvm.edu

Alan C. H. Ling
University of Vermont
33 Colchester Ave.
Burlington, VT 05401
aling@cems.uvm.edu

X. Sean Wang
University of Vermont
33 Colchester Ave.
Burlington, VT 05401
xywang@cs.uvm.edu

ABSTRACT
Key predistribution in wireless sensor networks refers to the problem of distributing secret keys among sensors prior to deployment. Solutions appeared in the literature can be classified into two categories: basic schemes that achieve fixed probability of sharing a key between any pair of sensors in a network and location-aware schemes that use a priori knowledge about sensors' communication needs, such as location information, to guarantee connectivity only among sensors that need to and can talk. Location-aware schemes achieve performance enhancement over the basic schemes by using resources efficiently. However, existing location-aware solutions are not compatible with combinatorial methods that use a set of key groups to generate sensors' key rings. Combinatorial methods are appealing as they achieve deterministic performance close to optimal. Besides, existing location-aware solutions do not have enough flexibility in terms of trade-off between connectivity and resilience. In this paper we propose a general key predistribution framework that can use any key predistribution method as its underlying scheme, including combinatorial ones. The proposed framework provides the user with options on how to allocate available resources to achieve desired performance based on the needs of the application. We also consider heterogeneous sensor networks consisting of nodes with different amount of memory and communication ranges and show that special treatment of this case results in substantial performance improvement. We confirm the good performance of our framework by providing experimental and analytical results.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: Security and Protection

General Terms
Management, Design, Security.

Keywords
Sensor Network, Security, Key Predistribution Scheme.

SASN'06, October 30, 2006, Alexandria, Virginia, USA.

INTRODUCTION
Wireless Sensor Networks (WSN) are envisioned as a promising tool for monitoring the areas where access of human beings is hindered. Their appealing feature is their ad-hoc nature: they do not need any pre-setup communication infrastructure, and once deployed, sensors start collecting data and sending it through peer-to-peer connections that they establish on the fly. WSN could cover geographically vast areas and could consist of thousands of sensor nodes. In order for a WSN to be economically justified, the price of each node should be very low, which implies limited hardware they use and the resources they have. Usually every node has limited computational resources, small amount of memory, and finite power [1]. At the same time such networks are often installed in adverse environments where traffic protection from eavesdropping and modification is essential. But limited resources and dynamic nature of the network make traditional asymmetric cryptographic techniques inapplicable and Key Predistribution Schemes based on private keys that are loaded to sensors prior to deployment are currently viewed as the most promising solution.

A number of Key Predistribution Schemes with different performance balances have appeared in literature. Among the common metrics used to evaluate a Key Predistribution Scheme are connectivity that indicates the likelihood that a pair of sensors would be able to establish secure but short communication path once deployed, and resilience that demonstrates the capability to maintain secure communication when a number of sensors are compromised. With only a small amount of available resources it is impossible to achieve both good connectivity and resilience; trade-offs between these metrics are inevitable. Nevertheless, an attempt should be made to use available resources as efficiently as possible. Any information that could provide the system with a-priori knowledge about sensors communication needs should be carefully considered to distribute keys among sensors in a way that only sensors that need to and can talk share keys. Distributing keys with respect to the communication needs results in significant performance enhancement as every key is used only among small fraction of sensors. As a result, compromise of a sensor has little impact on the whole network. In wide-area sensor networks, deployment information could be used to determine communication needs. This idea was proposed in [7] and [12] and resulted in substantial performance enhancement.

Existing location-aware schemes, however, have the following limitations. They are based on particular types of schemes and cannot incorporate combinatorial solutions easily, as

combinatorial schemes use a set of groups to generate sensors' key rings. And combinatorial approaches have significant advantages: they guarantee deterministic performance close to optimal. They create less communication overhead in the path establishment phase as each sensor can determine shared keys with any other sensors just by looking at a sensor identifier (sensors do not need to exchange a long list of key identifiers they have). Furthermore, existing location-aware schemes are not flexible enough in terms of the trade-off between connectivity and resilience. In addition, existing location-aware schemes do not consider heterogeneous sensor networks that consist of several types of nodes that differ in the amount of memory and the communication range. As shown in [15] such networks are of particular interest as they can improve effectiveness of routing protocols and present a cheaper solution for monitoring wide areas.

To overcome the aforementioned limitations we propose a novel key predistribution framework that can use any existing key predistribution method as its underlying scheme, and that provides the user with flexibility to achieve different performance balances based on his needs. The basic idea of the proposed framework is to partition deployment area into a set of square cells, called a grid. Each cell on the grid is assigned with a fixed number of sensors. The size of each cell is chosen in a way that only sensors from neighboring cells could be located within the communication range meaning that only such sensors should share keys. In order to provide connectivity between such sensors a key pool of each cell is constructed so that there is an overlap between the neighboring cells. The overlap factor between the key pools is a flexible parameter and determines the balance between connectivity and resilience. After key pools for each cell are constructed any existing key predistribution scheme could be used to construct sensors' key rings. As an example we present an instantiation based on the combinatorial scheme proposed by Lee and Stinson in [10].

We further study the heterogeneous sensor networks consisting of sensors with different amount of memory and different communication ranges, and propose an extension of our framework for the heterogeneous case.

We confirm good performance of our framework by providing both analytical and simulation results. All analytical solutions are presented in formulas making it easy for the user to choose the right parameters to achieve required performance.

The contribution of this paper is twofold. Proposed key predistribution framework is

- Suited for using any existing key predistribution scheme, including combinatorial ones,
- Customizable by providing the user with options on how to utilize available resources to achieve the desired trade-off between connectivity and resilience.

We also propose an extension of the framework for heterogeneous environments and present preliminary performance results of this extension.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we describe a common architecture of the WSN, and define metrics used to evaluate framework performance. In Section 4 we introduce our general framework and use combinatorial Lee and Stinson scheme [10] to instantiate it. In Sections 5 and 6, we present analytical and simulation results respectively. Section 7 presents preliminary study of the key predistribution problem in heterogeneous environments. We conclude in Section 8.

1. RELATED WORK

First key predistribution schemes concentrated on distributing keys among sensors in a way to guarantee high connectivity among all pairs of sensors and assumed no prior knowledge about sensors' locations. In [9] Eschenauer and Gligor proposed probabilistic key predistribution scheme. In this scheme a key pool is generated off-line and every sensor picks a random subset of keys from this key pool. Any two nodes in the communication range can talk with each other only if they share a common key. Depending on the size of the key pool and the size of the sensor key set, the scheme can achieve different connectivity and resilience. Chan, Perrig and Song [6] later suggested to use the same idea, but increased the intersection threshold from one to some $q > 1$. In their scheme any two nodes are allowed to establish communication only if they share at least q keys. This modification enhanced the resilience of the scheme when a small number of nodes are compromised.

In [4, 10, 14] authors proposed to apply combinatorial design theory to key predistribution problem. The authors showed that using well-explored set systems can increase the probability of establishing short communication paths between sensors. In [4] Camtepe and Yener suggested hybrid design to combine probabilistic and deterministic approaches to overcome limitations of combinatorial approaches: possible unavailability of a scheme with particular parameters a user might request.

In [2] Blom proposed to generate keys using matrix multiplications. He introduced symmetric key matrix K, where $[i, j]$ element of the matrix $K_{i,j} = K_{j\ i}$ is a pairwise key between i-th and j-th nodes. The key matrix is obtained by multiplication of public matrix G that is known to all nodes and private matrix D. Each node i stores a corresponding row of the private matrix and is able to compute a pairwise key with any other node j by multiplying the row of the private matrix it stores with the corresponding column of the public matrix. In [8] Du, Deng and others proposed to combine Blom scheme with random predistribution scheme. They called the set of keys that each node can generate a key space and suggested to use multiple key spaces. In this scheme first w key spaces are generated and each sensor is loaded with τ key spaces, $2 \le \tau \le w$. This further improved resilience of the scheme.

In [3] Blundo et al proposed polynomial based key generation. It uses polynomial evaluations to obtain a pairwise key. Each node i gets polynomial shares $f(i, x)$ of symmetric polynomials of degree λ. In order to calculate a common key with node j, node i needs to evaluate its polynomial with $x = j$: $f(i, x = j)$. Node j would in turn evaluate $f(j, x = i)$ and since polynomials are symmetric $f(i, j) = f(j, i)$ this value could be used as a common key. This scheme is resistant against node capture: compromise of less than $\lambda + 1$ nodes doesn't reveal any information about keys of other nodes. In [11] the authors proposed several applications of the Blundo scheme. First they combined the Blundo scheme with random predistribution scheme [9]. A pool of polynomials is generated off-line and then each node gets polynomial shares of subsets of polynomials. They also proposed a grid-based solution, where

each sensor is placed on a grid, such that other sensors located at the same row or the same column on the grid share a polynomial.

Although all these schemes could provide good performance for small sensor networks they do not scale well when the size of the network increases. In large-scale sensor networks a scheme that tries to guarantee fixed connectivity between every pair of sensors would result in unacceptable memory requirements and would use key space very inefficiently as only small fraction of sensors would likely need to talk with each other. Essential milestone in the area was the idea of using sensors' expected locations to guarantee connectivity only between sensors located close to each other. This idea was independently proposed by Du, Deng and others in [7] and Liu and Ning in [12]. In [7] the authors extended their previous work [8] that was based on the Blom scheme. They proposed a new group-based deployment model, where all the sensors belonging to one group are deployed together from the same deployment point such that deployment points constitute a rectangular grid. Under the assumption of normal distribution of sensors within one group, they assumed that nodes would need to talk only if they belong to the same or neighboring groups. In their scheme the whole deployment area is partitioned into a set of cells, where each cell is occupied by one group. Each group gets a key space S_c that contains $| S_c |$ elements. The key spaces are generated in a way to guarantee the following properties: two groups that are horizontal or vertical neighbors share a key spaces with $0 \leq a \leq 0.25$. Two cells that are diagonal neighbors share b key spaces with $0 \leq b \leq 0.25$ and since each cell on the grid (excluding the border cells) has 4 diagonal neighbors, 2 horizontal and 2 vertical neighbors the following equality always holds: $4a + 4b = 1$. From this equation it follows that in this scheme the maximum overlap factor between horizontal and vertical neighbor cells is 0.25 (if overlap between diagonal cells is set to zero). This constraint is a critical limitation when good connectivity between sensors belonging to neighbor cells is required. In the proposed framework we overcome this limitation and allow varying the value of the overlap factor by trading off the resilience of the scheme.

In [12] Liu and Ning proposed several schemes using deployment information. In the closest location-based pair-wise key pre-distribution scheme they proposed to use nodes expected location and made each sensor to share keys only with its c closest neighbors. The scheme achieves good resilience and connectivity, but it is sensitive to deployment errors and locations of individual sensors are hard to predict. Addition of new nodes at later time would involve a lot of overhead of notifying existing nodes about the new nodes. In the same paper the authors proposed to use Closest Polynomials Predistribution Scheme. In this scheme deployment area is partitioned into a set of cells. Each i, j cell is associated with the same polynomial. Each node is assigned to one cell and gets a polynomial of its home cell and polynomial shares of its four neighbors. Resilience of the scheme depends on the degree of polynomials: using low degree polynomials make the network vulnerable to node capture, while evaluating high degree polynomials is a costly operation.

Both location aware schemes [7, 12] are based on probabilistic approach and could not incorporate combinatorial solution. Besides, they require sensors to share keys only if they belong to the same or neighbor cells. This property makes the schemes sensitive to deployment errors: misplacement of a node just by one cell from its expected location negatively affects connectivity.

In [14] the authors were first to consider a Key Predistribution Scheme in heterogeneous environment, when sensor network consisted of nodes with different amount of memory. They applied probabilistic approach to the network: the nodes used one large key pool and randomly selected a key set of size equal to the available memory. In [14] the authors did not use location information. In our paper we study more general case of heterogeneous network: we assume that nodes could differ not only in the amount of memory, but also in the communication range. We use location knowledge to distribute keys among sensors and show that using a separate key pool to provide connectivity between more powerful sensors could be beneficial rather than just increasing the number of keys drawn from the common key pool by more powerful sensors.

2. OVERVIEW

2.1 Network Architecture and Functioning

We assume we are dealing with a wide-area wireless sensor network with the following simplifying communication architecture. The sensor network consists of thousands of sensors (or nodes). Each sensor has limited *communication range* and can reach directly only those sensors located within its *communication circle* (a circle of radius equal to the communication range R and centered at the sensor). Wide-area network covers an area that many times exceeds the area covered by sensors' communication circle.

In order to communicate securely, sensors encrypt traffic using a session key that they build based on the keys they share. Sensors need to have at least q ($q \geq 1$) common keys in order to build a session key. The number q is known as the *intersection threshold*. Whenever a node wants to communicate with any other node, it first must check if it shares at least q keys with the destination node. If it does, it can use the common keys the nodes share to construct a session key and start data transfer using this session key to encrypt data [5]. We say there is a 1-*hop path* between sensors S_i and S_j ($i \neq j$) in the communication range if the sensors share at least q common keys. If nodes don't share at least q keys, they can't communicate directly, but they could employ intermediate nodes that would form a path connecting these two nodes. There exists a *k-hop path* ($k > 1$) between sensors S_i and S_j ($i \neq j$) in the communication range if the nodes can find (k–1) different intermediate nodes S_{h1}, S_{h2}, … $S_{h(k-1)}$, ($h1 \neq h2 \neq \ldots \neq h(k–1) \neq j \neq i$) such that there is a one-hop path between any pair of the adjacent nodes in the path: there is a one-hop path from S_i to S_{h1}, from S_{h1} to S_{h2}, …, from $S_{h(k-1)}$ to S_j. We call parameter k a *length of a path*.

The search for intermediate nodes is known as the *Path Key Establishment*. There are two ways to find intermediate nodes [5]. First, each node might store the network connectivity information that would provide knowledge about what nodes share what keys. The other way is to discover this information on the fly. In this approach a node that wants to communicate with other node and doesn't share q keys, broadcasts request for intermediate nodes that are able to connect the node initiating the request to the destination node. In this request the node initiating the request

sends some information that is enough for other nodes to determine if they share q keys. Those nodes that share q keys respond to the node-initiator and proceed with the path key establishment in a recursive way until the destination node is reached or all possible paths are explored. Path key establishment on the fly creates a lot of communication overhead that increases exponentially with the number of intermediate nodes. In order to decrease the overhead, small number of hops should be used.

2.2 Performance Measures

In this section we describe three metrics used to evaluate the key predistribution framework and to compare it with existing solutions. For any scheme all these metrics should be carefully considered. A good key predistribution scheme should provide the user with flexibility to vary the balances between the metrics based on application requirements and maximize the performance using available resources.

Connectivity is defined to be the probability that any two randomly chosen nodes located within the communication range can communicate securely by establishing a path between them. For every possible number of hops k in the path, there is a corresponding metric $\Pr_k$ that contributes to the connectivity. We define these metrics as follows.

The probability $\Pr_1$ is the probability that any two randomly chosen nodes S_i and S_j ($i \neq j$) from the deployment area in the communication range can establish a one-hop path. The probability $\Pr_k$ ($k > 1$) is the probability that any two randomly chosen nodes S_i and S_j ($i \neq j$) from the deployment area in the communication range can establish a k-hop path, but there is no path with fewer hops between them. Connectivity is a sum of $\Pr_k$ for $k = 1$ to K, where K is the maximum path length (system parameter). In practice K is often limited to two. It is desirable that the sum ($\Pr_1 + \Pr_2$) gets close to one and that $\Pr_1$ gives the greatest contribution.

Resilience shows how robust the network is against node capture attacks. In these attacks node capture leads to the disclosure of node's key set and all the keys belonging to compromised nodes become known to an adversary. The adversary can later use the known keys to eavesdrop communication between non-compromised nodes that are using the same keys. More formally, resilience fail(s) is a function of number of compromised nodes s and is defined to be the probability that communication between two non-compromised nodes could be eavesdropped when s nodes are compromised at random. Communication between a pair of nodes could be eavesdropped if among the keys the nodes share at least q keys occurred in the compromised nodes.

Scalability. Scalability characterizes the ability of a scheme to accommodate growth. The scheme is considered scalable if its performance does not degrade when the network size increases. A key predistribution scheme should be able to provide enough key rings to support network extension and re-deployment. Ideally, there should be no limitations on the number of nodes the network could support.

3. FRAMEWORK DETAILS

In this section we describe our framework in homogenous environment, when all the sensors have the same communication range R and the same amount of memory.

3.1 Framework Overview

Location-aware scheme uses the knowledge of sensors' possible locations to distribute keys. Because of the large size of the network it is infeasible to know the precise location of each sensor a-priori. Using random scattering as a deployment technique does not provide any location information as each sensor could appear anywhere within the deployment field. However, clustering sensors into a set of groups, and partitioning the deployment area into a set of cells allows associating each group of sensors with a cell on the grid limiting each sensor's possible location within the boundaries of that cell. This provides useful location-information that could be used to determine sensors' communication needs. At the same time, the requirement of sensors from one group being deployed within some area is not too demanding and is practically achievable by using special group-based deployment methods.

Similar to other location-aware schemes [7, 12] we partition the deployment area into a set of $N \times N$ square cells that constitute a grid such that the length of a cell side is equal to $2h$, and the following inequality holds: $2h \geq 2R$ (See fig. 3). Each cell on the grid is further assigned with the fixed number of sensors that are distributed uniformly within a cell. Each cell is responsible for generating its deployment key pool, constructing key rings, and distributing them among the sensors it holds. The length of a cell side is chosen to be $2h$ with $h \geq R$ in order to guarantee that a pair of sensors could be located within the range R only if they belong to the same or neighbor cells on the grid, therefore the connectivity between such pairs of sensors should be maximized.

To maximize connectivity between sensors belonging to neighboring cells, key pools of neighboring cells should overlap. In order to achieve this property, key pools of each cell are constructed in two steps. At first, every cell $h_{i,j}$ generates its *original key pool* $OKP_{i,j}$ consisting of K_{orig} keys: $OKP_{i,j} = \{k_{i,j,z} \mid z = 1..K_{\text{orig}}\}$.
Original key pools of different cells do not intersect (all the keys are unique):
$k_{i,j,z} \neq k_{x,y,v}$ if ($i \neq x$ or $j \neq y$ or $z \neq v$).
At the second step, each cell constructs its *deployment key pool* $DKP_{i,j}$ by taking a union of original key pools of cells located inside $m \times m$ square window of cells that is clipped from the grid such that the upper-left corner of the window is the cell whose deployment key pool is being constructed ($m > 1$, $m \leq N$):
$DKP_{i,j} = \{\cup_{x,y} OKP_{x,y} \mid x = i \;..\; (i+m), y = j \;..\; (j+m)\}$.
Each cell further uses its deployment key pool $DKP_{i,j}$ to generate key rings and distribute them among the sensors it holds. Any key predistribution scheme could be used to construct its key rings from the deployment key pools.

In order to guarantee the uniform performance over the whole grid and avoid the negative effect of border cells the grid is augmented by ($m - 1$) cells in both horizontal and vertical dimensions before key pools are constructed. Cells in the augmented area generate their original key pools, but do not store any sensors. Note that original key pools are used to construct deployment key pools only. Each cell uses its deployment key pool to generate sensors' key rings.

The following algorithm describes construction of deployment key pools more formally:

For (i = 1, .. $\boldsymbol{N}$ + [$\boldsymbol{m}$ – 1], j = 1 .. $\boldsymbol{N}$ + [$\boldsymbol{m}$ – 1]) {

$OKP_{i,j} = \{k_{i,j,z} \mid z = 1..\ K_{\text{orig}}\}$

}

For (i = 1, .. $\boldsymbol{N}$, j = 1 .. $\boldsymbol{N}$) {

$DKP_{i,j} = \{\}$

for (y = 0; $y < m$; y++) {

for (x = 0; $x < m$; x++) {

$DKP_{i,j} = DKP_{i,j} \cup OKP_{i+y,j+x}$

}

}

}

Constructing key pools in the way described above leads to the following properties. Sensors belonging to the same cell use the same deployment key pool. Sensors belonging to neighboring cells use deployment key pools that are different, but that overlap. The further away the cells are located on the grid, the less the overlap between the deployment key pools is. The user can further choose any key predistribution scheme to construct key rings from the deployment key pools and distribute them among the sensors. Depending on the value of the parameter m that we call the *overlap factor* and a chosen mechanism to generate key rings from the deployment key pools, different resilience and connectivity could be achieved. In general increasing the value of m will result in better connectivity between sensors belonging to neighboring cells, but will also affect resilience by increasing the visibility of every key and making the network more vulnerable to node compromise. The trade-off between connectivity and resilience should be carefully considered by the user.

Consider an example on Fig. 1. The deployment area is partitioned into 4×4 cells (N = 4) and m = 2. First the grid is augmented by m – 1 = 1 cell in both vertical and horizontal dimensions. Each cell on the augmented 5×5 grid $h_{i,j}$ (i = 1 .. 5, j = 1 .. 5) constructs its original key pool $OKP_{i,j}$ by generating K_{orig} unique keys. Then each real cell $h_{i,j}$ (i = 1 .. 4, j = 1 .. 4) constructs its deployment key pool as

$DKP_{i,j} = \{\mathrm{U}_{x,y}\ OKP_{x,y} \mid x = i\ ..\ (i+2), y = j\ ..\ (j+2)\}$

So, deployment key pool of node $h_{1,1}$ $DKP_{1,1}$ consists of the following four sets $\{OKP_{1,1}$, $\boldsymbol{OKP_{1,2}}$, $OKP_{2,1}$, $\boldsymbol{OKP_{2,2}}\}$. Deployment key pool of node $h_{1,2}$ $DKP_{1,1}$ consists of these four sets $\{\boldsymbol{OKP_{1,2}}$, $OKP_{1,3}$, $\boldsymbol{OKP_{2,2}}$, $OKP_{2,3}\}$. Deployment key pools of these two neighboring nodes have two subsets in common: $\boldsymbol{OKP_{1,2}}$, and $\boldsymbol{OKP_{2,2}}$ meaning that the fraction of common keys in the deployment key pools of these two nodes is 1:2. Deployment key pools of cells that are not neighbors do not intersect. For example, deployment key pools of cells $h_{1,1}$ and $h_{1,3}$ consist of completely different key pools. If we increase m from 2 to 3, deployment key pools of each cell would consist of 3×3 = 9 original key pools, and key pools of neighboring cells would intersect in 6 key pools, so the fraction of common keys in the key pools of neighboring cells would be 2:3. But there is still an overlap between deployment key pools of cells that are located one cell apart.

a)

$OKP_{1,1}$	$\boldsymbol{OKP_{1,2}}$	$OKP_{1,3}$	$OKP_{1,4}$	$OKP_{1,5}$
$OKP_{2,1}$	$\boldsymbol{OKP_{2,2}}$	$OKP_{2,3}$	$OKP_{2,4}$	$OKP_{2,5}$
$OKP_{3,1}$	$OKP_{3,2}$	$OKP_{3,3}$	$OKP_{3,4}$	$OKP_{3,5}$
$OKP_{4,1}$	$OKP_{4,2}$	$OKP_{4,3}$	$OKP_{4,4}$	$OKP_{4,5}$
$OKP_{5,1}$	$OKP_{5,2}$	$OKP_{5,3}$	$OKP_{5,4}$	$OKP_{5,5}$

b)

$OKP_{1,1}$	$\boldsymbol{OKP_{1,2}}$	$OKP_{1,3}$	$OKP_{1,4}$	$OKP_{1,5}$
$OKP_{2,1}$	$\boldsymbol{OKP_{2,2}}$	$OKP_{2,3}$	$OKP_{2,4}$	$OKP_{2,5}$
$OKP_{3,1}$	$OKP_{3,2}$	$OKP_{3,3}$	$OKP_{3,4}$	$OKP_{3,5}$
$OKP_{4,1}$	$OKP_{4,2}$	$OKP_{4,3}$	$OKP_{4,4}$	$OKP_{4,5}$
$OKP_{5,1}$	$OKP_{5,2}$	$OKP_{5,3}$	$OKP_{5,4}$	$OKP_{5,5}$

Fig 1 Constructing deployment key pools: m =2 , N = 4. Original key pools used to construct deployment key pools are shaded:
a) constructing deployment key pool for node $h_{1,1}$;
b) constructing deployment key pool of node $h_{1,2}$

3.2 Review of Lee and Stinson Scheme

In this section we present detailed overview of Lee and Stinson scheme because we further use the scheme to instantiate the framework to generate key rings.

Lee and Stinson proposed using transversal designs for key predistribution scheme [10]. As the authors define in their paper given integers t, k, p and a finite set X, consisting of kp elements, generalized transversal design $TD(t, k, p)$ is a triple (X, H, A), where

- H is a partition of X into k groups of size p and
- A is a set of p^t blocks, where each block consists of k points from X.

Groups and blocks satisfy the following properties:

- Every block intersects every group in exactly one point: $|H_g \cap A_j| = 1$ for every $H_g \in H$ and every $A_j \in A$, and
- Any t elements chosen from t different groups are included in exactly one block.

It is proved that transversal designs exist when p is a prime and k doesn't exceed p.

Consider an example. Let's choose t = 2, k = 3, p = 5.

Let X be a set of letters of alphabet of cardinality kp = 3x5 = 15

$X = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o\}$.

H is a partition of X into 3 groups of size 5. Let's partition X as follows:

$H = \{\{a, b, c, d, e\}, \{f, g, h, i, j\}, \{k, l, m, n, o\}\}$

Then the following set A is a set of blocks that satisfy properties of the transversal design:

$A = \{\{a, f, k\}, \{a, g, m\}, \{a, h, o\}, \{a, i, l\}, \{a, j, n\},$
$\{b, g, l\}, \{b, h, n\}, \{b, i, k\}, \{b, j, m\}, \{b, f, o\},$
$\{c, h, m\}, \{c, i, o\}, \{c, j, l\}, \{c, f, n\}, \{c, g, k\},$
$\{d, i, n\}, \{d, j, k\}, \{d, f, m\}, \{d, g, o\}, \{d, h, l\},$
$\{e, j, o\}, \{e, f, l\}, \{e, g, n\}, \{e, h, k\}, \{e, i, m\}\}$

The triple (X, H, A) is a TD (2, 3, 5). The reader may verify that all the specified properties hold: every group intersects with any block in exactly one point; any subset of two elements from two different groups occurs in exactly one block.

In the paper [10] the authors applied transversal designs to key predistribution problem as follows. They used X as a key pool and every block $A_j \in A$ as a sensor key ring. Every sensor in the network is assigned with exactly one block that is randomly chosen from the set of key rings A.

Lee and Stinson proposed two different use cases of transversal designs: linear scheme uses TD(2, k, p) with the intersection threshold one (q = 1) and quadratic scheme uses TD(3, k, p) with the intersection threshold two (q = 2). The authors analyzed the performance of both schemes. In our paper we are using quadratic scheme due to its enhanced properties.

For the quadratic scheme TD(3, k, p) the authors showed the following properties hold. For a set system (X, H, A) that is a TD(3, k, p) any block $A_i \in A$ intersects

$a_1 = k(p - 1)(p - k + 2)$ blocks in one point,

$a_2 = \binom{k}{2}(p - 1)$ blocks in two points

and is disjoint from $a_0 = p^3 - 1 - a_1 - a_2$ blocks.

Using these properties the authors found the following estimations for connectivity:

The probability to establish a one-hop path

$$\Pr_1 = a_2 / (p^3 - 1) = k(k - 1) / 2(p^2 + p + 1). \quad (1)$$

Resilience against node capture is

$$\text{fail}(s) = 1 - \frac{2\binom{p^3 - p^2}{s} - \binom{p^3 - 2p^2 + p}{s}}{\binom{p^3 - 2}{s}} \quad (2)$$

3.3 Framework instantiation using Lee and Stinson Scheme

In this section we present an instantiation of the general framework with Lee and Stinson combinatorial scheme [10]. In this instantiation each cell constructs the key rings using transversal design (see Section 4.3). For each cell, deployment key pool acts as a finite set X and every original key pool, contributing to a cell's deployment key pool acts as a group H_g. The group size k is equal to K_{orig} – number of keys in the original key pool and the number of groups p is the number of the original key pools contributing to the deployment key pool that is equal to $m \times m = m^2$. Using the set of groups H (the set of the original key pools contributing to the deployment key pool), the set of blocks A is constructed using quadratic transversal design (transversal design with t = 3). We choose quadratic transversal design due to its enhanced properties [10]. Each block $A_j \in A$ serves as a sensor key ring. Each cell further distributes key rings to its sensors by randomly picking a block. After a block is appointed to a sensor, the block is deleted from the set. So, every sensor gets unique block.

Important relationships between parameters of the scheme are:

- For each cell deployment key pool X is the union of the original key pools of $m \times m$ cells. This means that the size of the finite set X is $K_{\text{depl}} = |X| = m^2 K_{\text{orig}}$.
- Each original key pool contributing to the deployment key pool serves as a group H_g, so the set of groups consists of $k = m^2$ groups.
- Each group H_g consists of $p = K_{\text{orig}}$ elements.

In order for overlapping deployment key pools to intersect in a known set system and provide deterministic performance, the following property should hold. Whenever deployment key pools of different cells contain original key pool of the same cell $h_{i,j}$ and use it as a group H_g, the group H_g should be assigned with the same index g in all the occurrences of the original key pool of the cell $h_{i,j}$. An example on fig. 2 shows a possible assignment of group indices to the cells on the grid. Let m = 2. Whenever original key pool of a cell $h_{i,j}$ is used, it should be used as a group H_g with the same index g shown to the right. For example, the cell $h_{1,1}$ uses original key pools of cells $h_{1,1}$, $h_{1,2}$, $h_{2,1}$, $h_{2,2}$. Following the assignment on fig. 2 original key pool of cell $h_{1,1}$ is used as the group H_1, $h_{1,2}$ –as the group H_2, $h_{2,1}$ – as the group H_3, and $h_{2,2}$ – as the group H_4. The cell $h_{1,2}$ uses original key pools of cells $h_{1,2}$, $h_{1,3}$, $h_{2,2}$, $h_{2,3}$. Following the assignment on fig. 2 original key pool of cell $h_{1,2}$ is used as the group H_2, $h_{2,2}$ – as the group H_4, $h_{1,3}$ – as the group H_1, and $h_{2,3}$ – as the group H_3. So, both cells use original key pools of the cell $h_{1,2}$ as the group H_2 with index 2 and the cell $h_{2,2}$ as the group H_4 with index 4.

$h_{1,1}$ --> H_1	$h_{1,2}$--> H_2	$h_{1,3}$--> H_1	$h_{1,4}$--> H_2	$h_{1,5}$--> H_1
$h_{2,1}$--> H_3	$h_{2,2}$--> H_4	$h_{2,3}$--> H_3	$h_{2,4}$--> H_4	$h_{2,5}$--> H_3
$h_{3,1}$--> H_1	$h_{3,2}$--> H_2	$h_{3,3}$--> H_1	$h_{3,4}$--> H_2	$h_{3,5}$--> H_1
$h_{4,1}$--> H_3	$h_{4,2}$--> H_4	$h_{4,3}$--> H_3	$h_{4,4}$--> H_4	$h_{4,5}$--> H_3
$h_{5,1}$--> H_1	$h_{5,2}$--> H_2	$h_{5,3}$--> H_1	$h_{1,1}$--> H_2	$h_{1,1}$--> H_1

Fig 2. Assigning labels to groups: m = 2, N = 4. Whenever original key pool of a cell $h_{i,j}$ is used, it should be used as a group H_g with index g shown to the right.

To make parameters more flexible each cell could generate c original key pools (c > 1). Then each cell would construct its deployment key pool using c original key pools of $m \times m$ cells. In this case every original key pool would act as a group and

parameters of the set system would be as follows. There would be cm^2 groups. Each group would consist of $p = K_{orig}$ elements, so the size of deployment key pool would be $K_{depl} = |X| = kp = cm^2 K_{orig}$.

4. ANALYSIS

In this section we analyze the performance of the framework, instantiated with Lee and Stinson scheme [10]. Similar analysis could be performed for any other instantiation.

4.1 Connectivity Estimation

In order to find the connectivity of the scheme we need to analyze the fraction of each type of nodes in a communication circle on average for any possible sensor location on the grid, and the probability of sharing keys between each type of sensors. We assume the length of a cell side is $2h$ and $2h \geq 2R$ (see fig. 3). The inequality $2h \geq 2R$ implies that a pair of nodes is located within the communication range R only if the nodes belong to the same or neighboring cells. Sensors belonging to non-neighboring cells are guaranteed to be outside of the range R. So, any node can reach only three types of sensors: nodes belonging to the same cell, nodes belonging to diagonal neighbors, and nodes belonging to horizontal/vertical neighbors. Because both horizontal and vertical neighbors have the same properties we treat this case as one and further refer to it as the *side neighbors*.

Let $f_{\text{in-cell}}$ be the fraction of sensors inside the communication circle of a sensor belonging to the same cell, f_{side} be the fraction of sensors belonging to the side neighbors, f_{diag} be the fraction of sensors belonging to the diagonal neighbors. Since any sensor can reach only these three types of sensors, the following equation holds:

$$f_{\text{in-cell}} + f_{\text{side}} + f_{\text{diag}} = 1. \tag{3}$$

We further present the final results only; derivations of these formulas are shown in the Appendix.

The fraction of in-cell neighbors is equal to:

$$f_{\text{in-cell}}(h, R) = \frac{9\pi - 5}{12\pi} \times \frac{R^2}{h^2} + \frac{2}{3} \frac{(h - R)R}{h^2} \times \frac{3\pi - 2}{\pi} + \frac{(h - R)^2}{h^2} - \frac{R^2 (6 + (4 - 3\pi))}{12 h^2 \pi} \tag{4}$$

The fraction of diagonal neighbors is:

$$f_{\text{diag}}(h, R) = \frac{R^2}{8 h^2 \pi} \tag{5}$$

Then from equation (3) the fraction of side neighbors is equal to:

$$f_{\text{side}}(h, R) = 1 - (f_{\text{in-cell}}(h, R) + f_{\text{diag}}(h, R)) \tag{6}$$

where

$f_{\text{in-cell}}(h, R)$ is found using the formula (4),

and $f_{\text{diag}}(h, R)$ uses formula (5).

From formulas (4, 5, 6) it follows that increasing the cell size would increase the fraction of in-cell neighbors and decrease the fraction of sensors of other types in sensor's communication circle.

We now need to analyze the probabilities of sharing keys between each type of pairs. We start from analyzing the probability of sharing a key between sensors belonging to the same cell. Sensors assigned to the same cell use the same deployment key pool and their key rings are constructed using transversal design. So the connectivity of this case simply complies with Lee and Stinson formula for connectivity of TD(3, k, p) (1), where $k = m^2$, $p = K_{orig}$. We further refer to the formula (1) as $P_{L\&S}(t, k, p)$ and use $t = 3$. So, the in-cell connectivity is equal to:

$$P_{\text{in-cell}} = P_{L\&S}(3, k = m^2, p = K_{orig}) \tag{7}$$

$$P_{\text{in-cell}} = \frac{m^2 (m^2 - 1)}{2 (K_{orig}^2 + K_{orig} + 1)}$$

Sensors belonging to diagonal neighbors use different key pools that have $(m^2 - 2m + 1)K_{orig}$ keys in common out of total $m^2 K_{orig}$ keys and key rings of these types of sensors intersect in TD(3, $k = m^2 - 2m + 1$, $p = K_{orig}$). So, the connectivity between these types of sensors is equal to:

$$P_{\text{diag}} = P_{L\&S}(k = m^2 - 2m + 1, p = K_{orig}) \tag{8}$$

$$P_{\text{diag}} = \frac{(m^2 - 2m + 1)((m^2 - 2m + 1) - 1)}{2 (K_{orig}^2 + K_{orig} + 1)}$$

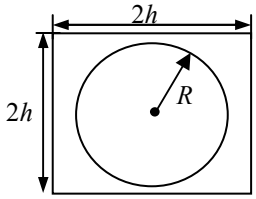

Fig 3. Geometry of a grid cell:
each cell covers a $2h \times 2h$ square region;
R – the communication range; $2R \leq 2h$

Sensors belonging to the side neighbors use the key pools that overlap in $m(m - 1)$ K_{orig} keys out of total $m^2 K_{orig}$ keys and the key rings of these types of sensors intersect in TD(3, $k = m(m - 1)$, $p = K_{orig}$). So, the connectivity between side neighbors is:

$$P_{\text{side}} = P_{L\&S}(k = m(m - 1), p = K_{orig}) \tag{9}$$

$$P_{\text{side}} = \frac{(m^2 - m)((m^2 - m) - 1)}{2 (K_{orig}^2 + K_{orig} + 1)}$$

The overall connectivity of the scheme is:

$P_1 = P_{\text{in-cell}} f_{\text{in-cell}} + P_{\text{side}} f_{\text{side}} + P_{\text{diag}} f_{\text{diag}}$.

Where $P_{\text{in-cell}}$ – is (7); $f_{\text{in-cell}}$ – is (4);

P_{side} – is (9); f_{side} – is (6);

P_{diag} – is (8); f_{diag} – is (5).

4.2 Resilience against node capture

Resilience of the framework when s sensors are compromised at random complies with the following equation:

$$\text{fail}_{\text{grid}}(s) = \sum_{i=0}^{s} \text{fail}_{\text{orig}}(i) \binom{s}{i} \left(\frac{m^2}{N^2}\right)^i \left(1 - \frac{m^2}{N^2}\right)^{s-i}$$

Where $\text{fail}_{\text{orig}}(s)$ is resilience of Lee and Stinson scheme (2)

5. SIMULATION RESULTS

In this section we study performance of the framework through simulations.

First we explore the impact of varying cell size on the connectivity. Fig. 4 shows the connectivity of the framework with $m = 4$ and $N = 10$ for different values of the fraction h/R and compares it to the connectivity of the original Lee and Stinson scheme with the same parameters. In both cases TD($t = 3$, $k = 16$, $p = 17$) was used to build sensor key rings, and each sensor was preloaded with 16 keys. In both cases network contained 4000 sensors. As follows from the plot increasing the cell size leads to enhanced connectivity of the framework. This coincides with analytic results as increasing the cell size leads to the fact that each sensor has larger number of in-cell neighbors inside its communication range. The value of connectivity reaches the value of the original Lee and Stinson scheme fast

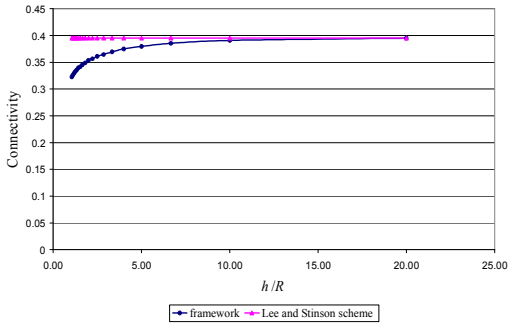

Fig 4. Connectivity of the framework and Lee and Stinson scheme

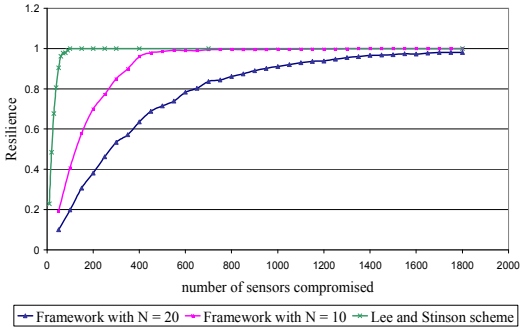

Fig 5. Resilience of the framework with Lee and Stinson scheme

We further study resilience of the scheme. Fig. 5 shows resilience of the scheme for different number of cells N and compares it with the resilience of the original scheme. In both cases the total number of sensors in the network is the same. TD($t = 3$, $k = 16$, $p = 17$) was used to build sensors' key rings; the same connectivity of 0.39 was achieved in both cases. As follows from the figure increasing number of cells leads to enhanced resilience. The framework substantially outperforms original Lee and Stinson scheme.

We further instantiate the framework with the Basic Probabilistic scheme [9] and compare the framework performance with different m to the basic scheme. Fig. 6 shows resilience of the framework for $m = 2$ and $m = 4$, and the Basic Probabilistic Scheme. The following parameters were used: $N = 20$, each cell contains 20 keys. Deployment key pool of cells for both cases contains 400 keys. Fig. 6 clearly illustrates the advantage of using the framework over the basic scheme. The following connectivity was achieved:

for $m = 2$ $Pr_1 = 0.52$;
for $m = 4$ $Pr_1 = 0.62$;
for the basic Probabilistic scheme $Pr_1 = 0.65$.

Increasing m leads to increasing the connectivity while degrades the resilience. In case of a larger grid (N on the order of 100) the degradation in resilience would not be so dramatic, while enhancement in connectivity would still be important.

Fig. 6. Resilience of the framework compared to the Basic Probabilistic Scheme

6. PRELIMINARY STUDY OF HETEROGENEOUS NETWORK

In this section we extend the framework when the network consists of sensors with different amount of resources. As pointed out in [15] such sensor networks are of particular interest because they could improve the performance and effectiveness of routing protocols while minimizing the price of the network.

Without loss of generality we assume there are two types of sensors: sensors that are more powerful (we call them type S [*S*tronger] sensors). These sensors can store up to mem_s keys and communicate with other sensors located within the range r_s. And there are sensors that are less powerful (we call them type W sensors [*W*eaker]). They can store up to mem_w keys ($mem_w < mem_s$) and can communicate with other sensors inside the range r_w ($r_w < r_s$).

In such environment type W sensors are able to communicate with other sensors located within the range r_w. Type S sensors are able to communicate with other type S sensors located within the range r_s, and type W sensors located within the range r_w. Despite the fact that type S sensors would be able to reach type W sensors within the larger range r_s, type W sensors would not be able to respond to them if they are more than r_w apart, and we are not interested in one-way communications as there is no way to confirm packet delivery.

So, such system has the following communication needs:

- Each sensor of type W needs to communicate with other sensors (both type W and type S) within the range r_w;

- Each sensor of type S needs to communicate with type W sensors within the range r_w and with type S sensors within the range r_s.

In order to satisfy the aforementioned communication needs and to achieve location-aware solution, similar to the homogenous case, deployment area is partitioned into a set of cells constituting a grid. All the sensors are clustered into a set of groups and each group is associated with a cell on the grid: sensors from one group could be deployed anywhere within the corresponding cell. Every group of cells contains sensors of both types. Just as in the homogeneous case the cell size $2h$ is bounded by the diameter of the communication circle of weaker sensors $2r_w$. In order to provide connectivity between weaker sensors as well as between weaker and stronger sensors within the range r_w we imagine that all the sensors are of type W: have mem_w free memory and r_w communication range. So, the problem reduces to the problem of key predistribution in homogeneous environment, which could be solved using the proposed framework. This step provides connectivity between all pairs of sensors within the range r_w. We follow the normal steps of the framework: we choose the value of the overlap factor m_w. Each cell constructs original key pool and builds its deployment key pool as a union of original key pools of cells lying inside $m_w \times m_w$ square window. We call the deployment key pools built on this step type W key pool. Each cell uses its deployment key pool to construct key rings of size mem_w and distributes them among all the sensors it holds.

After the first step is completed each sensor is assigned with mem_w keys. Since type S sensors have memory available for mem_s keys and $mem_s > mem_w$, type S sensors still have unoccupied memory that is enough for (mem_s - mem_w) keys. There are several options how this memory could be utilized. First, type S sensor could get (mem_s - mem_w) keys from the same type W deployment key pool it was using to get its first mem_w keys. This solution is not good enough as type W deployment key pool was constructed in a way to maximize the connectivity between the sensors within the communication range r_w, and the actual communication range of type S sensors is r_s that could be much larger than r_w. We need to provide good connectivity between type S sensors that can't be done using type W key pool as type S sensors could reach other type S sensors located several cells apart from it and type W key pool is built in a way to guarantee good connectivity only between sensors belonging to neighboring cells. Therefore, just getting more keys from type W key pool will not provide good connectivity between type S sensors.

In order to guarantee connectivity between type S sensors within the range r_s, separate deployment key pools are constructed. At first, existing cells are grouped into super-cells, such that each super-cell covers an area of the communication circle of stronger sensor r_s. Then each super-cell constructs its original key pool (we call it type S original key pool) and chooses the overlap factor m_s. Note that all the keys in type S key pools are different from those used in type W key pools. Then each super-cell constructs type S deployment key pools for stronger sensors as a union of the original key pools of super-cells lying inside $m_s \times m_s$ square window of super cells. Keys from type S deployment key pools are further used to construct key rings of size ($mem_s - mem_w$) and distribute them among type S sensors that belong to corresponding super-cell. The process is illustrated on fig. 7.

This solution uses two different key pools separating the role of each key pool: type W key pool is used to provide the connectivity between sensors within the range r_w, while type S key pool is used to provide the connectivity between stronger sensors. Solutions in the middle are possible when first all the sensors get mem_w keys from the type W key pool. And then type S sensors split their available memory ($mem_s - mem_w$) in some parts between the two components: it gets i keys ($i = 0 \,..\, mem_s - mem_w$) from type S key pool and the remaining ($mem_s - mem_w - i$) keys from the type W key pool.

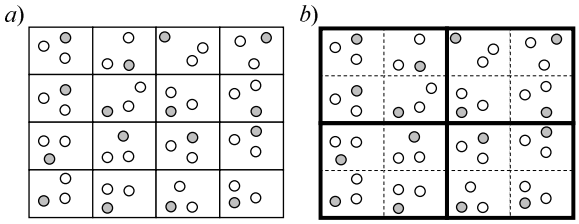

Fig 7. Key Predistribution framework for heterogeneous sensor network. Network consists of two types of sensors: type S sensors are shown as gray circles; type W sensors are shown as white circles. Key predistribution follows these steps:
step 1: all sensors are viewed to be type W sensors and key predistribution follows normal steps of the framework in homogeneous environment: deployment area is partitioned into a set of cells, each cell constructs its deployment key pool and distribute key rings among all the sensors it holds;
step 2: cells are grouped into super-cells. Each super cell constructs its deployment key pool, following the steps of the framework;
step3: key rings of super-cells are fulfilled with keys from the deployment key pools constructed in step 1 and step 2.

Fig. 8 shows an impact of this decision on the connectivity and resilience by simulating the network with the following parameters. Communication range of weaker sensors is $r_w = 2$ units; communication range of stronger sensors is $r_s = 20$ units. Weaker sensors can store up to $mem_w = 20$ keys, stronger sensors could store up to $mem_s = 40$ keys. Deployment area is partitioned into 16×16 cells ($N = 16$). Size of a cell side is $2h = 4$ units. Key rings are constructed from the deployment key pools by randomly selecting subset of keys. 12% of the sensors in each cell are of type S. From this figure it follows that it is more beneficial to use a separate key pool for stronger sensors and use more keys from this key pool to distribute them among type S sensors rather than use one and the same key pool and load all the keys from the type W key pool. In this case we get substantial improvement in connectivity.

By separating key pools used for different purposes we are able to enhance connectivity and resilience. If stronger sensors draw the same number of keys from type W key pool as weaker sensors, and draw the remaining (mem_s - mem_w) keys from type S key pool ($i = 0$), then compromise of stronger sensors have the same impact on the weaker sensors as the compromise of weaker sensors.

Further careful analysis of the behavior of the scheme for different network parameters is necessary and would be a part of our future work.

7. CONCLUSION

In this paper we presented a novel key predistribution framework for wide-area wireless sensor networks that is well suited for using with any existing key-predistribution scheme. The framework provides the user with flexibility in terms of the trade-off between connectivity and resilience depending on the needs of the application. By simulation and analysis we confirmed the good performance of the framework.

We also presented a preliminary study of heterogeneous sensor networks that incorporate nodes with different communication ranges and amount of memory. We showed that using different key pools to guarantee connectivity among stronger sensors results in significant performance enhancement. Further analysis of the framework performance in the heterogeneous environment will be our future work.

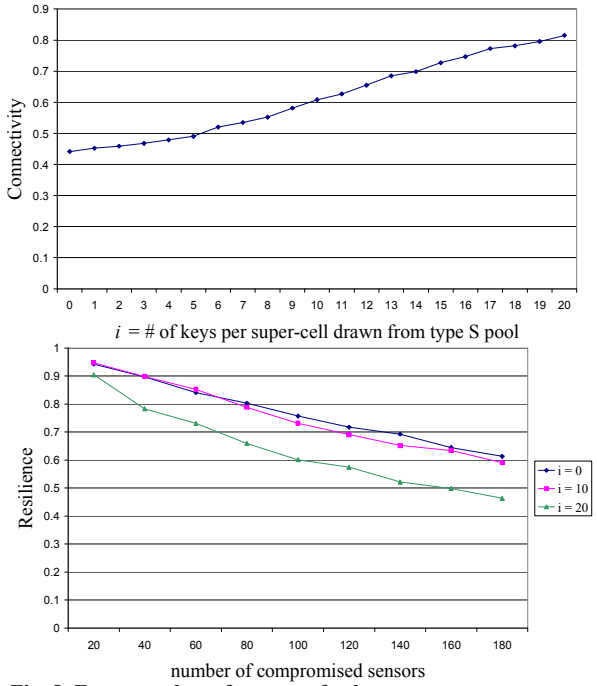

Fig 8. Framework performance for heterogeneous sensor network:
i keys out of 20 are drawn from type *S* key pool;
(20 - *i*) keys are drawn from type *W* key pool.
Increasing *i* improves connectivity substantially, but doesn't dramatically decrease resilience

8. ACKNOWLEDGEMENTS

The work of Simonova and Wang is partially supported by the NSF grant IIS-0430165.

9. REFERENCES

[1] F. Akylidiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4): 392 – 422, 2002

[2] R. Blom An optimal class of symmetric key generation systems. *Advances in Cryptology: Proceedings of EUROCRYPT* 84. Lecture Notes in Computer Science, Vol. 209. Springer Verlag, New York, 335 – 338, 1985

[3] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In Advances in Cryptology (CRYPTO '92), 471 – 486, 1993

[4] S. A. Camtepe, and B. Yener. Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks. *Computer Security – ESORICS*, Springer-Verlag, LNCS 3193, 2004, pp 293-308.

[5] S.A. Camtepe, and B. Yener. Key Distribution Mechanisms for Wireless Sensor Networks: a Survey. Technical Report TR-05-07, Rensselaer Polytechnic Institute

[6] H. Chan, A. Perrig, and D. Song Random Key Predistribution Schemes for sensor networks. *Proceedings of the IEEE Symposium on Research in Security and Privacy.* Infocom. IEEE Computer Society Press, Los Alamitos, CA 197 – 203, 2003

[7] W. Du, J. Deng, Y. S. Han, S. Chen, P. Varashney. A key management scheme for wireless sensor networks using deployment knowledge. *Proceedings of IEEE INFOCOM'04.* IEEE Computer Society Press, Los Alamitos, CA, 2004

[8] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili "A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks", *ACM Transactions on Information and System Security* 8(2): 228-258, 2005

[9] L. Eschenauer, V.D. Gligor. A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM Conference on Computer and Communication Security* (*CCS'02*). ACM, New York, 41 – 47, 2002

[10] J. Lee, D. R. Stinson. On the Construction of Practical Key Predistribution Schemes for Distributed Sensor Networks using Combinatorial Designs CACR 2005-40 Technical report, 2005

[11] D. Liu and P. Ning. Establishing Pairwise keys in distributed sensor networks. *Proceedings of the 10th ACM Conference on Computer and Communication Security* (*CCS'03*). ACM, New York, 52 – 61, 2003

[12] D. Liu and P. Ning. Location-Based pairwise key establishments for static sensor networks. *Proceedings of the 2003 ACM Workshop on Security in Ad Hoc and Sensor Networks* (*SASN'03*). ACM, New York, 72 – 82, 2003

[13] D. R. Stinson. Combinatorial Designs: Constructions and Analysis, Springer-Verlag, New York, 2003

[14] P. Traynor, H. Choi, G. Cao, S. Zhu, T. La Porta, "Establishing Pair-Wise Keys in Heterogeneous Sensor Networks" *IEEE INFOCOM*, April 2006.

[15] http://www.intel.com/research/exploratory/heterogeneous.htm

10. APPENDIX

In the appendix we show derivation of the fractions of different types of sensors within the communication circle of a sensor discussed Section 5.

In-cell neighbors. We first find an average fraction of nodes in the communication circle of a sensor that belong to the same cell. On average the fraction of reachable sensors that belong to the same cell is equal to the fraction of the communication circle falling inside the same cell. As sensors are uniformly distributed, we further find an average value of this fraction for any point inside the cell. In order to find this value we first look at several particular locations of a sensor and later find a general expression for any possible location. When a sensor is located at the upper-left border of a cell (Fig. 10 *a*), a quarter of the communication circle falls inside the same cell, so in this position only $1/4^{th}$ of all sensors belong to the same cell. If we move the sensor within the first quarter of the cell to the location (x_0, y_0) $(x_0 \le h, y_0 \le h)$ we observe the following behavior (assume that the coordinates' origin is located at the upper-left corner of a cell.). When we increase x_0 and y_0 coordinates of a sensor from 0 to R, the fraction of the area falling inside the same cell starts growing. At the location $x_0 = R$ and $y_0 = R$ the whole circle falls inside the cell, so the fraction is equal to 1. Since $R \le h$ if we keep increasing x_0 and y_0 until we reach the middle point of the circle, the fraction stays equal to 1. The other three quadrants of a cell are symmetric to the first quadrant and would result in the same value of the fraction, so it is sufficient to find the fraction for the first quadrant only (see fig. 10).

We further look at the regions of the first quadrant of a cell. On fig. 9 we show these regions and designate them as Part *A*, Part *B*', part *B*'', and Part *C*.

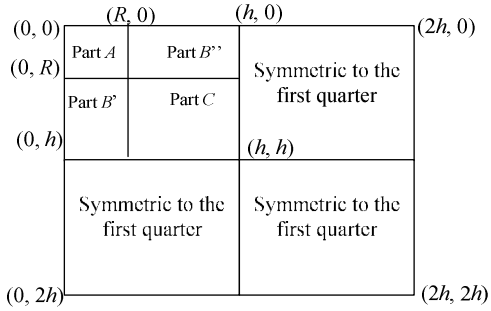

Fig 9. Finding an average fraction of the communication circle falling inside the same cell.
Parts *A*, *B*', *B*'': only part of the circle falls inside the cell;
Part *C*: the whole circle falls inside the cell

We now look at sensor located at arbitrary (x_0, y_0) coordinates within Part *A* (see fig. 9): $0 \le x_0 \le R$, $0 \le y_0 \le R$. For any location of a sensor within this region only a part of the area is falling inside the same cell and it consists of four pieces: an area of a quarter of a circle + an area of rectangle + an area of two circle segments (fig. 10 *d*). If an upper left corner of a square lies outside the circle, a part of a rectangular piece is cut off by the upper bound of a circle (fig. 10 *e*). That means the area of the region is bounded by the minimum value of two curves – the upper bound of the circle and the upper bound of the cell. If we shift the coordinate origin to the location of the sensor (the origin of the communication circle), then the coordinates of the upper left corner of the cell would have the (x_0 , y_0) coordinates (fig. 10 *d*, *e*). So, the area of the circle falling inside the cell for a sensor located within Part *A* is described by the following expression (fig. 10 *d*, *e*):

$$Area\,(\text{part}A) = \int_0^{x_0} \sqrt{R^2 - x^2}\, dx + \int_0^{y_0} \sqrt{R^2 - y^2}\, dy + \int_0^{x_0} \text{Min}\left[y_0,\ \sqrt{R^2 - x^2}\right] dx + \frac{\pi R^2}{4}$$

$$Area\,(\text{part}A) = Area_{\text{sect1}} + Area_{\text{sect2}} + Area_{\text{rect/cut rect}} + \frac{\pi R^2}{4}$$

We now look at the Part *B*': $0 \le x_0 \le R$ and $R \le y_0 \le h$. Part B''($R \le x_0 \le h$ and $0 \le y_0 \le R$) is symmetric to this case and would give the same expression. In this case a segment of a circle gets cut-off (see fig. 11) and the expression for the area falling inside the circle is:

$$Area\,(\text{part}B'') = Area\,(\text{part}B') = \pi R^2 - Area_{\text{segm}} = \pi R^2 - \frac{1}{2} R^2 (2\theta - \text{Sin}(2\theta))$$

where $\theta = \text{ArcCos}\left(\frac{x_0}{R}\right)$

For Part *C* (see fig. 9): $R \le x_0 \le h$ and $R \le y_0 \le h$ the whole circle falls inside the cell, so the area is equal to $Area\,(\text{part}C) = \pi R^2$

Since the sensor is equally likely to be located anywhere within the first quarter of the cell, we need to integrate the area for every location within the cell (when x_0 and y_0 goes from 0 to h) and divide the integral by the length of the integration region by each coordinate (h). The following equation describes the average fraction of sensors belonging to the same cell when sensor is located anywhere within the first quadrant of a cell. Since in the remaining three quarters we get the same expression (because of symmetry) the expression describes the fraction of sensors falling inside the cell for any location of a sensor on the grid.

$$f_{\text{in-cell}}(h,\ R) = \frac{1}{h}\frac{1}{h}\frac{1}{\pi R^2} Area$$

$$= \frac{1}{h}\frac{1}{h}\frac{1}{\pi R^2} [Area\,(\text{part}A) + Area\,(\text{part}B') + Area\,(\text{part}B'') + Area\,(\text{part}C)]$$

$$= \frac{1}{h}\frac{1}{h}\frac{1}{\pi R^2} [Area\,(\text{part}A) + 2 \times Area\,(\text{part}B') + Area\,(\text{part}C)]$$

$$= \frac{1}{h}\frac{1}{h}\frac{1}{\pi R^2} \left[\int_0^R \int_0^R \left(\int_0^{x_0} \sqrt{R^2 - x^2}\, dx + \int_0^{y_0} \sqrt{R^2 - y^2}\, dy + \int_0^{x_0} \text{Min}\left[y_0,\ \sqrt{R^2 - x^2}\right] dx + \frac{\pi R^2}{4} \right) dx_0\, dy_0 + 2 \times \int_R^h \int_0^R \left(\pi R^2 - \frac{1}{2} R^2 \left(2\,\text{ArcCos}\left[\frac{x_0}{R}\right] - \text{Sin}\left[2\,\text{ArcCos}\left[\frac{x_0}{R}\right]\right]\right)\right) dx_0\, dy_0 + \int_R^h \int_R^h \pi R^2\, dx_0\, dy_0 \right]$$

This expression evaluates to:

$$f_{\text{in-cell}}(h,\ R) = \frac{9\pi - 5}{12\pi} \times \frac{R^2}{h^2} + \frac{2}{3} \frac{(h - R)R}{h^2} \times \frac{3\pi - 2}{\pi} + \frac{(h - R)^2}{h^2}$$

Diagonal neighbors. To evaluate the fraction of sensors belonging to diagonal neighbors we use similar analysis. When a sensor is located at the upper right corner of the cell $1/4^{th}$ of the circle falls inside its diagonal neighbor (see fig. 12). If we move the sensor toward the center of the cell, the fraction of the circle falling inside its diagonal neighbor decreases. We further shift

coordinate origin to the center of the circle and we get the following expression:

$$f_{\text{diag}}(h,\ R) = \frac{1}{h}\frac{1}{h}\frac{1}{\pi R^2}\int_0^R\int_0^{\sqrt{R^2-x_0^2}}\int_{x_0}^{\sqrt{R^2-y_0^2}}\left(\sqrt{R^2-x^2}\ -\ y_0\right)dx\,dy_0\,dx_0$$

$$f_{\text{diag}}(h,\ R) = \frac{R^2}{8\,h^2\,\pi}$$

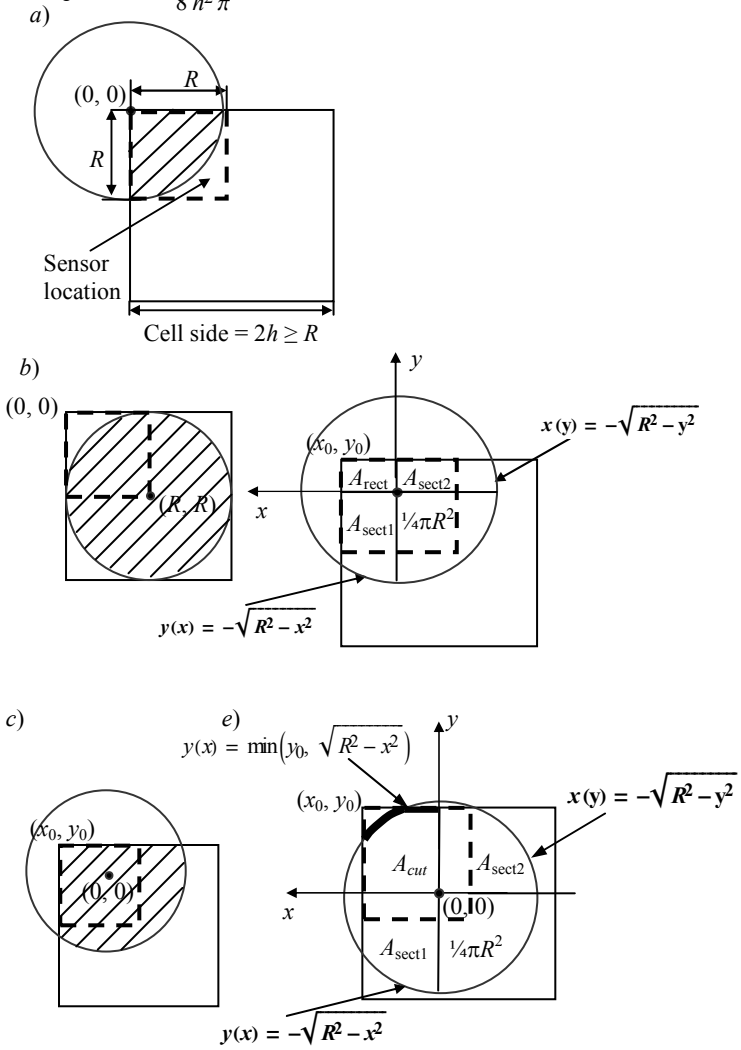

Fig 10. Analyzing the fraction of the communication range for a sensor in Part *A*:
$0 \le x_0 \le R$ and $0 \le y_0 \le R$ – the bounds of the region are shown as a dashed rectangle

a) 1/4th of the communication area falls inside the cell for a sensor located at the upper-left corner of the cell;

b) The whole communication circle falls inside the cell when a sensor is located at the right- bottom position of the region;

c) Area falling inside the cell for a sensor located at some arbitrary (x_0, y_0) coordinates;

d) An area for arbitrary (x_0, y_0) coordinates ($0 \le x_0 \le R$ and $0 \le y_0 \le R$): consists of four pieces: A_{rect} – an area of the rectangle, A_{sect1}, A_{sect2} – area of sectors, $¼\pi R^2$ – an area of the quarter of the circle;

e) An area for arbitrary (x_0, y_0) coordinates when upper-right corner of a square falls outside the communication circle.

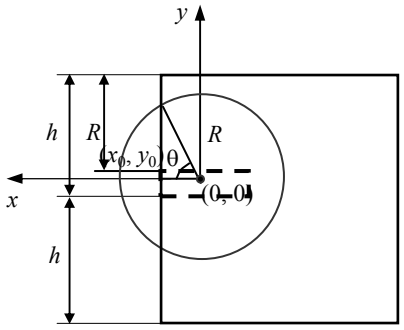

Fig 11. Analyzing the fraction of the communication range for a sensor in Part *B'*:
$0 \le x_0 \le R$ and $R \le y_0 \le h$ – the bounds of the region are shown as a dashed rectangle.

Area falling inside the cell is equal to the area of the circle minus an area of a segment

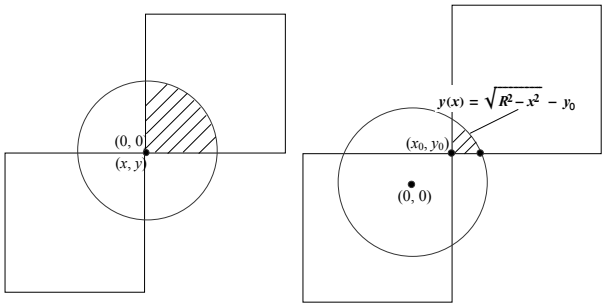

Fig 12. Analyzing fraction of communication range falling inside diagonal neighbor cell:

a) 1/4th of the communication area falls inside the diagonal neighbor cell for a sensor located at the upper-right corner of the cell;

b) An area falling inside the diagonal neighbor cell for some arbitrary (x_0, y_0) coordinates:

$x_0\ [0, R], y_0\ [0, \sqrt{R^2 - x_0^2}\ ]$