CHAPTER 4

VLKM: VIRTUAL LOCATION-BASED KEY MANAGEMENT SCHEME FOR WSNs

This chapter presents a new virtual location based key management scheme (VLKM). This scheme uses virtual location to generate a round key for each sensor. Simulation results show that proposed scheme performs better than other existing schemes without increasing the communication overheads.

Rest of the chapter is organized as follows:

Key issues in key management schemes have been described in Section 4.1. Section 4.2 presents key management requirements in WSNs. Types of key management schemes have been described in Section 4.3. System model is explained in Section 4.4 followed by result and discussions in Section 4.5. Section 4.6 describes the presented of proposed key management scheme. Finally, chapter has been summarized in Section 4.7.

4.1. ISSUES IN KEY MANAGEMENT

Designing an energy efficient key management scheme to secure Wireless Sensor Networks is a challenging task because sensor nodes in the network are resource constrained. If an initial key is used in the network lifetime, a key stolen by an unauthorized node results in data compromised generated in the network. So a re-keying is necessary after a specified number of rounds for secure network. In clustering environment, a number of keys are needed for every sensor. If the role of sensor is cluster head, then one key is required to collect data from all cluster members. This key is shared between the cluster head and members of that cluster. The other key is required to transfer the aggregated data to base station which only shared between the cluster head and the base station. If the role of cluster head changes from sensor to sensor randomly, requirement of new keys arise to be shared between sensor node and new cluster head and between new cluster head and base station. In this scheme, re-keying after every round is a bottleneck of the network as more than one re-keying is required for every sensor.

4.2. KEY MANAGEMENT REQUIREMENTS

The basic goal of key management scheme in WSNs is to protect the information communicated over the network from attacks and misbehavior. The key management requirements in WSNs include:

- *Memory Storage (MS):* A sensor node is usually resource constrained in terms of memory. Therefore, it is important that the amount of memory needed by a key management scheme is minimum. In general, the simpler the protocol is, the more memory space should be made available for storing the security credentials.
- *Communication Overhead (CO):* In most key management scheme (KMS), the nodes must exchange control messages with their group members through communication channels in order to establish a group key. Some protocols may require the exchange of a small amount of control messages to share a group key while other protocols may need to undergo a complex negotiation among them.
- *Resource overhead (RO):* Sensor nodes are resource constrained in terms of processing and storage. So it is important to determine the amount of resources (such as time and storage) necessary to execute the encryption algorithm to implement the security in the existing protocol. Fortunately, there are many KMS that are not very computationally efficient.
- *Key Security (KS):* It should be necessary that the key distribution or key updation process must be secure by itself. So the protocols do not need to exchange sensitive information (e.g. Key, node ID, etc) for this purpose.
- *Forward secrecy:* This ensures that a compromised current secrets or keys should not be able to compromise any secret or key used in future.
- *Backward secrecy:* This ensures that a compromised current secrets or key should not be able to compromise any earlier secret or key.

4.3. TYPES OF KEY MANAGEMENT SCHEMES

There are two fundamental key management schemes that are used in WSNs, i.e. static and dynamic. In static key management scheme, the sensors have a fixed number of keys loaded prior to network deployment but in dynamic key management schemes, the key is redistributed periodically or on demand as needed in the network. One significant

disadvantage of this scheme is increased communication overhead due to keys redistribution to each and every node in the network.

Presented scheme calculates dynamic key by running a key generation function periodically or on demand as needed by the network. This function generates a dynamic key for each sensor with the help of its virtual location. There are many reasons for key refreshment including updating keys after a key revocation has occurred, refreshing key so that it does not become stale or changing keys due to dynamic changes in the topology.

Since a sensor node is either forwarding the aggregated data after collecting it from all cluster members if it is a cluster head or injecting its own data to the cluster head if it is not a cluster head. In case if it is a cluster head, it requires two different keys, one that is shared between this sensor and the base station only and second key that is shared between this sensor and all the cluster members of the same cluster. In other case when it is not a cluster head, it needs a single key to be shared between this sensor and the cluster head.

4.4. PRESENTED MODEL FOR KEY MANAGEMENT

This section presents a Virtual Location based dynamic Key Management scheme (VLKM). Overview of the presented scheme has been discussed first followed by the detailed discussion of the protocol.

4.4.1. Network Clustering

Presented scheme uses the concept of physical clustering, i.e. every cluster is identified by its physical boundary. The boundary of each cluster is similar in size. Number of clusters into horizontal and vertical directions is decided in the setup phase. All the clusters are equal in size. Each cluster is indentified by its cluster ID as shown in Figure 4.1.

There are two type of clustering scheme that is used in wireless sensor networks, i.e. static and dynamic. In static clustering scheme, clusters are fixed and there is no update into size and member of the cluster after formation. But in dynamic clustering scheme, the size and members of every cluster is changed in every phase of cluster formation. The members of every cluster increases or decreases in every cluster formation, thus the size of cluster grows or shrinks accordingly. Presented model uses the concept of static clustering. Once the cluster is decided for any sensor, it is fixed and permanent for every round throughout the

network lifetime. In this model, the cluster for any sensor is decided by its physical location in the network as shown in Figure 4.2.

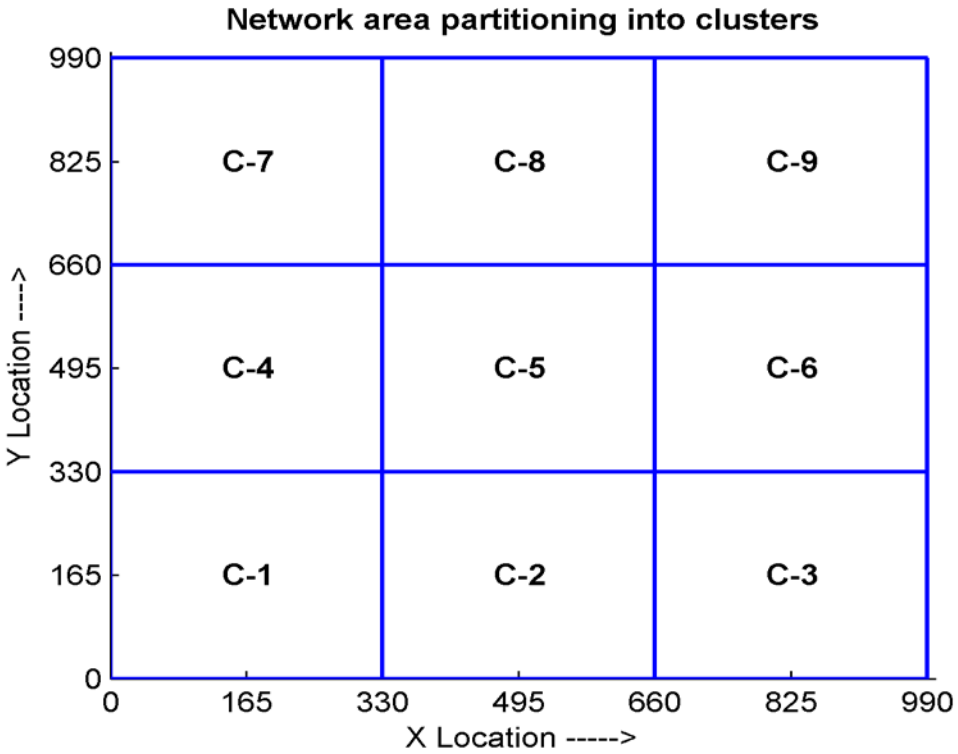

Figure 4.1: Static Cluster Formation

Figure 4.2: Clustering

4.4.2. Random Locations

Every sensor is assigned a virtual location to identify its location in two dimensional plane with its X and Y coordinates value. This virtual location is used to generate a key to be used in the network. This location is given to every sensor within its cluster boundary. Similarly every cluster has also provided a random cluster virtual location. This virtual location is also a random location within cluster boundary and is saved in the memory of all the cluster members of the same cluster. Virtual location of all the clusters is shown in Figure 4.3 along with a link showing random actual and random virtual locations for all the sensors.

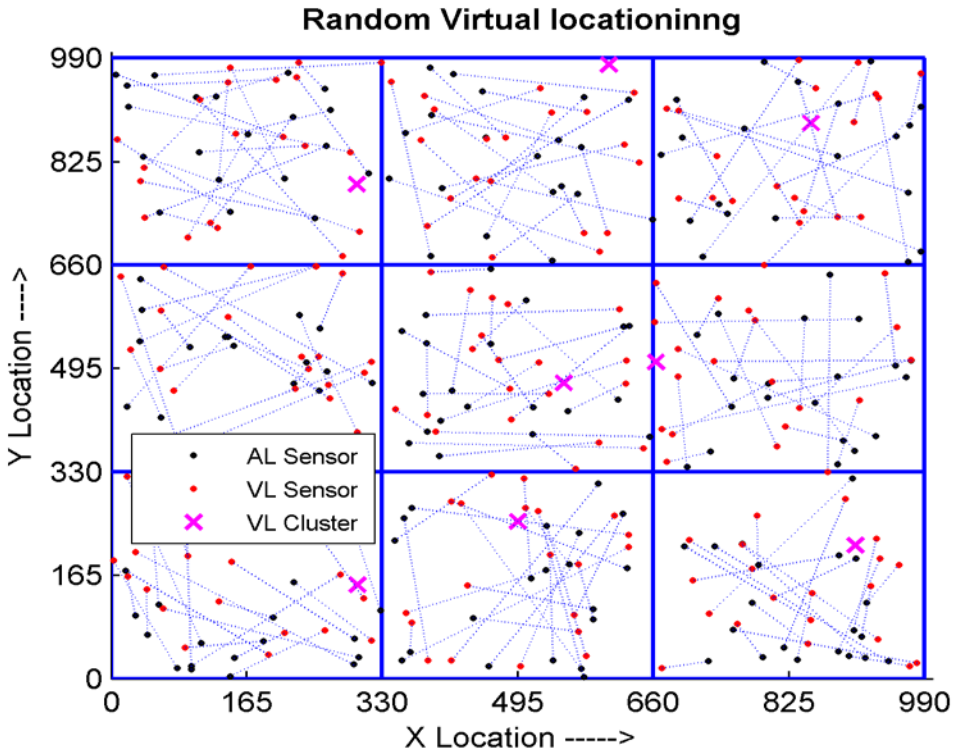

Figure 4.3: Link showing Actual and Virtual location

4.4.3. Virtual Origins

A virtual origin is used for the virtual locations, i.e. a random virtual origin is provided to the network and all clusters map their virtual locations on this virtual origin of the network. Similarly virtual origin is also provided to every cluster and this origin is any location within the cluster boundary as shown in Figure 4.4.

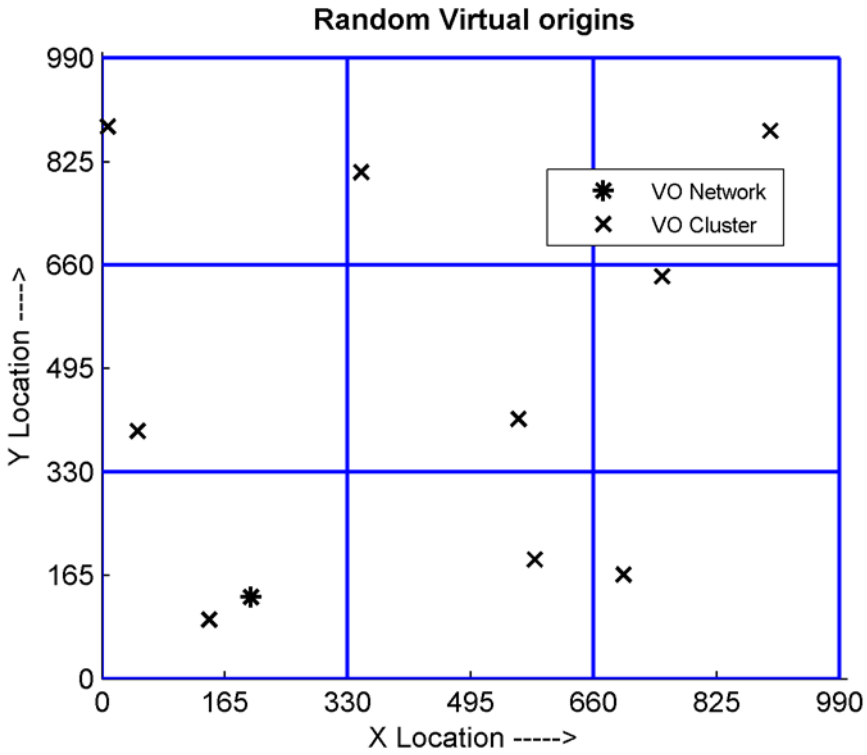

Figure 4.4: Virtual Origin of Network and Cluster

This model is used in the networks where more than one key is required for any type of sensor. One key is shared between sensor node and base station and the other key is shared between all the group members. The types of keys used in wireless sensor networks are given below:

4.4.4. Cluster key

The key of j^{th} cluster, also known as Cluster key (KC_j) is shared between all the sensors of j^{th} cluster. Where the value of j exists between one and number of clusters (NOC) in the network, i.e. $1 \leq j \leq NOC$. This cluster key is shared between more than two sensors and is known to all the members of j^{th} cluster and also to the base station. If a normal sensor node 'A' which is not a cluster head, wants to transfer the sensed data to the cluster head sensor 'B' then 'A' uses its cluster key to transfer the data to 'B'.

a) Every cluster has an initial virtual location, virtual boundary, virtual speed of movement, virtual angle of movement and virtual direction of movement only. These parameters are known to all the members of that cluster.

b) The network is provided a virtual origin and all the clusters map their virtual locations on this virtual origin as shown in Figure 4.5. The coordinates of this virtual origin can't be (0, 0). For example if virtual location of any cluster is (5, 2) and the coordinates of virtual origin is (3, 2) then the virtual location of this cluster after mapping is (8, 4).

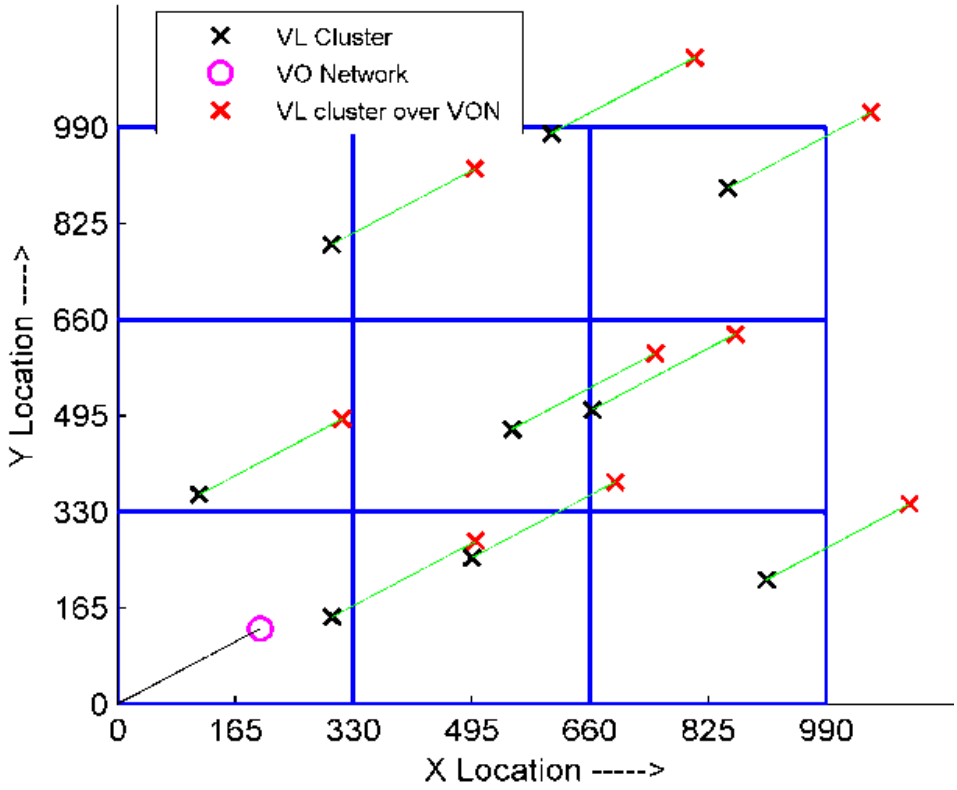

Figure 4.5: Cluster Virtual Location Mapping

c) All the sensor of same cluster moves virtually in same direction, with same speed and angle of movement.

d) Current location of any cluster is updated, every time when the members of that cluster are moved. This current location is used to generate a key.

e) All the sensors calculate current virtual location for current round. This loction contains 'X' and 'Y' coordinates (decimal value upto two digit decimal place). This current location is same for all the members of same cluster.

f) All the sensors apply one way hash function on current virtual location to produce the cluster key used in current round.

g) Whenever all the sensors of a cluster has been elected as a cluster head, all sensors updates their virtual location thus current virtual location is updated which results in updation of cluster key.

h) To change the cluster key for any clusters, only virtual location of that cluster is updated.

i) Virtual origin of the network is updated when all the sensors of entire network has already been elected as a cluster head.

4.4.5. Sensor key

The Cluster key (K_i) is shared only between the sensor node 'i' and base station. This key is different for all the sensors in the network. Where the value of i is between one and number of sensors (N) in the network, i.e. $1 \leq i \leq N$. If some sensor node 'A' is elected as a cluster head then 'A' uses its sensor key to transfer the information to the base station.

a) Every sensor has an initial virtual location, boundary, speed, angle and direction of movement, which are known to the sensor and the base station only. This location and other parameters are different and independent than the parameters of a cluster.

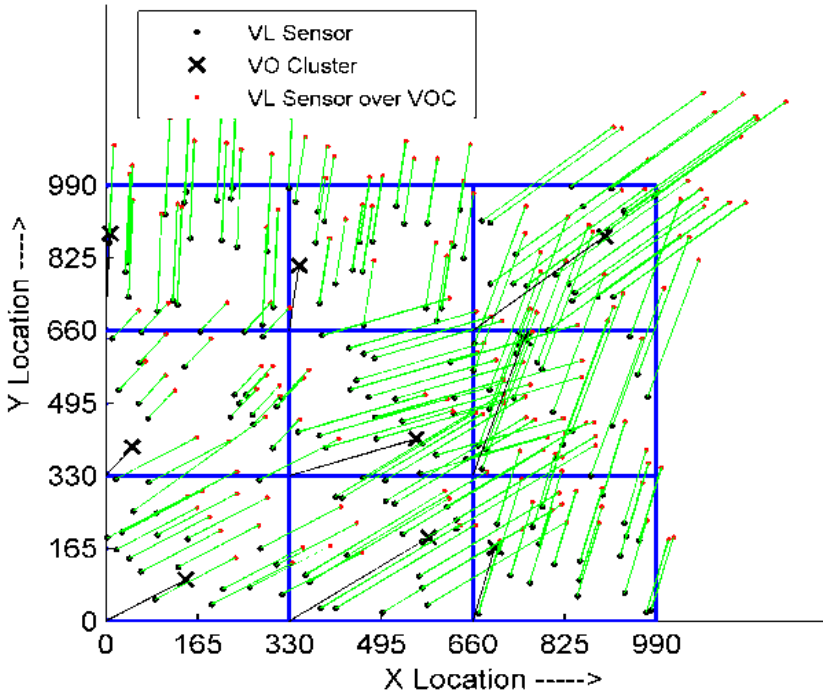

Figure 4.6: Sensor Virtual Location Mapping

b) All the sensors map their virtual locations on virtual origin of the cluster as shown in Figure 4.6.

c) Every sensor moves virtually in its virtual boundary with a specified direction, speed and angle of movement.

d) Sensors computes its current virtual location for current round. This loction contains ‘X’ and ‘Y’ coordinates (decimal value upto two decimal places). This current location is different and independent for all the members in a network.

e) Sensors apply one way hash function on its initial virtual location and current virtual location to produce the sensor key used in current round.

f) Whenever all the sensors of the cluster has already been elected as a cluster head, all sensors updates their virtual location thus current virtual location is updated which results in updation of sensor key.

g) To change the sensor key for a particular sensor, only virtual location of that sensor is updated.

h) Virtual origin of the network is updated when all the sensors of entire network has already been elected as a cluster head.

4.4.6. Virtual Movements

Every sensor has a unique virtual boundary. This virtual boundary is a surrounding area around the sensor where the senor is free to move with a specific angle in a particular direction with a constant speed. When sensor hits to its virtual boundary by moving virtually, its direction of movement is changed accordingly, i.e. from Bottom Right (BR) to Right Top (RT) similarly from RT to Top Left (TL) and from TL to Left Bottom (LB) and from LB to BR. Whenever any sensor moves, its current virtual location is changed accordingly as shown in Figure 4.7.

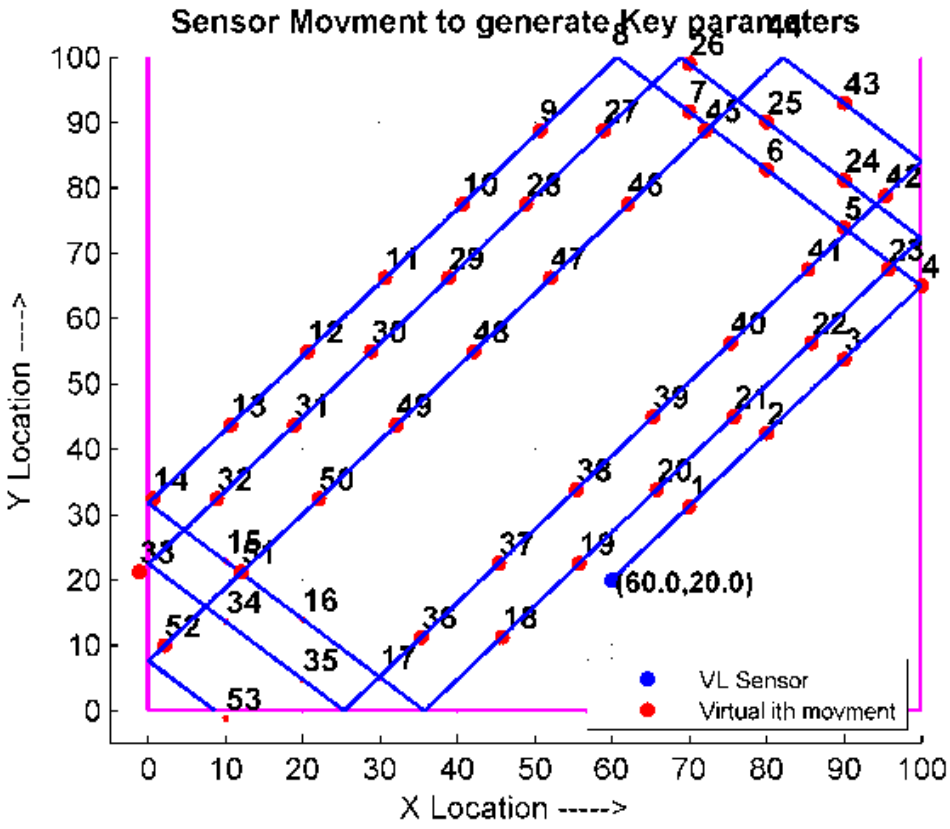

Figure 4.7: Sensor movement within virtual boundary

All the Sensors initially map its virtual locations to the virtual origin, i.e. to generate the sensor key it map its virtual location over virtual origin of the cluster but if it generates a cluster key then it map its virtual cluster location over virtual origin of the network. With this movement speed, angle and virtual boundary, the sensor can generate any number of keys. There is no need to update the keys manually. The key is dynamic and is updated every time when the sensor changes its virtual location by its virtual movement. When all the sensors of a cluster are elected as a cluster head, there is a need to update the cluster key. So all the sensor of that cluster updates there virtual location by moving themselves to a new location with a given virtual angle, speed and boundary to generate a new cluster key for next phase, and continues with this new cluster key until all the sensors in a cluster are not elected as a cluster head once again. Similarly when there is a need to update the sensor key which is shared between a particular sensor and the base station, the sensor updates its virtual location by moving itself to a new location with a given virtual angle, speed and boundary to generate a new sensor key for next phase. The sensor generates dynamic key by applying a one way

hash function on its initial virtual location and current virtual locations. To generate a next dynamic key, it changes its current virtual location by moving itself to a new virtual location again with same angle and direction with a constant speed.

Its initial virtual location, angle of movement speed of movement and virtual boundary are already known to base station so base station easily calculates dynamic key of all the sensors of network. Normal sensors transmit the sensed data to the cluster head sensor with cluster key. Similarly, cluster head receives data from normal sensors by using a cluster key. Cluster head transmits the aggregated data to the base station by using its sensor key shared between cluster head and base station only. As base station receives data from cluster heads only; it calculates keys only for cluster head sensors. To balance the power consumption between all the nodes in the cluster it is necessary to rotate the role of cluster head and then base station needs a different key to be shared between base station and new cluster head.

Table 4.1: Compute Virtual Location Notations

Notation	Description
IVL	Initial Virtual Location (IVX, IVY) mapped onto network virtual origin
Cr	Current round
VL_{Cr-1}	Virtual Location (VX_{Cr-1}, VY_{Cr-1}) in previous round
VL_{Cr}	Virtual location (VX, VY) for current round
VB	Virtual Boundary
VA	Virtual Angle of movement in degree
VS	Virtual Speed (distance covered by a sensor in moving a single round)
VD	Current virtual Direction of movement, i.e. Left to Right (LR), Right to Top (RT), Top to Left (TL) or from Left to Bottom (LB).

If all the sensors in a network has same size virtual boundary, the virtual locations of two different sensors is different irrespective of their boundary because their boundary surrounds to their virtual location which is different for each sensor and only known to that sensor. Otherwise there are several other methods to produce this difference, i.e. with different size of virtual boundary, with different angle of virtual movement or with different speed of

virtual movement etc. *Compute Virtual Location (CVL)* is an algorithm that specifies how the sensor is moving into its virtual boundary. The abbreviations used in this algorithm are given in Table 4.1.

Algorithm 4.1: Compute Current Virtual Location (VX, VY) for Current round (Cr)

Input: Virtual Location in previous round VL_{Cr-1}, Virtual Boundary (VB), Virtual angle of movement in degree (VA), Virtual speed of movement (VS) and current virtual direction of movement (VD);
Output: Virtual location (VL_{cr});
Procedure: CVDL (VL_{Cr-1}; VB;, VA; VS; VD)

1. **Begin**
2. **Set previous location as a start location**
   a. X1=VX_{Cr-1};
   b. y1=VY_{Cr-1};
      i. opposite= VS*sin(ang);
      ii. adjacent= VS*cos(ang);
3. **Switch(VD)**
   a. case(BR)then
      i. if((x1+adjacent)>Vmax_x) then
         a) VX=Vmax_x
         b) Direction=RT
      ii. elseif((y1+ opposite)>Vmax_y) then
         a) VY=Vmax_y
         b) Direction=TR
      iii. Else
         a) VX=x1+adjacent
         b) VY=y1+opposite

   Endcase (3.a)
   b. case(RT)then
      i. if((x1-adjacent)<Vmin_x) then
         a) VX=Vmin_x
         b) Direction=RL
      ii. elseif((y1+ opposite)>Vmax_y) then
         a) VY=Vmax_y;
         b) Direction=TL
      iii. Else
         a) VX=x1-adjacent
         b) VY=y1+opposite

   Endcase (3.b)
   c. case(TL)then
      i. if((x1-adjacent)<Vmin_x) then
         a) VX=Vmin_x
         b) Direction=LB
      ii. elseif((y1- opposite) < Vmin_y) then
         a) VY=Vmin_y;

b) Direction=BR
iii. Else
a) VX=x1 - adjacent
b) VY=y1 - opposite
Endcase (3.c)
d. case(LR)then
i. if((x1 + adjacent) > Vmax_x) then
a) VX=Vmax_x
b) Direction=BR
ii. elseif((y1- opposite) < Vmin_y) then
a) VY=Vmin_y;
b) Direction=RT
iii. Else
a) VX=x1+ adjacent
b) VY=y1 - opposite
Endcase (3.d)
4. End

4.4.7. Key Generation

The virtual dynamic Keying module generates dynamic key by applying one way hash function on initial and current virtual locations.

The location function produces a current virtual location which is calculated when this function runs or when the sensor changes its virtual location by moving virtually within virtual boundary to calculate the key to be used in the encryption process. This module takes old Virtual Location (VL) used in the previous round, Virtual Direction of movement (VD) used or updated in the previous round, Virtual Boundary (VB) which is virtual surrounding rectangular area around the sensor and virtual angle of movement. This module calculates current virtual location (CVL). This location is used by the keying module to generate the dynamic key used in the encryption process.

4.4.7.1. Cluster key Generation

The cluster key (KC_j), key of j^{th} cluster is calculated by applying a one way hash function on cluster initial virtual location $(CIVL_j)$ and current virtual location (CVL_j).

So Cluster key $(KC_j) = \int(CIVL_j, CVL_j)$

Where CIVL is the j^{th} cluster's initial virtual X and Y locations and CVL_j is j^{th} clusters current virtual X and Y locations and $\int$ is a one way hash function. Both these X and Y location are decimal numbers up to two digits. Figure 4.8 shows different virtual locations of different clusters with same boundary size, same angle of movement and same direction of movement.

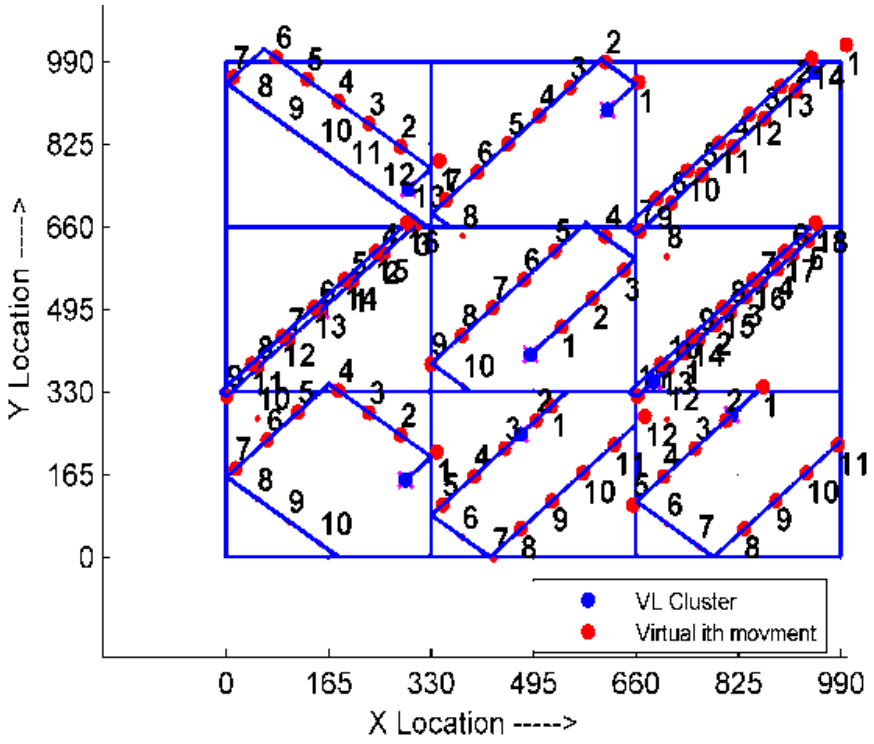

Figure 4.8: Sensor movement within cluster virtual boundary

4.4.7.2. Sensor key Generation

Sensor key (K_i), key of sensor node 'i' is calculated by applying a one way hash function on sensor's initial virtual location (IVL_i) and current virtual location (VL_i).

Sensor key $(KS_i) = \int(IVL_i, VL_i)$

Where IVL is the i^{th} sensor's initial virtual X and Y locations and VL is the i^{th} sensors current virtual X and Y locations and $\int$ is a one way hash function. Both X and Y locations are represented by a decimal number up to two digit. Figure 4.9 shows different virtual locations of different sensors in same cluster with same boundary size, same angle of movement and same direction of movement.

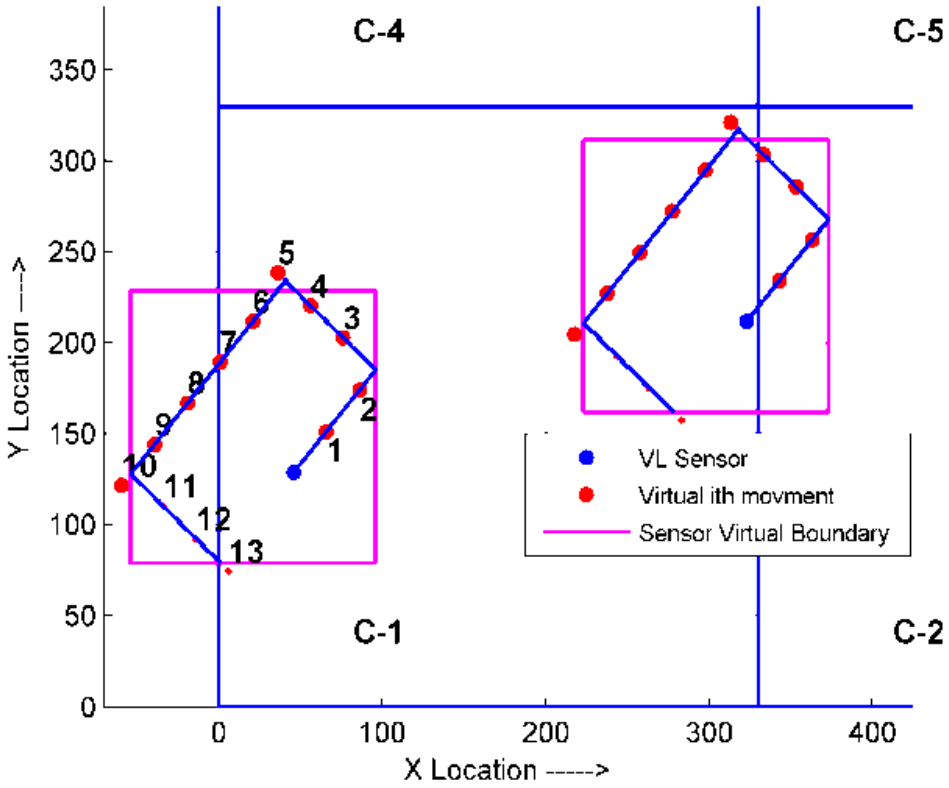

Figure 4.9: Two different Sensors moving in same clusters

Whenever any sensor is elected as a cluster head, all the sensor sent data to cluster head by encrypting the data with the help of a cluster key and when cluster head receives the data from any sensor then it decrypt the data with the same key and calculate aggregate value by mixing the data of all the sensors and sent this aggregated data to base station after encrypting this data with the help of a sensor key shared between the sensor node and the base station. Compute Dynamic Key (CDK) is an algorithm that specifies how the sensor generates a key used for current round with the help of initial and current virtual locations. The abbreviations used in this algorithm are shown in Table 4.2.

Table 4.2: Compute Dynamic Key Notations

Notations	Description
IVL	Initial Virtual Location, i.e. IVX, IVY
CVL	Current Virtual Location, i.e. CVX, CVY
FIX	Folding Addition of Integer parts (three digits) in ‘X’ coordinates of

	initial and current virtual locations.
FIY	Folding Addition of Integer parts (three digits) in 'Y' coordinates of initial and current virtual locations.
FIZ=FIX+FIY	Folding Addition of Folded Integers parts in initial and current virtual locations.
FDX	Folding Addition of Decimal parts (two digits) for 'X' locations in initial and i^{th} Virtual movement
FDY	Folding Addition of Decimal parts (two digits) for 'Y' locations in initial and i^{th} Virtual movement
FDZ=FDX+FDY	Folding Addition of Folded decimal parts in initial and current virtual location.
KEY	Key of sensor node in current round (cr)

4.4.8. Virtual Location-Based Keying Module

The Virtual Location-Based Keying Module (VLKM) is one of the prime contributions of this chapter. This is a method to produce a dynamic key to be used by the crypto module to encrypt the packet sense by a node.

After deployment, all the sensor nodes traverse virtually thus change its virtual location. The current value of the virtual location (CVX, CVY), along with the initial virtual location after origin mapping (IVX, IVY) is used as the input to the key generation function, F. The keying module uses the concept of Folding Addition (FA) instead of using the simple addition. Folding addition is a one way function which is non-invertible, i.e. input produces an output but the output never produces an input. The method of folding addition is given in algorithm 4.2.

In folding addition method, if two integer numbers of three bit wide are added and the result is more than three digits then LSB is added after extraction to MSB in the result. The process is repeated till the result is not a number that has three or less number of digits as given in example:

A=995, B=994, $C = A \hat{+} B$ where '$\hat{+}$' is a folding addition, i.e. C=995+994=1989

Algorithm 4.2 : Folding Addition

While (C>999)// Folding Addition

a. LSB=mod(C,10)
ii. C=floor(C/10)
iii.C=C+(LSB*100)

End while

C=909

So it is completely impossible to generate two input values A=995 and B=994 from output value C=909.

The process of key generation is initiated when sensor changes its current virtual location after sensing data and thus, no separate mechanism is needed to update or refresh the key. Moreover, the dynamic nature of the keys makes it difficult for an intruder to intercept any packets to break the security parameters. As described earlier, in the presented scheme all X and Y locations are in three digit numbers up to two digit decimal place. The hierarchy can be increased up to any label. This algorithm first separates integer parts from initial (IVX, IVY) and current virtual locations (CVX, CVY) and then separate decimal parts from both initial (IVX, IVY) and current virtual locations (CVX, CVY). After that it adds both integer parts of X locations by folding method. Similarly it adds both integer parts of Y locations using folding method. Then it adds both the results using folding method. Similar procedure is followed for decimal parts also, and addition is performed using folding method. And finally, both the results of integer and decimal parts are concatenates together to prepare dynamic key. The details are given in Algorithm 4.3.

Algorithm 4.3: Compute Dynamic Key for current round by initial virtual location (IVX, IVY) and current virtual movement (CVX, CVY))

Input: Initial virtual location (IVX, IVY); Current virtual location (CVX, CVY));
Output: KEY_{cr};
Procedure: ComputeDynamicKey (IVX, IVY, CVX, CVY)

1. **Begin**
2. **Seprate integer parts from initial (IVX, IVY) and current virtual locations (CVX, CVY)**

a. IX1=floor(IVX)
b. CX1=floor(CVX)
c. IY1=floor(IVY)
d. CY1=floor(CVY)

3. **Seprate Decimal parts from initial (IVX, IVY) and current virtual locations (CVX, CVY)**
   a. IX2=10^2*(IVX-IX1)
   b. CX2=10^2*(CVX-CX1)
   c. IY2=10^2*(IVY-IY1)
   d. CY2=10^2*(CVY-CY1)
4. **Add both integer parts of X Locations in folding method**
   a. I1=IX1+CX1
   b. **While (I1>999)**
      i. LSB=mod(I1,10)
      ii. I1=floor(I1/10)
      iii. I1=I1+LSB*100
      End While 4.b
5. **Add both integer parts of Y Locations in folding method**
   a. I2=IY1+CY1
   b. **While (I2>999)**
      i. LSB=mod(I2,10)
      ii. I2=floor(I2/10)
      iii. I2=I2+LSB*100
      End While 5.b
6. **Add both integer part of folded X and Y locations using folding method**
   a. I3=I1+I2
   b. **While (I3>999)**
      i. LSB=mod(I3,10)
      ii. I3=floor(I3/10)
      iii. I3=I3+LSB*100
      End While 6.b
7. **Add both Decimal parts of X Locations using folding method**
   a. D1=IX2+CX2
   b. **While (D1>99)**
      i. LSB=mod(D3,10)
      ii. D1=floor(D1/10)
      iii. D1=D1+LSB*10
      End While 7.b
8. **Add both Decimal parts of Y Locations using folding method**
   a. D2=IY2+CY2
   b. While (D2>99)
      i. LSB=mod(D2,10)
      ii. D2=floor(D2/10)
      iii. D2=D2+LSB*10
      End While 8.b
9. **Add both Decimal part of folded X and Y locations using folding method**

a. D3=D1+D2
b. **While (D3>99)**
    i. LSB=mod(D3,10)
    ii. D3=floor(D3/10)
    iii. D3=D3+LSB*10
    End While 9.b
10. **KEY=str2num(strcat(num2str(D3),num2str(I3)))**
11. **Return KEY**
12. **End**

Example 4.1 to generate key from virtual locations

The sensor is moving into its virtual boundary. Virtual boundary of sensor is represented with bottom left X (minx), bottom left Y (miny), Top right X (maxx) and Top Right Y (maxy). The initial virtual location of sensor is (VX, VY). And this location is mapped over virtual origin (VOX=15, VOY=5). The initial direction of movement is BR. The virtual speed of movement (VS) is 20 (meter). And finally the virtual angle (VA) of movement is 40 degree as shown in Figure 4.10.

Network parameters to produce different keys for different virtual movement are given below:

Virtual Boundary: minx=900, maxx=1000, miny=900, maxy=1000

Virtual Origin: NVOX=15, NVOY=5

Initial Virtual location: VX= 927.00, VY=927.00

Virtual location over virtual origin (IVL): VXN=942.00, VYN=932.00

Virtual Direction: VD='BR'

Virtual Angle of movement: VA= 40°

Virtual speed of movement: VS=20 meter

Current Virtual location (CVL): CVX= 957.32, CVY= 944.86

FIX=588, FIY=287, FIZ=875, FDX=32, FDY=86, FDZ=91

KEY=91875

KEY in binary format: 10110011011100011

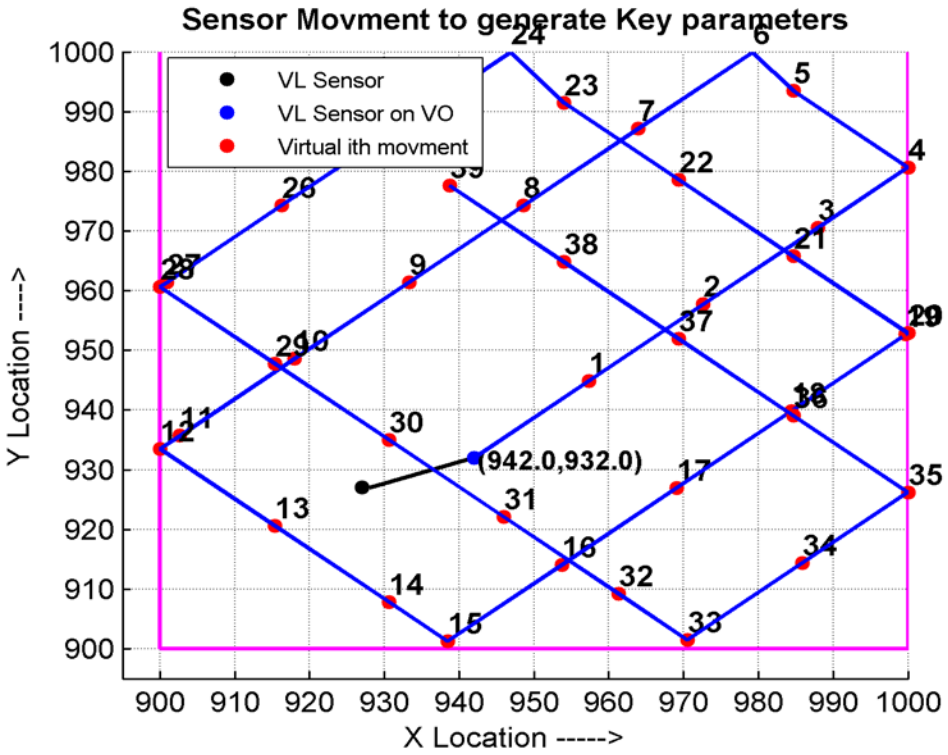

Figure 4.10: Virtual Movement after Origin Mapping

Table 4.3 shows different keys generated from different virtual locations. Based upon these keys simulation results are calculated and explained in Section 4.5.

Table 4.3: Key generation for Different Virtual Movements

Sr. No.	CVL	FIX	FIY	FIZ	FDX	FDY	FDZ	KEY	KEY in binary format
1	957.32,944.86	588	287	875	32	86	91	91875	10110011011100011
2	972.64,957.71	900	588	948	64	71	63	63948	01111100111001100
3	987.96,970.57	591	889	148	96	57	45	45148	01011000001011100
4	1000.0,980.67	892	890	378	0	67	67	67378	10000011100110010
5	984.68,993.52	291	192	483	68	52	12	12483	00011000011000011
6	979.24,1000	790	892	368	24	0	24	24368	00101111100110000
7	963.92,987.14	189	591	780	92	14	70	70780	10001010001111100
8	948.6,974.29	687	290	977	60	29	89	89977	10101111101111001
9	933.28,961.43	186	988	517	28	43	71	71517	10001011101011101
10	917.96,948.58	584	687	227	96	58	55	55227	01101011110111011
11	902.64,935.72	308	386	694	64	72	73	73694	10001111111011110
12	900.00,933.51	882	186	906	0	51	51	51906	01100101011000010
13	915.32,920.65	384	884	926	32	65	97	97926	10111111010000110
14	930.64,907.79	885	583	946	64	79	44	44946	01010111110010010
15	938.44,901.25	686	982	966	44	25	69	69966	10001000101001110
16	953.76,914.11	188	284	472	76	11	87	87472	10101010110110000

17	969.08,926.97	789	485	527	8	97	60	60527	01110110001101111
18	984.4,939.82	291	786	807	40	82	32	32807	01000000000100111
19	999.72,952.68	792	808	160	72	68	14	14160	00011011101010000
20	1000.0,952.91	892	808	170	0	91	91	91170	10110010000100010
21	984.68,965.77	291	389	680	68	77	64	64680	01111110010101000
22	969.36,978.62	789	690	804	36	62	98	98804	11000000111110100
23	954.04,991.48	288	991	802	4	48	52	52802	01100111001000010
24	946.89,1000	487	892	803	89	0	89	89803	10101111011001011
25	931.57,987.14	985	591	757	57	14	71	71757	10001100001001101
26	916.25,974.29	484	290	774	25	29	54	54774	01101010111110110
27	900.93,961.43	882	988	187	93	43	73	73187	10001110111100011
28	900.00,960.66	882	888	177	0	66	66	66177	10000001010000001
29	915.32,947.8	384	587	971	32	80	31	31971	00111110011100011
30	930.64,934.94	885	286	217	64	94	95	95217	10111001111110001
31	945.96,922.09	387	508	895	96	9	60	60895	01110110111011111
32	961.28,909.23	988	783	277	28	23	51	51277	01100100001001101
33	970.52,901.49	889	982	287	52	49	20	20287	00100111100111111
34	985.84,914.34	391	284	675	84	34	91	91675	10110011000011011
35	1000.0,926.22	892	485	837	0	22	22	22837	00101100100110101
36	984.68,939.08	291	786	807	68	8	76	76807	10010110000000111
37	969.36,951.94	789	987	777	36	94	13	13777	00011010111010001
38	954.04,964.79	288	289	577	4	79	83	83577	10100011001111001
39	938.72,977.65	686	590	727	72	65	83	83727	10100011100001111

4.5. RESULTS

4.5.1. Key Duplication

In every round a new dynamic key is provided to each and every sensor in the network. Key duplication is the process when a key is similar between two or more sensors in same round. In previous schemes, dynamic key is provided to each and every sensor in every round from a fixed key pool. To check the behavior of network, the presented scheme is compared with the traditional scheme. In simulation, 54 sensors have been considered in the network. The network is assumed to run for 50 rounds. To provide a new key in each round, the size of fix key pool is enhanced from a key pool of single key to a key pool with length of 100 keys. A random key is chosen from the key pool and compared with the presented scheme. Simulation results in Figure 4.11 shows that key duplication is 100% in case of key pool with length 1, i.e. in every round the key of every sensor is matched with all other sensors in the network. When the length of key pool is increased from 1 to 100, key duplication is

decreased from 100% to 20%. In presented key management scheme, the key duplication is between 0 to 1%.

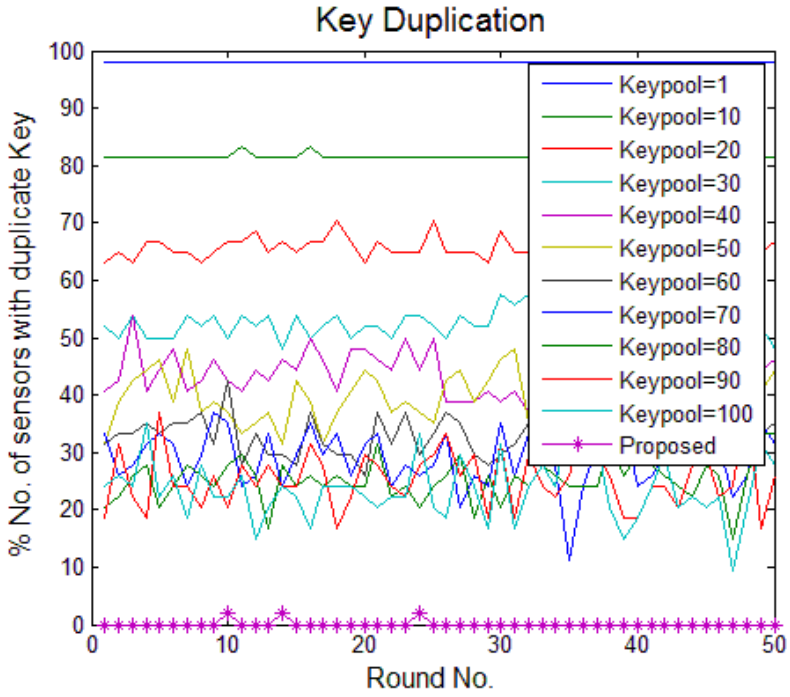

Figure 4.11: Key Duplication

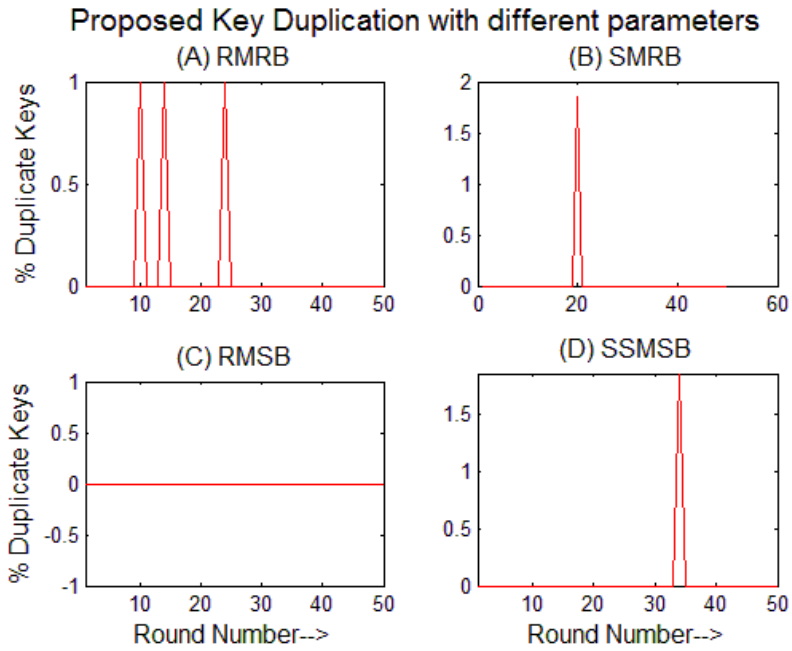

Figure 4.12: Key Duplication for Different Key Parameters

Figure 4.12 shows key duplication with different values for different parameters used in key computation for individual sensor. Figure 4.12-A shows key duplication when Random Movement and Random Boundary (RMRB) are provided to every sensor in the network. Figure 4.12-B shows key duplication when Same Movement but Random Boundary (SMRB) is provided to every sensor. Figure 4.12-C shows key duplication when Random Movement but Same Boundary (RMSB) is provided to every sensor. Figure 4.12-D shows the behavior of network when Same Movement and Same Boundary (SMSB) are provided to every sensor in the network. In all the cases, the performance of the presented system is still better.

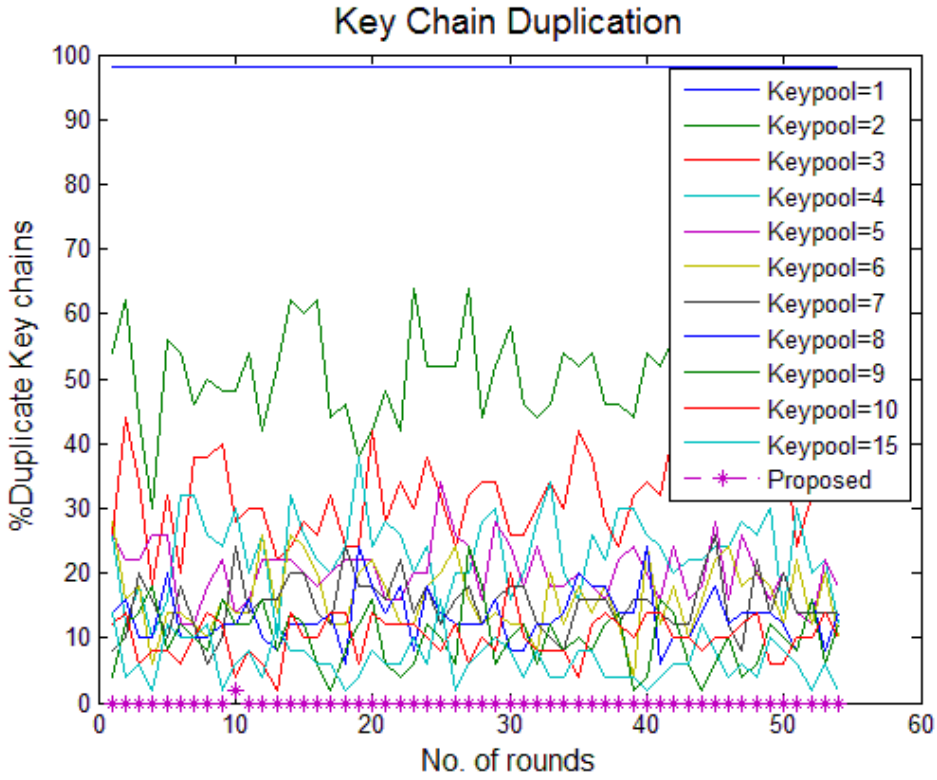

Figure 4.13: Number of Two or More Keys Match in Series Between Any Two Consecutive Rounds

4.5.2. Key Chain Duplication

Key chain duplication is the process when a sensor has the common key in consecutive two or more rounds. Simulation results in Figure 4.13 shows that key chain duplication is 100% in case of key pool with length 1, i.e. in every two consecutive rounds the same key is provided to every sensor in the network. When the length of key pool is increased from 1 to

15, key chain duplication is decreased from 100% to 4%. But in the presented key management scheme, key chain duplication is 0.1%.

Figure 4.14 shows key chain duplication with different values for different parameters used in key computation for individual sensor. Figure 4.16-A shows key chain duplication when Random Movement and Random Boundary (RMRB) are provided to every sensor in the network. Figure 4.16-B shows key chain duplication when Same Movement but Random Boundary (SMRB) is provided to every sensor. Figure 4.16-C shows key chain duplication when Random Movement but Same Boundary (RMSB) is provided to every sensor. Figure 4.16-D shows the behavior of network when Same Movement and Same Boundary (SMSB) are provided to every sensor in the network. In all the cases, the performance of the presented system is still better.

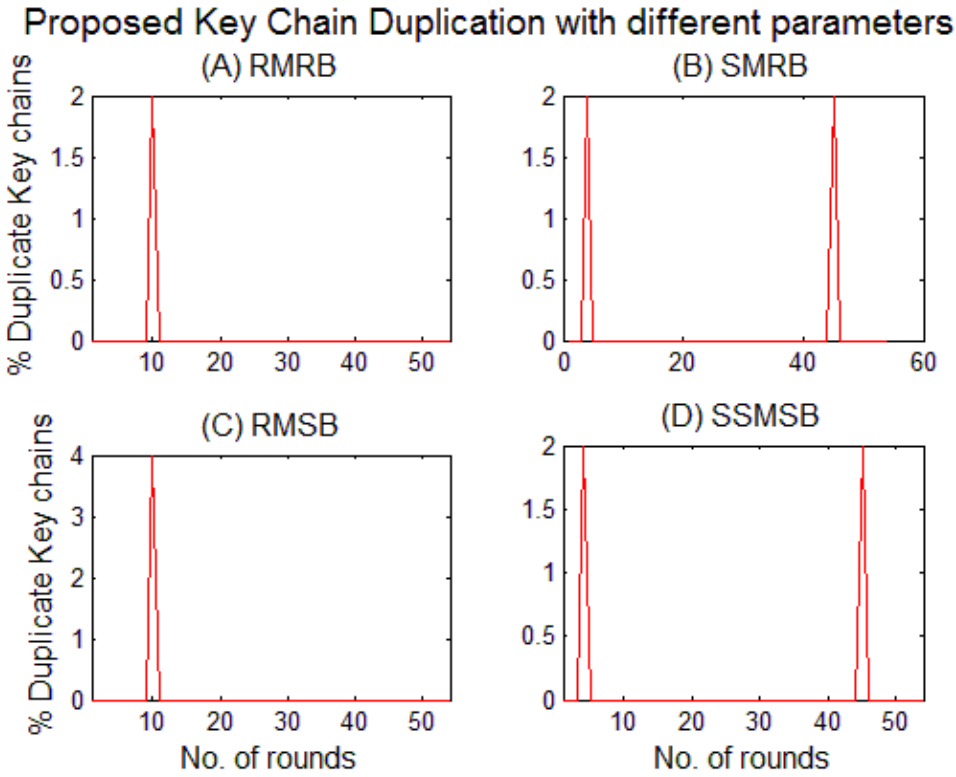

Figure 4.14: Key Chain Duplication for Different Key Parameters

4.5.3. Key Uniqueness

Total number of unique keys that are used by any sensor for all rounds is known as key uniqueness. Simulation results in Figure 4.15 shows that key uniqueness is 1% in case of key

pool with length 1, i.e. in every round the same key is provided to every sensor in the network. When the length of key pool is increased from 1 to 100, key uniqueness is increased from 1% to 85%. In presented key management scheme, key uniqueness is 100%.

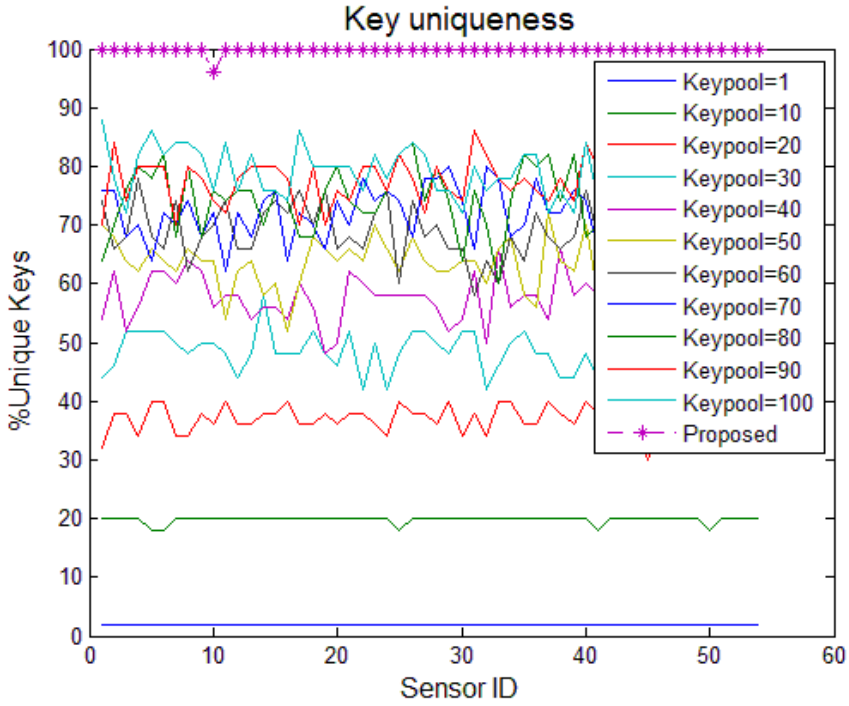

Figure 4.15: Keys Uniqueness

Figure 4.16 shows key uniqueness with different values for different parameters used in key computation for individual sensor. Figure 4.16-A shows key uniqueness when Random Movement and Random Boundary (RMRB) are provided to every sensor in the network. Figure 4.16-B shows key uniqueness when Same Movement but Random Boundary (SMRB) is provided to every sensor. Figure 4.16-C shows key uniqueness when Random Movement but Same Boundary (RMSB) is provided to every sensor. Figure 4.16-D shows the behavior of network when Same Movement and Same Boundary (SMSB) are provided to every sensor in the network. In all the cases, the performance of the presented system is still better.

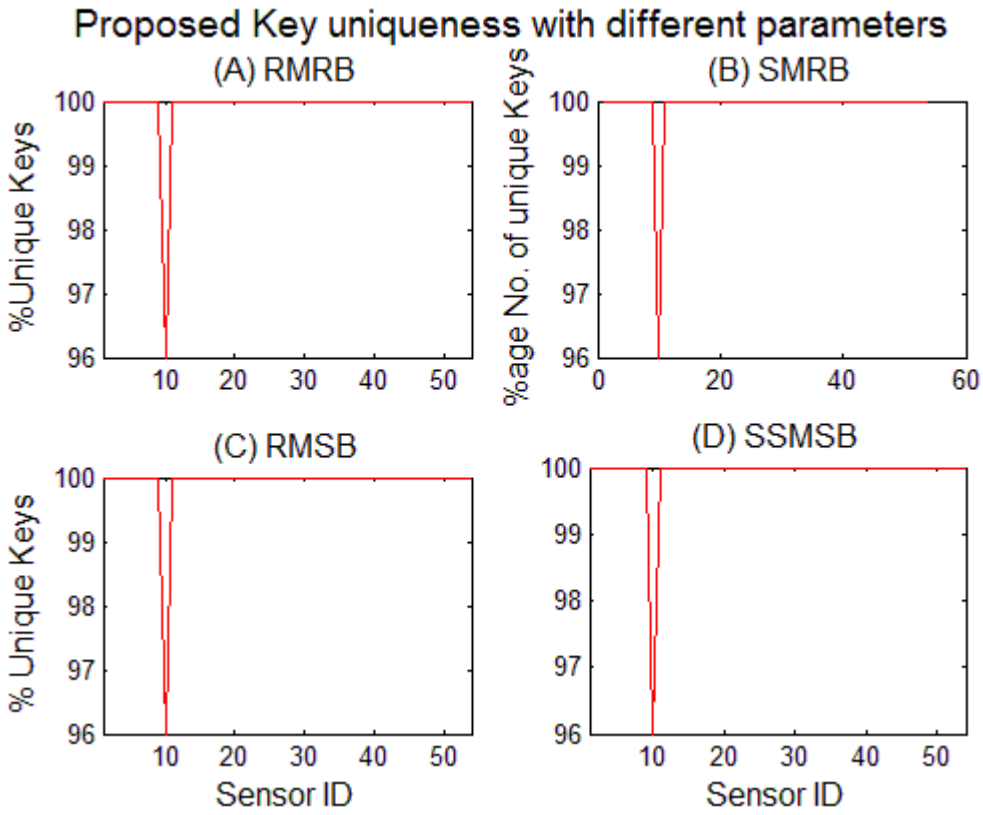

Figure 4.16: Key Uniqueness for Different Key Parameters

4.6. DISCUSSIONS

Presented scheme is better than existing schemes as it has no increased communication overhead and more over it supports multilevel security before compromising data against any attack. The effect of compromising several key generation parameters are given below:-

- *Key compromise:* Suppose that a key of current round is compromised. Keying module in VLKM uses the concept of Folding Addition (FA) which is already a one way function. So it is completely impossible to generate a virtual location with the help of a key.
- *Current virtual location compromise:* Suppose the virtual location of current round is compromised. To generate the next virtual location, several other parameters are necessary, i.e. virtual boundary, virtual angle of movement and virtual speed of movement which is never being communicated over the network. So with the help of

only current virtual location, it is impossible to generate virtual locations for next round or previous round.

- *Virtual location compromise:* Suppose the virtual location that is used in keying module is compromised. Now to generate the key, several other parameters are also necessary, i.e. initial virtual location that is mapped onto virtual origin of cluster (if sensor key is generated) or virtual origin of network (if cluster key is generated). So again with the help of a virtual location, it is impossible to generate a key.
- *All Key parameters of a sensor are compromised:* Suppose all the key generation parameters used to generate a key are compromised. Now the parameters of one sensor are different than other sensor in the network and similarly the parameters of one cluster are different than other cluster. So parameters of a one sensor never estimate the parameters of other sensor or cluster. The key of that sensor is also compromised till a new virtual location is not allocated to that sensor. Once a new virtual location is allocated to that sensor or the virtual origin is updated, compromised key parameters are changed, thus old parameters are useless.

4.7. SUMMARY

In this chapter, an energy-efficient Virtual location based key management (VLKM) scheme has been described for WSNs. The scheme significantly reduces the number of transmissions needed for rekeying. The presented scheme is very suitable for clustering environment where more than one key is required to every sensor for its functioning in the network. Simulation results show that the presented scheme performs better without increasing the communication overheads.

In the next chapter, a scheme for key updation for removing & replacement of compromised sensor nodes from wireless sensor networks has been presented to remove a compromised sensor node from the network in such a way that it maintains the principle of Forward and Backwards secrecy.