Int. J. Inf. Secur. (2006) 5(2): 105–114

SPECIAL ISSUE PAPER

Dibyendu Chakrabarti · Subhamoy Maitra · Bimal Roy

A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design

Abstract In this paper, combinatorial design followed by randomized merging strategy is applied to key pre-distribution in sensor nodes. A transversal design is used to construct a (v, b, r, k) configuration and then randomly selected blocks are merged to form the sensor nodes. We present detailed mathematical analysis of the number of nodes, number of keys per node and the probability that a link gets affected if certain number of nodes are compromised. The technique is tunable to user requirements and it also compares favourably with state of the art design strategies. An important feature of our design is the presence of more number of common keys between any two nodes. Further, we study the situation when properly chosen blocks are merged to form sensor nodes such that the number of intra-node common key is minimized. We present a basic heuristic for this approach and show that it provides slight improvement in terms of certain parameters than our basic random merging strategy.

Keywords Combinatorial design · Sensor network · Key pre-distribution · Random merging

1 Introduction

Recently secure communication among sensor nodes has become an active area of research [2, 4, 6, 7, 10, 11, 12]. One may refer to [9] for broader perspective in the area of sensor networks. Based on the architectural consideration, wireless sensor networks may be broadly classified into two categories viz. (i) Hierarchical Wireless Sensor Networks (HWSN) and (ii) Distributed Wireless Sensor Networks (DWSN). In HWSN, there is a pre-defined hierarchy among the participating nodes. There are three types of nodes in the descending order of capabilities: (a) base stations, (b) cluster heads, and (c) sensor nodes.

The sensor nodes are usually placed in the neighbourhood of the base station. Sometimes the network traffic (data) is collected by the cluster heads which in turn forward the traffic to the base station. There may be three different modes of data flow as follows: Unicast (sensor to sensor), multicast (group wise), broadcast (base station to sensor). However, it may be pointed out that the HWSN is best suited for applications where the network topology is known prior to deployment. On the other hand, there is no fixed infrastructure in the case of a DWSN and the network topology is unknown before the deployment. Once the nodes are scattered over the target area, the nodes scan their radio coverage area and find out their neighbours. In this case also, the data flow may be divided into three categories (as discussed above) with the only difference that the broadcast might take place between any two nodes. Unless mentioned otherwise, we shall always talk about DWSNs. Hence all the nodes are equal in their capabilities.

Consider a scenario where N number of sensor nodes are dropped from an airplane in the battlefield. Thus, the geographical positioning of the nodes cannot be decided a priori. However, any two nodes in radio frequency range are expected to be able to communicate securely. One option is to maintain different secret keys for each of the pairs. Then each of the nodes needs to store $N - 1$ keys. Given (i) the huge number of sensor nodes generally deployed, (ii) the memory constraint of the sensor nodes, this solution is not practical. On the other hand, on-line key exchange is not very popular till date since implementation of public key framework demands processing power at the higher end. Very recently implementations of ECC and RSA on 8-bit CPUs have been proposed [8]. Still a closer scrutiny of [8, Table 2, Sect. 3.3] reveals that the algorithms execute in seconds (the range being 0.43–83.26 s); whereas the key pre-distribution just involves the calculation of inverse of an integer modulo a prime number, which is bound to be much faster than the former.

This paper is an extended and revised version of the paper presented in *8th Information Security Conference, ISC'05*, pp. 89–103, Lecture Notes in Computer Science, vol. 3650, Springer Verlag.

D. Chakrabarti (✉) · S. Maitra · B. Roy
Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700108, India
E-mail: {dibyendu_r, subho, bimal}@isical.ac.in

Hence, key pre-distribution to each of the sensor nodes before deployment is a thrust area of research and the most used mathematical tool for key pre-distribution is combinatorial design. Each of the sensor nodes contains M many keys and each key is shared by Q many nodes, (thus fixing M and Q) such that the encrypted communication between two nodes may be decrypted by at most $Q-2$ other nodes if they fall within the radio frequency range of the two communicating nodes. Similarly, one node can decrypt the communication between any two of at most $M(Q-1)$ nodes if it lies within the radio frequency range of all the nodes who share a key with it.

Let us present an exact example from [11]. Take $N = 2401$, $M = 30$, $Q = 49$. The parameters are obtained using a Transversal Design (for a basic introduction to Transversal Designs, refer to [15, p. 133]). It has been shown that two nodes share either 0 or 1 key. In this case, $M(Q-1)$ gives the number of nodes with which one node can communicate. The expected number of keys that is common between any two nodes is $\frac{M(Q-1)}{N-1} = 0.6$, (in [11], this is called the probability that two nodes share a common key). Further, it can be checked that if two nodes do not share a common key, then they may communicate via another intermediate node. Let nodes ν_i, ν_j do not share a common key, but ν_i, ν_k share a common key and ν_k, ν_j share a common key, i, j, k are all distinct. Hence the secret communication between ν_i and ν_k needs a key (encrypted by ν_i, decrypted by ν_k) and that between ν_k and ν_j needs another secret key (encrypted by ν_k, decrypted by ν_j). It has been shown in [11] that the communication between two nodes is possible in almost 0.99995 proportion of cases (this is based on some assumptions on the geometric distribution of nodes, which we do not use for our analysis). However, the following problems are immediate:

1. Communication between any two nodes in 60% of the cases will be in one step (no involvement of any other node), but the communication between any two of them needs two steps for the rest 40% of the cases, making the average of 1.4 steps in each communication. This is an overhead. Thus, we need a design where we can guarantee that there is a common key between any two nodes.
2. The direct communication between any two nodes can be decrypted by at most $Q-2$ other nodes. However, if one takes the help of a third intermediate node, then the communication can be decrypted by at most $2(Q-2)$ nodes. Thus, any communication can be decrypted by at most $1.4(Q-2)$ many nodes on an average.
3. In an adversarial situation, if s many nodes are compromised, it has been shown that $1-(1-\frac{Q-2}{N-2})^s$ proportion of links becomes unusable. In this specific design, for $s = 10$, out of 2401 nodes, the proportion of unusable links becomes as high as 17.95%.

The solution to all these problems is based on the fact that we need to increase the number of common keys between any two nodes. The issues at this point are as follows:

1. The number of keys to be stored in each node will clearly increase. So one needs to decide the availability of storage space. In [11, p. 4], it has been commented that storing 150 keys in a sensor node may not be practical. On the other hand, in [6, p. 47], [10, Sect. 5.2], scenarios have been described with 200 many keys. If one considers 4 Kbytes of memory space for storing keys in a sensor node, then choosing 128-bit key (16 byte), it is possible to accommodate 256 many keys.
2. It is not easy to find out combinatorial designs with prespecified number of common keys (say for example 5) among any two nodes for key pre-distribution [5, 14]. Consider the following technique. Generally, a sensor node corresponds to a block in combinatorial design [2, 11]. Here we merge a few blocks to get a sensor node. Thus, the key space at each node gets increased and the number of common keys between any two nodes can also be increased to the desired level. It will be shown that this technique provides a much better control over the design parameters in key pre-distribution algorithms.
3. Further, it is also shown that by this random merging strategy, one gets more flexible parameters than [11].

Thus, the goal in this paper is to present a randomized block merging based design strategy that originates from Transversal Design. We differ from the existing works where it is considered that any two nodes will have either 0 or 1 common key and motivate a design strategy with more number of common keys. This is important from resiliency consideration in an adversarial framework since if certain nodes are compromised, the proportion of links that becomes unusable can be kept low, i.e. the connectivity of the network is less disturbed.

The computation to find out a common key is also shown to be of very low time complexity under this paradigm as explained in Sect. 5. Note that Blom's scheme [1] has been extended in recent works for key pre-distribution in wireless sensor networks [6, 10]. The problem with these kinds of schemes is the use of several multiplication operations (as example see [6, Sect. 5.2]) for key exchange.

The randomized key pre-distribution is another strategy in this area [7]. However, the main motivation is to maintain the connectivity (possibly with several hops) in the network. As example [7, Sect. 3.2], a sensor network with 10,000 nodes has been considered and to maintain the connectivity, it has been calculated that it is enough, if one node can communicate with only 20 other nodes. Note that the communication between any two nodes may require a large number of hops. However, as we discussed earlier, only the connectivity criterion (with too many hops) can not suffice in an adversarial condition. Further, in such a scenario, the key agreement between two nodes requires exchange of the key indices.

The use of combinatorial and probabilistic design (also a combination of both – termed as hybrid design) in the context of key distribution has been proposed in [2]. In this case also, the main motivation was to have low number of common keys as in [11]. On the other hand, we propose the idea of good number of common keys between any two nodes. The novelty of our approach is to start from a combinatorial design and then apply a probabilistic extension in the form of random merging of blocks to form the sensor nodes and in this case there is good flexibility in adjusting the number of common keys between any two nodes.

Note that in our approach, we first consider the block merging strategy in a completely randomized fashion. In such a case there is a possibility that the constituent blocks (which are merged to form a sensor node) may share common keys among themselves. This is a loss in terms of the connectivity in the designed network as no shared key is needed since there is no necessity for 'intra-node communication'. Thus, we further consider a merging strategy towards minimizing the number of common keys among the blocks that are being merged. We present a heuristic for this and it works better than our initial random merging strategy. The scheme is a hybrid one as combinatorial design is followed by a heuristic.

The organization of the paper is as follows. We present some preliminaries of combinatorial design and the main idea of [11] in the following section. Section 3 presents the results related to the proposed random merging strategy of the blocks to form a node. Further, some heuristic improvements are suggested in Sect. 4. The key exchange protocol between any two nodes is described in Sect. 5. Section 6 concludes the paper.

2 Preliminaries

2.1 Basics of combinatorial design

Let A be a finite set of subsets (also known as blocks) of a set X. A *set system* or *design* is a pair (X, A). The degree of a point $x \in X$ is the number of subsets containing the point x. If all subsets/blocks have the same size k, then (X, A) is said to be uniform of rank k. If all points have the same degree r, (X, A) is said to be regular of degree r.

A regular and uniform set system is called a $(v, b, r, k)-1$ design, where $|X| = v$, $|A| = b$, r is the degree and k is the rank. The condition $bk = vr$ is necessary and sufficient for existence of such a set system. A $(v, b, r, k)-1$ design is called a (v, b, r, k) configuration, if any two distinct blocks intersect in zero or one point.

A (v, b, r, k, λ) BIBD is a $(v, b, r, k)-1$ design in which every pair of points occurs in exactly λ many blocks. A (v, b, r, k) configuration having deficiency $d = v-1-r(k-1) = 0$ exists if and only if a $(v, b, r, k, 1)$ BIBD exists.

Let g, u, k be positive integers such that $2 \le k \le u$. A group-divisible design of type g^u and block size k is a triple $(X, \mathcal{H}, \mathcal{A})$, where X is a finite set of cardinality gu, $\mathcal{H}$ is a partition of X into u parts/groups of size g, and $\mathcal{A}$ is a set of subsets/blocks of X. The following conditions are satisfied in this case:

1. $|H \bigcap A| \le 1 \ \forall H \in \mathcal{H}, \ \forall A \in \mathcal{A}$,
2. every pair of elements of X from different groups occurs in exactly one block in $\mathcal{A}$.

A Transversal Design $TD(k, n)$ is a group-divisible design of type n^k and block size k. Hence, $H \cap A = 1 \ \forall H \in \mathcal{H}, \ \forall A \in \mathcal{A}$.

Let us now describe the construction of a transversal design. Let p be a prime power and $2 \le k \le p$. Then there exists a $TD(k, p)$ of the form $(X, \mathcal{H}, \mathcal{A})$ where $X = \mathbb{Z}_k \times \mathbb{Z}_p$. For $0 \le x \le k-1$, define $H_x = \{x\} \times \mathbb{Z}_p$ and $\mathcal{H} = \{H_x : 0 \le x \le k-1\}$.

For every ordered pair $(i, j) \in \mathbb{Z}_p \times \mathbb{Z}_p$, define a block $A_{i,j} = \{x, (ix + j) \bmod p : 0 \le x \le k-1\}$. In this case, $\mathcal{A} = \{A_{i,j} : (i, j) \in \mathbb{Z}_p \times \mathbb{Z}_p\}$. It can be shown that $(X, \mathcal{H}, \mathcal{A})$ is a $TD(k, p)$.

Now let us relate a $(v = kr, b = r^2, r, k)$ configuration with sensor nodes and keys. X is the set of $v = kr$ number of keys distributed among $b = r^2$ number of sensor nodes. The nodes are indexed by $(i, j) \in \mathbb{Z}_r \times \mathbb{Z}_r$ and the keys are indexed by $(i, j) \in \mathbb{Z}_k \times \mathbb{Z}_r$. Consider a particular block $A_{\alpha,\beta}$. It will contain k number of keys $\{(x, (x\alpha+\beta) \bmod r) : 0 \le x \le k-1\}$. Here $|X| = kr = v$, $|\mathcal{H}_x| = r$, the number of blocks in which the key (x, y) appears for $y \in \mathbb{Z}_r$, $|A_{i,j}| = k$, the number of keys in a block. For more details on combinatorial design refer to [11, 15].

Note that if r is a prime power, we will not get an inverse of $x \in \mathbb{Z}_r$ when $\gcd(x, r) > 1$. This is required for key exchange protocol (see Sect. 5). So basically we should consider the field $GF(r)$ instead of the ring $\mathbb{Z}_r$. However, there is no problem when r is a prime by itself. In this paper we generally use $\mathbb{Z}_r$ since in our examples we consider r to be prime.

2.2 Lee–Stinson approach [11]

Consider a (v, b, r, k) configuration (which is in fact a (rk, r^2, r, k) configuration). There are $b = r^2$ many sensor nodes, each containing k distinct keys. Each key is repeated in r many nodes. Also v gives the total number of distinct keys in the design. One should note that $bk = vr$ and $v - 1 > r(k-1)$. The design provides 0 or 1 common key between two nodes. The design $(v = 1470, b = 2401, r = 49, k = 30)$ has been used as an example in [11]. The important parameters of the design are as follows:

1. *Expected number of common keys between two nodes:* This value is $p_1 = \frac{k(r-1)}{b-1} = \frac{k}{r+1}$. In the given example, $p_1 = \frac{30}{49+1} = 0.6$.
2. *Consider an intermediate node:* There is a good proportion of pairs (40%) with no common key, and two such nodes will communicate through an intermediate node. Assuming a random geometric deployment, the example shows that the expected proportion such that two nodes are able to communicate either directly or through an intermediate node is as high as 0.99995.
3. *Resiliency:* Under adversarial situation, one or more sensor nodes may get compromised. In that case, all the keys

present in those nodes cannot be used for secret communication any longer, i.e. given the number of compromised nodes, one needs to calculate the proportion of links that cannot be used further. The expression for this proportion is

$$fail(s) = 1 - \left(1 - \frac{r-2}{b-2}\right)^s,$$

where s is the number of nodes compromised. In this particular example, $fail(10) \approx 0.17951$. That is, given a large network comprising as many as 2401 nodes, even if only 10 nodes are compromised, almost 18% of the links become unusable.

3 Our strategy: merging blocks in combinatorial design

We use the concept of merging blocks to form a sensor node. Initially we do not specify any merging strategy and consider that blocks will be merged randomly. In this direction we present the following technical result.

Theorem 1 *Consider a (v, b, r, k) configuration with $b = r^2$. We merge z many randomly selected blocks to form a sensor node. Then*

1. *There will be $N = \lfloor \frac{b}{z} \rfloor$ many sensor nodes.*
2. *The probability that any two nodes share no common key is $(1 - p_1)^{z^2}$, where $p_1 = \frac{k}{r+1}$.*
3. *The expected number of keys shared between two nodes is $z^2 p_1$.*
4. *Each node will contain M many distinct keys, where $zk - \binom{z}{2} \leq M \leq zk$. The average value of M is $\hat{M} = zk - \binom{z}{2}\frac{k}{r+1}$.*
5. *The expected number of links in the merged system is*
$$\hat{L} = \left(\binom{r^2}{2} - \left\lfloor \frac{r^2}{z} \right\rfloor \binom{z}{2}\right)\frac{k}{r+1} - (r^2 \bmod z)k.$$
6. *Each key will be present in Q many nodes, where $\lceil \frac{r}{z} \rceil \leq Q \leq r$. The average value of Q is $\hat{Q} = \frac{1}{kr}(\lfloor \frac{b}{z} \rfloor)(zk - \binom{z}{2}\frac{k}{r+1})$.*

Proof The first item is easy to see.

Since the blocks are merged randomly, any two sensor nodes will share no common key if and only if none of the keys in z blocks constituting one sensor node are available in the z blocks constituting the other sensor node. Thus, there are z^2 many cases where there are no common keys. As we have considered random distribution in merging z blocks to form a node, under reasonable assumption (corroborated by extensive simulation studies), all these z^2 events are independent. Note that p_1 is the probability that two blocks share a common key. Hence the proof of the second item.

The number of common keys between two blocks approximately follows binomial distribution. The probability that two blocks share i many common keys is given by $\binom{z^2}{i} p_1^i (1 - p_1)^{z^2 - i}$, $0 \leq i \leq z^2$. Thus, the mean of the distribution is $z^2 p_1$ which proves the third item.

For the fourth item, note that each block contains k many distinct keys. When z many blocks are merged, then there may be at most $\binom{z}{2}$ common keys among them. Thus, the number of distinct keys M per sensor node will be in the range $zk - \binom{z}{2} \leq M \leq zk$. The average number of common keys between two nodes is $\frac{k}{r+1}$. So the average value of M is $zk - \binom{z}{2}\frac{k}{r+1}$.

Consider that z blocks are merged to form a node, i.e. given a $(v = rk, b = r^2, r, k)$ configuration we get $\lfloor \frac{r^2}{z} \rfloor$ many sensor nodes. The total number of links was $\binom{r^2}{2}\frac{k}{r+1}$ before the merging of blocks. For each of the nodes (a node is z many blocks merged together), $\binom{z}{2})\frac{k}{r+1}$ many links become intra-node links and totally, there will be a deduction of $\lfloor \frac{r^2}{z} \rfloor \binom{z}{2}\frac{k}{r+1}$ links (to account for the intra-node links) on an average. Further, as we use $\lfloor \frac{r^2}{z} \rfloor$ many sensor nodes, we discard $(r^2 \bmod z)$ number of blocks, which contribute to $(r^2 \bmod z)k$ many links. There will be a deduction for this as well. Thus, the expected number of links in the merged system is

$$\left(\binom{r^2}{2} - \left\lfloor \frac{r^2}{z} \right\rfloor \binom{z}{2}\right)\frac{k}{r+1} - (r^2 \bmod z)k.$$

This proves the fifth item.

Note that a key will be present in r many blocks. Thus, a key may be exhausted as early as after being used in $\lceil \frac{r}{z} \rceil$ many sensor nodes. On the other hand a key may also be distributed to a maximum of r many different nodes. Hence the number of distinct nodes Q corresponding to each key is in the range $\lceil \frac{r}{z} \rceil \leq Q \leq r$. Now we try to find out the average value of Q, denoted by $\hat{Q}$. Total number of distinct keys in the merged design does not change and is also kr. Thus, $\hat{Q} = \frac{N\hat{M}}{kr} = \frac{1}{kr}(\lfloor \frac{b}{z} \rfloor)(zk - \binom{z}{2}\frac{k}{r+1})$. This proves the sixth item. □

3.1 Calculating $fail(s)$ when a block is considered as a node (no merging)

The expression $fail(s)$, the probability that a link become unusable, if s many nodes are compromised, has been calculated in the following way in [11]. Consider that there is a common secret key between the two nodes N_i, N_j. Let N_h be a compromised node.Now the key that N_i, N_j share is also shared by $r - 2$ other nodes. The probability that N_h is one of those $r - 2$ nodes is $\frac{r-2}{b-2}$. Thus, the probability that compromise of s many nodes affect a link is approximately $1 - (1 - \frac{r-2}{b-2})^s$. Given the design $(v = 1470, b = 2401, r = 49, k = 30)$ and $s = 10, fail(10) \approx 0.17951$.

We calculate this expression in a little different manner. Given $b = r^2$ many nodes, the total number of links is $\binom{r^2}{2}\frac{k}{r+1}$. Now compromise of one node reveals k many

Table 1 Calculation of *fail*(s) and *Fail*(s)

s	1	2	3	4	5	6	7	8	9	10
fail(s)	0.019591	0.038799	0.057631	0.076093	0.094194	0.111940	0.129338	0.146396	0.163119	0.179515
Fail(s)	0.020408	0.040408	0.060000	0.079184	0.097959	0.116327	0.134286	0.151837	0.168980	0.185714
Expt.	0.020406	0.040609	0.059986	0.078376	0.096536	0.117951	0.135109	0.151639	0.165508	0.184885

keys. Each key is repeated in r many nodes, i.e. it is being used in $\binom{r}{2}$ many links. Thus, if one key is revealed, it disturbs the following proportion of links:

$$\frac{\binom{r}{2}}{\binom{r^2}{2}\frac{k}{r+1}} = \frac{1}{kr}.$$

Now s many nodes contain $ks - \binom{s}{2}\frac{k}{r+1}$ many *distinct* keys on an average. This is because there are $\binom{s}{2}$ many pairs of nodes and a proportion of $\frac{k}{r+1}$ of them will share a common key. Thus, in our calculation, on an average

$$Fail(s) = \frac{ks - \binom{s}{2}\frac{k}{r+1}}{kr} = \frac{s}{r}\left(1 - \frac{s-1}{2(r+1)}\right).$$

Note that to distinguish the notation we use *Fail*(s) instead of *fail*(s) in [11]. Note that considering the design $(v = 1470, b = 2401, r = 49, k = 30)$, we tabulate the values of *fail*(s), *Fail*(s) and experimental data (average of 100 runs for each s) regarding the proportion of links that cannot be used after compromise of s many nodes. The results look quite similar. However, it may be pointed out that our approximation is in better conformity with the experimental values (see Table 1) than that of [11], which looks a bit underestimated.

3.2 Calculation of *Fail*(s) when more than one blocks are merged

Let N_a and N_b be two given nodes. Define two events E and F as follows:

1. E: N_a and N_b are disconnected after the failure of s number of nodes,
2. F: N_a and N_b were connected before the failure of those s nodes.

The sought for quantity is

$$Fail(s) = P(E \mid F) = \frac{P(E \cap F)}{P(F)}.$$

Let X be the random variable denoting the number of keys between N_a and N_b and following the proof of Theorem 1(2), we assume that X follows $B(z^2, \frac{k}{r+1})$. Thus,

$$P(F) = P(X > 0) = 1 - P(X = 0) = 1 - \left(1 - \frac{k}{r+1}\right)^2.$$

Next define two sets of events:

1. E_{1i}: i number of keys (shared between N_a and N_b) are revealed consequent upon the failure of s nodes,
2. E_{2i} : i number of keys are shared between N_a and N_b.

Let $E_i = E_{1i} \cap E_{2i}$ for $i = 1, 2, \ldots, z^2$. So, $E_i \cap E_j = \emptyset$ for $0 \leq i \neq j \leq z^2$. As $E \cap F = \cup_{i=1}^{z^2} E_i$, we have $P(E \cap F) = P(\cup_{i=1}^{z^2} E_i) = \sum_{i=1}^{z^2} P(E_i) = \sum_{i=1}^{z^2} P(E_{1i}|E_{2i})P(E_{2i})$ and also $P(E_{2i}) = \binom{z^2}{i}(\frac{k}{r+1})^i (1 - \frac{k}{r+1})^{z^2-i}$.

Now we estimate $P(E_{1i} \mid E_{2i})$ by hypergeometric distribution. Consider the population (of keys) of size kr and γ number of defective items (the number of distinct keys revealed). We shall draw a sample of size i (without replacement) and we are interested in the event that all the items drawn are defective.

Note that γ is estimated by the average number of distinct keys revealed, i.e. $\gamma = szk(1 - \frac{sz-1}{2(r+1)})$. So $P(E_{1i} \mid E_{2i}) = \frac{\binom{\gamma}{i}}{\binom{kr}{i}}, i = 1, 2, \ldots, z^2$.

Finally, $P(E|F) = \frac{P(E \bigcap F)}{P(F)}$

$$= \frac{\sum_{i=1}^{z^2} \frac{\binom{\gamma}{i}}{\binom{kr}{i}} \binom{z^2}{i} (\frac{k}{r+1})^i (1 - \frac{k}{r+1})^{z^2-i}}{1 - (1 - \frac{k}{r+1})^2}.$$

The estimate γ is a quadratic function of s and hence is not an increasing function (though in reality, it should be an increasing function of s $\forall s$). That is why *Fail*(s) increases with s as long as γ increases with s. Given $\gamma = szk(1 - \frac{sz-1}{2(r+1)})$, it can be checked that γ is increasing for $s \leq \frac{2r+3}{2z}$. As we are generally interested in the scenarios where a small proportion of nodes are compromised, this constraint on the number of compromised nodes s is practical.

Based on the above discussion, we have the following theorem.

Theorem 2 *Consider a (v, b, r, k) configuration. A node is created by random merging of z many nodes. For $s \leq \frac{2r+3}{2z}$,*

$$Fail(s) \approx \frac{\sum_{i=1}^{z^2} \frac{\binom{\gamma}{i}}{\binom{kr}{i}} \binom{z^2}{i} (\frac{k}{r+1})^i (1 - \frac{k}{r+1})^{z^2-i}}{1 - (1 - \frac{k}{r+1})^2},$$

where

$$\gamma = szk\left(1 - \frac{sz-1}{2(r+1)}\right).$$

Table 2 Comparison of $Fail(s)$ values in different cases (a node is formed by merging four blocks)

s	1	2	3	4	5	6	7	8	9	10
$Fail(s)$ (Theorem 3)	0.020408	0.040408	0.060000	0.079184	0.097959	0.116327	0.134286	0.151837	0.168980	0.185714
$Fail(s)$ (Theorem 2)	0.022167	0.044369	0.066527	0.088560	0.110385	0.131917	0.153069	0.173756	0.193891	0.213388
Expt. (random)	0.022987	0.045345	0.068904	0.090670	0.114853	0.135298	0.158633	0.181983	0.203342	0.222167
$Expt.(heuristic)$	0.022414	0.041457	0.066885	0.091181	0.106597	0.132854	0.156337	0.177984	0.200212	0.218968

It may be mentioned that while estimating $P(E_{1i} \mid E_{2i})$ by $\frac{\binom{\gamma}{i}}{\binom{kr}{i}}$, we are allowing a higher quantity in the denominator. The number of distinct keys revealed is under the restriction that the keys are distributed in s distinct blocks. However, the denominator is the expression for choosing i number of distinct keys from a collection of kr keys without any restriction. As a consequence, the resulting probability values will be under estimated, though the experimental results reveal that the difference is not significant at all (see Table 2).

Note that in Theorem 2, there is a restriction on s. Next we present another approximation of *Fail*(s) as follows where such a restriction is not there. However, the approximation of Theorem 3 is little further than that of Theorem 2 from the experimental results (see Table 2).

Theorem 3 *Consider a* $(v = kr, b = r^2, r, k)$ *configuration. A node is prepared by merging* $z > 1$ *nodes. Then in terms of design parameters, Fail*$(s) \approx$

$$\frac{1}{1-(1-\frac{k}{r+1})^{z^2}} \sum_{i=1}^{z^2} \binom{z^2}{i} \left(\frac{k}{r+1}\right)^i \left(1-\frac{k}{r+1}\right)^{z^2-i} \pi^i,$$

where

$$\pi = szk\left(1-\frac{sz-1}{2(r+1)}\right)\frac{\hat{Q}(\hat{Q}-1)}{2\hat{L}}.$$

Proof Compromise of one node reveals $\hat{M}$ many keys on an average. Thus, there will be $s\hat{M}$ many keys. Further, between any two nodes, $z^2\frac{k}{r+1}$ keys are common on an average. Thus, we need to subtract $\binom{s}{2}z^2\frac{k}{r+1}$ many keys from $s\hat{M}$ to get the number of distinct keys. Thus, the number of distinct keys in s many merged nodes is $= s\hat{M}-\binom{s}{2}z^2\frac{k}{r+1} = s(zk-\binom{z}{2}\frac{k}{r+1})-\binom{s}{2}z^2\frac{k}{r+1} = szk(1-\frac{sz-1}{2(r+1)})$.

We have $N = \lfloor\frac{b}{z}\rfloor$ many sensor nodes, and $\hat{L} = (\binom{r^2}{2} - \lfloor\frac{r^2}{z}\rfloor\binom{z}{2})\frac{k}{r+1} - (r^2 \bmod z)k$ many average number of total links. Each key is repeated in $\hat{Q}$ many nodes on an average, i.e. it is being used in $\frac{\hat{Q}(\hat{Q}-1)}{2}$ many links. Thus, if one key is revealed that disturbs $\frac{\hat{Q}(\hat{Q}-1)}{2\hat{L}}$ many links on an average. Hence compromise of one key disturbs $\frac{\frac{Q(Q-1)}{2}}{\hat{L}}$ proportion of links. Hence, compromise of s nodes disturbs $\pi = szk(1-\frac{sz-1}{2(r+1)})\frac{\hat{Q}(\hat{Q}-1)}{2\hat{L}}$ proportion of links on an average. Thus, we can interpret π as the probability that one link is affected after compromise of s many merged nodes.

Now the probability that there are i many links between two nodes given at least one link exists between them is $\frac{1}{1-(1-\frac{k}{r+1})^{z^2}}\binom{z^2}{i}(\frac{k}{r+1})^i(1-\frac{k}{r+1})^{z^2-i}$. Further, the probability that all those i links will be disturbed due to compromise of s nodes is π^i. Hence *Fail*(s)

$$= \frac{1}{1-(1-\frac{k}{r+1})^{z^2}} \sum_{i=1}^{z^2} \binom{z^2}{i} \left(\frac{k}{r+1}\right)^i \left(1-\frac{k}{r+1}\right)^{z^2-i} \pi^i.$$

□

The following example illustrates our approximations vis-a-vis the experimental results. Consider a $(v = 101 \cdot 7, b = 101^2, r = 101, k = 7)$ configuration and merging of $z = 4$ blocks to get a node. Thus, there will be 2, 550 many nodes. In such a situation we present the proportion of links disturbed if s many $(1 \le s \le 10)$ nodes are compromised, i.e. this can also be seen as the probability that two nodes get disconnected which were connected earlier (by one or more links). In Table 2 we present the values that we get from Theorem 3, Theorem 2 and also experimental results which are the average of 100 runs.

3.3 Comparison with [11]

In the example presented in [11], the design $(v = 1470, b = 2401, r = 49, k = 30)$ has been used to get $N = 2401$, $M = 30$, $Q = 49$, $p_1 = 0.6$, $1 - p_1 = 0.4$.

Now we consider the design $(v = 101 \cdot 7 = 707, b = 101^2 = 10201, r = 101, k = 7)$. Note that in this case $p_1 = \frac{k}{r+1} = \frac{7}{102}$. We take $z = 4$. Thus, $N = \lfloor\frac{10201}{4}\rfloor = 2550$. Further, the probability that two nodes will not have a common key is $(1-\frac{7}{102})^{16} = 0.32061$. Note that this is considerably lesser (better) than the value 0.4 presented in [11] under a situation where the number of nodes is greater $(2550 > 2401)$ and number of keys per node is lesser $(28 < 30)$ in our case. Thus, our strategy is clearly more efficient than that of [11] in this respect. On the other hand, the *Fail*(s) value is worse in our case than what has been achieved in [11]. In Table 3, for our approaches, we present the experimental values which are average over 100 runs. For the time being let us concentrate on the comparison between our contribution in this Sect. 3 and the idea presented in [11]. In the next Sect. 4, we will present a better idea and the result of that is also included in Table 3 for brevity.

Table 3 Comparison with an example presented in [11]

Comparison	Random merging Sect. 3	Heuristic Sect. 4	[11]
Number of nodes	2, 550	2, 550	2, 401
Number of keys per node	≤ 28	≤ 28	30
Probability that two nodes do not share a common key	0.320555	0.30941	0.4
Fail(s), for $s = 10$	0.222167	0.218968	0.185714

The comparison in Table 3 is only to highlight the performance of our design strategy with respect to what is described in [11] and that is why we present a design with average number of common keys between any two nodes ≤ 1. However, we will present a practical scenario in the next subsection where there are more number (≥ 5) of common keys (on an average) between any two nodes and consequently the design achieves much less $Fail(s)$ values.

One more important thing to mention is that we consider the average case analysis for our strategy. The worst case situation will clearly be worse than the average case, but that is not of interest in this context as we will first try to get a merging configuration which is close to the average case. As this is done in preprocessing stage, we may go for more than one attempts for the configuration and it is clear that in a few experiments, we will surely get a configuration matching the average case result. On the other hand, it is very important to identify the best case as this will provide a solution better than the average case. However, this is open at this point of time.

The strength of our scheme is in the presence of several common keys between two nodes, which in fact makes it more resilient. Of course, this is at the cost of an obvious increase in number of keys in each node by a factor of z. The example presented in Sect. 3.3 and Sect. 3.4 illustrate this fact. In Sect. 3.3, we deliberately allowed a very low number of common keys (so that the node size is comparable to that of [11]) and hence the negative resiliency measure $Fail(s)$ increased slightly. In what follows, we demonstrate that with an increase in the node capacity, the negative resiliency measure $Fail(s)$ assumes a negligible value.

3.4 A practical design with more than one keys (on average) shared between two nodes

We start with the idea that a node can contain 128 keys and as we like to compare the scenario with [11], we will consider the number of sensor nodes ≥ 2401, as it has been used in the examples in [11].

Consider a $(v = rk, b = r^2, r = 101, k = 32)$ configuration. If one merges $z = 4$ blocks (chosen at random) to construct a node, the following scheme is obtained (refer to Theorems 1 and 2).

1. There will be $\lfloor \frac{10201}{4} \rfloor = 2550$ sensor nodes.
2. The probability that two nodes do not share a common key is approximately $(1 - \frac{32}{102})^{16} = 0.0024$.
3. Expected number of keys shared between two nodes $= \frac{16 \cdot 32}{102} \geq 5$.
4. Each node will contain on an average $\hat{M} = 4 \times 32 - \binom{4}{2}\frac{32}{102} \approx 126$ many distinct keys and at most 128 many keys.
5. $Fail(10) = 0.019153 \approx 2\%$ and $Fail(25) = 0.066704 \approx 7\%$.

This example clearly uses more keys (≤ 128) per sensor node than the value 30 in the example of [11]. Note that directly from a (v, b, r, k) configuration, it is not possible to have $k > r$. However, in a merged system that is always possible. Moreover, the average number of keys shared between any two nodes is ≈ 5. It is not easy to get a combinatorial design [15] to achieve such a goal directly. This shows the versatility of the design proposed by us.

4 A Heuristic: merging blocks attempting to minimize the number of intra node common keys

So far we have used the concept of merging blocks to form a sensor node without any constraints on how the blocks will be chosen to form a node. Now we add the constraint that the blocks that will be merged to form a node such that the number of common keys between two blocks of the same node is minimized (the best case is if the number is zero). For this we present the following heuristic.

Heuristic 1

```
1. $flag = true$; $count = 0$; all the blocks are marked as unused;
2. an array $node[\ldots]$ is available, where each element of the array can store $z$ many blocks;
3. while($flag$){
   (a) choose a random block, mark it as used and put it in $node[count]$;
   (b) for ($i = 1; i < z; i++$){
       i. search all the unused blocks in random fashion and put the first available one in $node[count]$ which has no common key with the existing blocks already in $node[count]$;
       ii. mark this block as used;
       iii. if such a block is not available then break the for loop and assign $flag = false$;
   (c) } (end for)
   (d) if $flag = true$ then $count = count + 1$;
4. } (end while)
5. report that count many nodes are formed such that there is no intra node connectivity.
```

6. *for rest of the* $(r^2 - count \cdot z)$ *many blocks, merge z blocks randomly to form a node (they may have intra node connectivity) to get* $(\lfloor \frac{r^2}{z} \rfloor - count)$ *many extra nodes; this constitutes the initial configuration.*
7. *assign the initial configuration to current configuration and run step 8 for i many iterations.*
8. *make m many* ***moves*** *(explained below) on the current configuration and choose the one that gives rise to the maximum increase in connectivity; update the current configuration with this chosen one.*

We define a **move** as follows:

1. start **move**;
2. copy the current configuration in a temporary configuration and work on the temporary configuration;
3. from the list of pairs of nodes sharing more than one common keys, select one pair of nodes randomly; call them a and b;
4. from the list of pairs of nodes sharing no common key, select one pair of nodes randomly; call them c and d.
5. select one block each from a and b (say block α from node a and block β from node b) and remove them such that α and β intersect each other and nodes a and b are still connected after the removal of α, β, respectively; if this condition is not satisfied then go to step 9;
6. select one block each from nodes c and d and remove them; let the removed blocks be γ and δ, respectively;
7. put γ in a, δ in b, α in c and β in d;
8. store this temporary configuration in some container;
9. end **move**.

In Heuristic 1 we use a simple hill climbing technique and for experimental purposes we took $m = 100, i = 100$. It will be encouraging to apply more involved meta heuristic techniques in step 8 of Heuristic 1. This we recommend for future research.

Note that in [3], we have considered only up to step 5 of Heuristic 1. It is very clear that given (v, b, r, k) configuration with $b = r^2$, if one merges z many blocks to get each node then the maximum possible nodes that are available could be $N \leq \lfloor \frac{b}{z} \rfloor$. However, it is not guaranteed that given any configuration one can really achieve the upper bound $\lfloor \frac{b}{z} \rfloor$ with the constraint that the blocks constituting a node cannot have any common key among themselves. Using Heuristic 1 up to step 5, one can use all the blocks in some cases, but sometimes it may not be possible also (see details in [3]). That is the reason we go for step 6 for merging the rest of the blocks where we remove the constraints that no two blocks of a node can have a common key.

The following example illustrates the experimental results and we show that using this technique we get better (lower) $Fail(s)$ value than Sect. 3 as evident from the last row of Table 2. Consider a $(v = 101 \cdot 7, b = 101^2, r = 101, k = 7)$ configuration and merging of $z = 4$ blocks to get a node. Thus, there will be 2, 550 many nodes. In such a situation we present the proportion of links disturbed if s many $(1 \leq s \leq 10)$ nodes are compromised, i.e. this can also be seen as the probability that two nodes get disconnected which were connected earlier (by one or more links).

4.1 Experimental results with this heuristic

Let us refer to Table 3 for the comparison. As usual, we consider the $(v = 101 \cdot 7 = 707, b = 101^2 = 10201, r = 101, k = 7)$ configuration to attain a comparable design after merging. Note that in this case $p_1 = \frac{k}{r+1} = \frac{7}{102}$. We take $z = 4$. Thus, $N = \lfloor \frac{10201}{4} \rfloor = 2550$. Considering the binomial distribution presented in Theorem 1(3), the theoretical probability that two nodes will not have a common key is $(1 - \frac{7}{102})^{16} = 0.32061$. Experimentally with 100 runs we find the average value as 0.30941 which is less (better) than the theoretically estimated value and also the experimental value 0.320555 as explained in Sect. 3 under the same experimental set up. Note that this is considerably lesser than the value 0.4 presented in [11]. The average number of common keys between any two nodes is $z^2 p_1 = z^2 \frac{k}{r+1} = 16.7/102 = 1.098039$. Experimentally with 100 runs we get it as 1.098362 on an average which is a higher (improved) value than the theoretical estimate and also the experimental value 1.098039 as given in Sect. 3 under the same experimental set up. Further, note that the last row of the Table 2 provides better (lesser) $Fail(s)$ values available from the heuristic than the random search.

4.2 More keys shared between two nodes

As in Sect. 3.4, consider a $(v = rk, b = r^2, r = 101, k = 32)$ configuration. If one merges $z = 4$ blocks to construct a node according to Heuristic 1, the following scheme is obtained.

1. There are $\lfloor \frac{10201}{4} \rfloor = 2550$ many sensor nodes.
2. The probability that two nodes do not share a common key is approximately $(1 - \frac{32}{102})^{16} = 0.002421$. The experimental value on an average is 0.002094 with 100 runs which is lesser (better) than the theoretically estimated value.
3. Expected number of keys shared between two nodes $= \frac{16 \cdot 32}{102} \geq 5.019608$. The experimental value with 100 runs is 5.021088 on an average, little better than the theoretically estimated value.

In Table 4 we present the experimental value for $Fail(s)$, where we take the average over 100 runs for each s.

5 Key exchange

In this section, we present the key exchange protocol between any two nodes. First we present the key exchange protocol (as given in [11]) between two blocks N_a, N_b having identifiers (a_1, a_2) and (b_1, b_2), respectively. We take a $(v = kr, b = r^2, r, k)$ configuration. Thus, the identifier of

Table 4 Experimental *Fail*(s) values

s	1	2	3	4	5	6	7	8	9	10
Fail(s)	0.000724	0.001713	0.002750	0.004243	0.005912	0.008095	0.010394	0.013063	0.016221	0.019339

a block is a tuple (a_1, a_2) where $a_1, a_2 \in \{0, \dots, r-1\}$ and the identifier of a key is a tuple (k_1, k_2) where $k_1 \in \{0, \dots, k-1\}, k_2 \in \{0, \dots, r-1\}$.

Algorithm 1

```
1. Consider two blocks $N_a, N_b$ having identifiers $(a_1, a_2)$ and $(b_1, b_2)$, respectively.
2. if $a_1 = b_1$ (and hence $a_2 \neq b_2$), then $N_a$ and $N_b$ do not share a common key.
3. else $x = (b_2 - a_2)(a_1 - b_1)^{-1} \bmod r$. If $0 \leq x \leq k-1$, then $N_a$ and $N_b$ share the common key having identifier $(x, a_1 x + a_2)$. If $x \geq k$, then $N_a$ and $N_b$ do not share a common key.
```

They can independently decide whether they share a common key in $O(\log_2^2 r)$ time as inverse calculation is used [13, Chapter 5].

In the proposed system, a node comprises of z number of blocks. Since each block has an identifier (which is an ordered pair $(x, y) \in Z_r \times Z_r$), a node in the merged system has z number of such identifiers which is maintained in a list.

Algorithm 2

```
1. for the $t$th block in the node $N_a$, $t = 1, \dots, z$
    (a) send the identifier corresponding to the $t$th block to the other node $N_b$;
    (b) receive an identifier corresponding to a block in $N_b$;
    (c) compare the received identifier from $N_b$ with each of the $z$ identifiers in it (i.e. $N_a$) using Algorithm 1;
    (d) if a shared key is discovered acknowledge $N_b$ and terminate;
    (e) if an acknowledgment is received from $N_b$ that a shared key is discovered then terminate;
2. report that there is no shared key;
```

Since N_a and N_b participate in the protocol at the same time, the above algorithm is executed by N_a and N_b in parallel. There will be $O(z)$ amount of communications between N_a and N_b for identifier exchange and the decision whether they share a common key. At each node at most z^2 many inverse calculations are done (each identifier of the other node with each identifier of the node), which gives $O(z^2 \log_2^2 r)$ time complexity.

6 Conclusion and future research

In this paper, we first present a randomized block merging strategy in proposing a key pre-distribution scheme for secure communication among the sensor nodes. Our idea presents a departure from the usual combinatorial design in the sense that the designs are readily available according to user requirements. Our merging strategy results into schemes that are not directly available from combinatorial designs.

Our main target is to get more than one common keys among any pair of nodes that provides a robust network in terms of security under adversarial conditions where some nodes may get compromised. We present detailed mathematical analysis in presenting our results with supporting experimental data.

Next we present a heuristic improvement of the basic randomized block merging strategy. In this case we present a strategy for merging blocks in a (v, b, r, k) configuration in such a manner that the blocks constituting a node should not share any common key among themselves (may be there is a departure in a few cases where the blocks in a node may intersect, but we like to minimize it). This provides better parameters than our basic random merging strategy. The heuristic we have attempted is at a very basic level which is nothing but simple hill climbing kind of strategy and we got certain improvement with this only. Thus, it seems encouraging to attempt more advanced meta heuristic searches like simulated annealing or genetic algorithm for further improvement. We recommend this for future research.

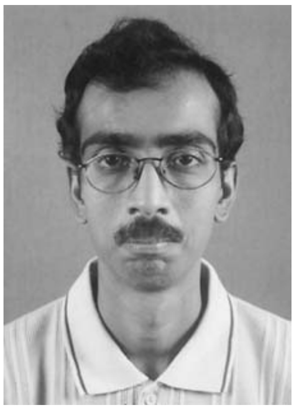
It will be interesting to regularize the key pre-distribution after random merging. In the strategy presented in this paper, the number of common keys between any two nodes follow binomial distribution. Thus, there is a probability (though very low) that there may be no common key between two nodes (for the time being, to get around this difficulty, two nodes can always communicate via an intermediate node with almost certainty). It looks promising to apply more sophisticated heuristic re-arrangement of blocks among the nodes available after the merging so that the number of common keys between any two nodes becomes more or less constant and always ≥ 1.

References

1. Blom, R.: An optimal class of symmetric key generation systems. Eurocrypt, LNCS 209 **84**, 335–338 (1985)
2. Camtepe, S.A., Yener, B.: Combinatorial design of key distribution mechanisms for wireless sensor networks. Eurosics (2004)
3. Chakrabarti, D., Maitra, S., Roy, B.: A hybrid design of key pre-distribution scheme for wireless sensor networks. In: The 1st International Conference on Information Systems Security, ICISS 2005, vol. 3803, Lecture Notes in Computer Science, pp. 228–238. Springer Verlag (2005)
4. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Research in Security and Privacy, pp. 197–213 (2003)

5. Colbourn, C.J., Dinitz, J.H.: The CRC Handbook of Combinatorial Designs. CRC Press, Boca Raton, Florida (1996)
6. Du, W., Ding, J., Han, Y.S., Varshney, P.K.: A pairwise key predistribution scheme for wireles sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 42–51. ACM CCS (2003)
7. Eschenauer, L., Gligor, V.B.: A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 41–47. ACM CCS (2002)
8. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing elliptic curve cryptography and RSA on 8-bit CPUs. CHES, LNCS 3156, pp. 119–132 (2004)
9. Kahn, J.M., Katz, R.H., Pister, K.S.J.: Next century challenges: mobile networking for smart dust. In: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 483–492 (1999)
10. Lee, J., Stinson, D.: Deterministic key predistribution schemes for distributed sensor networks. SAC, LNCS 3357, pp. 294–307 (2004)
11. Lee, J., Stinson, D.: A combinatorial approach to key predistribution for distributed sensor networks. In: IEEE Wireless Computing and Networking Conference (WCNC 2005). New Orleans, LA, USA (2005)
12. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security. ACM CCS (2003)
13. Stinson, D.: Cryptography: Theory and Practice, 2nd edn. Chapman & Hall, CRC Press, Boca Raton, Florida (2002)
14. Stinson, D.R.: Combinatorial Designs: Constructions and Analysis. Springer, New York (2003)
15. Street, A.P., Street, D.J.: Combinatorics of Experimental Design. Clarendon Press, Oxford (1987)

Dibyendu Chakrabarti received his Master of Technology in Computer Science in the year 1998 from the Indian Statistical Institute, Kolkata. Currently he is pursuing his Ph.D. from the Indian Statistical Institute, Kolkata. He is working in the area of Sensor Networks.



Subhamoy Maitra received his Bachelor of Electronics and Telecommunication Engineering degree in the year 1992 from Jadavpur University, Kolkata and Master of Technology in Computer Science in the year 1996 from the Indian Statistical Institute, Kolkata. He has completed Ph.D. from the Indian Statistical Institute in 2001. Currently he is an Associate Professor at the Indian Statistical Institute. His research interest is in Cryptology, Digital Watermarking, and Sensor Networks.

Prof. Bimal Roy obtained his Master's degree from the Indian Statistical Institute, Calcutta, India in 1979 and Ph.D. from University of Waterloo, Canada in 1982. He is currently a professor at the Indian Statistical Institute, Kolkata. His research area includes Cryptography, Security, Combinatorics etc. His special topics of interest are: Sensor Networks, Visual Cryptography, Hash Functions and Stream Ciphers.