Secure IoT Using Weighted Signed Graphs

Pinaki Sarkar[1(✉)] and Morshed Uddin Chowdhury[2]

[1] Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, Karnataka, India
pinakisark@csa.iisc.ernet.in
[2] School of Information Technology, Deakin University,
Burwood Campus, Burwood, VIC, Australia
morshed.chowdhury@deakin.edu.au

Abstract. Key management has always remained a challenging problem for the entire security community. Standard practice in modern times is to agree on symmetric keys using public key protocols. However, public key protocols use heavy computations; rendering them inappropriate for application to low cost devices of Internet of Things (IoT). This led to proposals of various key management strategies for low cost networks; a prominent discovery being key predistribution technique for Wireless Sensor Network (WSN)–a prototype of IoT. Such schemes require several communicating nodes to share the same cryptographic key. This leads to interesting (combinatorial) graphical models and related optimality problems, that get intense for hierarchical architecture. Most protocols meant for hierarchical (low cost) networks employ separate designs for individual levels and/or clusters. Consequently only local optimal values can be computed. We develop a single universal platform using weighted signed graph (WSG) that designs the entire network for a hierarchical setup. This model can be used as itself or clubbed with a key predistribution scheme (KPS) to enhance the latter's security when applied to a WSN. After generic presentation, we combine our universal model with prominent KPS to facilitate comparative study with existing protocols.

Keywords: IoT · WSN · Weighted Signed Graph · Key management · Smart attacks

1 Introduction

Internet of Things (IoT) is a sophisticated concept that aims to connect our world more than we ever thought possible. IoT employs various types of devices to gather information about physical surroundings, process them and communicate these data intelligently among themselves before sending feedback to an end user. Since devices can be resource constrained and are expected to exchange large volumes of data, communication and storage overheads should be minimized. Prominent applications of IoT are smart homes, smart cities, smart grids, smart water networks, vehicular networks, peer-to-peer (P2P) networks, agriculture, health-care, etc.

Wireless Sensor Networks (WSN) are nice prototype of IoT. They are regarded as revolutionary information gathering systems owing to their easy deployment and

R. Deng et al. (Eds.): SecureComm 2016, LNICST 198, pp. 241–256, 2017.
DOI: 10.1007/978-3-319-59608-2_13

flexible topology. They consists of numerous low-cost identically resource starved wireless devices (sensors or nodes) that deal with sensitive IoT data. WSN finds wide-scale applications in military and scientific arenas (listed above) where security is a premium.

Due to resource constraints in constituent sensors, symmetric key cryptosystems (SKC) are preferred over a public key setup in such networks. SKC schemes requires both sender and receiver to possess same encryption-decryption key before message exchange. This is achievable by various techniques; key predistribution being preferred due to cost effective implementation. Ideally any key predistribution schemes (KPS) should have small key-rings, and yet support large number of nodes with appreciable resiliency, scalability and communication probability (or connectivity). However, renowned scientists proved the impossibility of constructing a 'perfect KPS' that meet all these criteria [17, 18, 21]. This motivated proposals of several designs that are robust for specific purpose(s). We try to unify them under a single banner after investigation.

Employing a hierarchy with certain powerful special nodes is perhaps wiser and more practical approach for IoT. An extensive literature survey reveals that all prominent hierarchical schemes [1–4, 10, 23, 25–28] try to glue local and global graphs quite artificially. We try to give a more natural description of the local and global models of any hierarchical network using an uniform banner of Weighted Signed Graph (WSG) and thereby demonstrate the impact on various aspect of such networks. To date, to the best of our knowledge, no scheme represents an hierarchical structure by a single (deterministic) model though there have been some elegant trials [2, 4, 26, 27].

1.1 Motivation and Plan of Work

A critical challenge encountered while designing secure protocols for low cost IoT networks like WSN is to ensure secure communication between two nodes that are not in each other's communication range. Priors works use intermediate users who gets access to these communications in clear text. Our work ascertain that these communications remain protected by use of two different cryptosytems possessing separate keys. This concept is set out in Sect. 6 while applying of our WSG model to low cost networks.

KPS involves preloading of symmetric cryptographic keys before deployment and establishing them immediately after deployment. Use of unique association of keys to their ids that are transmitted in open during key establishment ensure that actual keys are not revealed; though the network graph becomes public. A node's secondary id (set of key ids or their unique function–node ids [25]) is extra information that gets hidden during adaptation of our WSG model. These converted private information of individual nodes are used during key establishment to hide the network graph from an adversary.

This helps to eradicate selective node attack or smart attack. Refer to Sect. 2.

Ruj and Pal [23] state that random graph models are well suited for 'scalability' and 'resilience'. Thereby they try to justify their random designs based on preferential attachment models with degree bounds. Unfortunately, all designs of [23] suffers from

highly skewed load distribution, poor connectivity and resiliency; and hence, are inappropriate for IoT applications. Interestingly, renowned researchers [17, 18, 21, 25] report that deterministic schemes possess advantages like predictable connectivity, resilience, scalability etc., which occurs with certain probability in random ones.

Sensitive IoT applications require protocols to yield connected networks (thereby reduce hops; so attacks). So we opt deterministic protocols for security applications in IoT networks, despite most them having restricted scaling operations. Our WSG model can supports large number of such scaling operations (not unrestricted, though), as demonstrated in Sect. 4.3.

2 Basic Concepts and Definitions

Design of our graphical model requires formalizing of fundamental notions, like hierarchical WSN–various connectivity radii and neighbors; two types of network graphs– global and local for individual clusters; hence respective keys; security models.

- **Hierarchical WSN (HWSN):** A standard method to incorporate an hierarchy in these networks is to inject special (purpose) devices. They are relatively more resourceful than an ordinary sensor; however, much weaker than any Base Station (BS). Such devices are generally called Cluster Heads (CH) or Gateway Nodes (GN). Some authors also term them as super nodes [3] or agents [1, 25].
- **Radius of communication:** of a device is the maximum of the distances that it can transmit and/or receive messages from other devices. This maximum distance, denoted by r_{device}, and is varied for different type of entities. The identical sensors have the least value 'r_{node}'; while any BS has the highest communication radius. The communication radius for any CH (assumed identical for all) is usually greater than the identical nodes, however, less than the BS(s). Since any designer's target is to increase network connectivity, we focus on r_{node} and simply denote as r.
- **Neighboring devices or Neighbors:** two devices with *same* r_{device} are neighbors if they are within communication radius of each other. In case $r_{device_1} < r_{device_2}$ for two devices $device_1$ (say a node) and $device_2$ (say a CH) and distance between $device_1$ and $device_2$ is greater than r_{device_1} but less than r_{device_2}, then $device_1$ is a neighbor of $device_2$ but not otherwise. Providing security to such communication that often happens among varied type of IoT devices is a major challenge.
  More critical challenge is to secure communication of two low power devices that are not neighbors. Priors works usually use intermediate users who gets to see these communications in clear text. Our work ensure that these communications remain protected by recursive use of two different keys of independent cryptosytems.[1]
- **Global, Local graphs, respective keys:** Our model has two types of graphs–global (network) graph and local (cluster) graphs. Any device will be treated as vertex and a link between two devices as an edge between them. For the same pair of vertex,

[1] Use of double encryption requires careful implementation. For instance, double encryption with two smartly chosen AES − 128 keys may enhance the security level by 1.5 times. That is, from 120 − BIT security to approximately 180 − BIT against any present day adversary.

there may be multiple edges–exactly one local and others, global (if any, see Sect. 3). These graphs are used to employ independent cryptosystems; hence respective (complementary sets) of keys. Refer to footnote 1 (above).

- **Local graphs:** are 'in-cluster' or 'local' graphs for individual cluster that depicts the key sharing between devices of a given cluster. A generic construction presented in Sect. 3 denotes the edges of these graphs by negative sign for consistency. Here we assume that all the nodes in a cluster are in each other's communication radius.
- **Global graph and key sharing:** Our weighted signed graph can simultaneously model a complete network. This owes to the fact that the graphical representations supports any number of links (weights) and assigns signs. We exploit the positive sign to denote secure links for pairs of nodes that share a common key, globally. Due to limited memory of any sensor and large sized network, single key must be shared by multiple nodes to assure desired (high) level of (secure) connectivity.
- **Secure communication:** between a pair of entities is assured if their communication is secured by some *shared* cryptographic key, either local or global (or both).

Sharing of global (network) keys leads to a certain weakness in any protocol, specially considering that IoT networks are vulnerable to device compromise. We consider the robustness of our networks against two such attacks, as described below:[2]

- **Random node compromise:** may lead to partial disclosure key-rings of existing devices; thereby restrict the use of links that were secured by these keys. A system's resilience against such an attack may be measured by a standard metric, viz., *fail(s)* which estimates the ratio of links broken of non-compromised nodes to the remaining number of links in the network after random compromise of s nodes.
- **Smart Attack [22] or Selective Node Capture Attack [25]:** Essentially in this type of attack, an attacker tries to break communications of two specific nodes by selectively capturing other node(s) that share the same key(s) being used for communications of the former nodes. This happens because sharing of (global) keys usually leads to exchange of *(unencrypted) key ids* during key establishment. This reveals the *global key sharing graph*. This key sharing knowledge may aid an adversary in selectively (or smartly) targeting specific nodes that share key(s) with the communicating nodes. All existing works, for instance [5, 7–14, 16–19, 21–25, 28, 29] are prone to this attack. Readers are referred to [22] for a more technical definition.

3 Graphical Model and Its Representation

We use weighted signed graph to design an entire hierarchical network. Enumerate each ordinary network user (node for a WSN) by a specific *id (node id./node no.)*. Special purpose users (CHs for a WSN) are separately enumerated. Their enumeration is prefixed with specific number of 0's denoting their height above the node level. The following convention are adapted while defining our local and global graphs:

[2] This can be best analyzed by employing a particular KPS as a candidate for our global graph.

- Denote a *link* between two specific entities of the global (distributed) graph as: **(lower entity no.)(e)(higher entity no.)**. In most case, we will only have **(lower node no.)(e)(higher node no.)**. This is because we do not allow hierarchical communication using global keys. This will make our model have wide-spread applicability to any distributed system. Such networks employ a fixed cryptosystem with single type of key; for instance, application of KPS in WSN.
- *Local graphs* have two types of entities, viz several lower level devices (nodes/CHs) and an upper level user (corresponding CH/BS). So we need to define separate links for distinct type of connections:
- *Local graph links or simply local links* that involve *user at different level* shall be denoted by: **(−)(lower level entity no.)(e)(higher level entity no.)**. Resultant local links triplet for a CH at penultimate level (one level above ordinary node level) and its CH at two level above node level is: −(0)(*CH no.*)(*e*)(0)(0)(*CH no.*); while that of a node and its CH (one level above the level of this ordinary node) is: −(*node no.*)(*e*)(0)(*CH no.*). Specifically, connectivity link between any Node *i* under its CH *A* is denoted by $-ie0A$. Here '−*ve*' implies local link.
- *Local links* involving *same level* entities are denoted by: **(−)(lower entity no.)(e)(higher entity no.)**. We use ordering of node id (or CH id) due to global network to ensure unique representation of this link and '−*ve*' sign implies local link.[3]
- *Local link* between *CH A* and *CH B* is denoted by $-0Ae0B$ whenever $A < B$. '−*ve*' denotes that this edge is due to local graph at cluster head level. Convention prefix of '0' symbolizes that these CHs are hierarchically one level above nodes.

These definition have canonical generalization for (corresponding) users higher up in the hierarchy. Parenthesis are used for clarity of representation and shall be dropped later when there is no ambiguity. That is, for consistency entities in each hierarchical level adds a 0's to the prefix of the already available representation of the links. The following example clarifies the concept:

1. Suppose we want global link between nodes 2 and 5, then their edge or link is represented by: **(2)(e)(5)** or simply **2e5**.
2. various local links between various devices are as below:
   - Same level local link between nodes 7 and 3 is represented by: **−(3)(e)(7)** or simply **−3e7**. Figure 1 depicts the scenario pictorially.
   - Connectivity link between a node (say, 2) and its cluster head (CH 1) is $-2e01$.
   - link between CH 2 and CH 1 is $-01e02$. Observe order of representation in each.

Parallel edges/links: occur between a pair of nodes when they simultaneously possess both local and global connectivity links. Evidently, they must be in the *same cluster* (refer to Fig. 1) and share a global key. One edge (global link) will thus be represented by a *positive ('+')* sign and another edge (local link) will have *negative ('−')* sign. Since the local graphs results in less key sharing (in fact, pairwise key sharing in most cases), using this local key may provide optimal security. In case, pairwise key sharing

[3] Usually node ids are positive number (like KPS applications). Therefore 0 or negative numbers are not used for global links. So we make extensive use of 0 and −*ve* sign for our local graph.

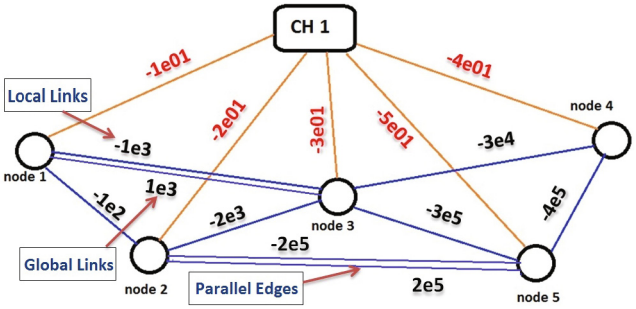

Fig. 1. Connectivity between nodes of same cluster: parallel edges

is not assured by the local graphs (storage factor), use of hash of both the local and global keys may give ideal resilience (up to security of underlying cryptosystem).

Suppose, for simplicity, that nodes with *ids* 2 and 5 belong to same cluster and share both global and local link. Thus there will be *two* parallel edges or links connecting these two nodes. The global link will be denoted by $2e5$, while the local edge/link will be represented by $-2e5$. Note the difference in sign. We propose use of the local link, i.e. $-2e5$ in case this is achieved pairwise; else use hash (shared local keys$\|$global key).

Sharing of multiple global keys is another case of occurrence of parallel links between a pair of nodes. An example of this situation is to consider the global graph to be a KPS design [24] where each pair of nodes share a minimum of 4 keys. This will give rise to a minimum of 4 global key links. In such multiple key sharing cases, a standard method [11] is to use hash of all the shared keys to effective obtain one global link.

4 Design of Key Management Scheme

We propose a key management scheme that may find suitable applications in Internet of Things (IoT). Primary focus is on low cost networks–WSN being a prototype. Basic devices in such low cost IoT networks are resources starved; example ordinary nodes of a WSN. This imposes certain restrictions while designing key management schemes for such low cost networks; forcing us to opt for key predistribution strategies. So we aim to apply our weighted signed graph design to a key predistribution scheme and ensure enhanced security with minimum strain on ordinary devices. This upper bounds the key storage in individual devices (*key-ring*) and imposes a degree bound on each of them.

4.1 Keyrings and Degree Bound on Devices

Most key management protocols for IoT [1, 3, 5–14, 16–19, 21–25, 28, 29], distributed or hierarchical, allow intermediate entities to access communication of non-neighboring devices in clear text. This is because either there was single global graph model (example, any KPS) or in case of two distinct graphical models (in-cluster and global), their application did not complement each other. Thereby simultaneous use of two separate cryptosystem and hence repeated encryption was prohibited. Though some odd schemes like [2, 4, 26, 27] uses repeated encryption, their constructions are actually super-imposition of three local graphs and a global. All existing protocols can now be visualized as subgraphs of our WSG applied to individual schemes. By nature, WSG permits titanic pliability and can support wide range of designs for arbitrary networks; however constraints of devices of particular application platform may compel additional restrictions.

Bearing in mind the constraints in resource of ordinary devices of a low cost IoT network, we propose applications of weighted signed graphs with degree bound ($WSG - DB$) to design security model for such networks. For this, a bound is first fixed on the maximum number of keys k a node can have and the maximum number r of nodes on the cycle of each key; thereby fixing the degree of each device and the maximum degree (d_{max}) that the graph can have. We primarily try to assume a reasonable (uniform) bound for d_{max} of ordinary nodes as their resources are at premium; whereas the value of d_{max} may be much greater for relatively resourceful (fewer) CHs. For the sake of simplicity of computation, let every ordinary device have k_{user} keys and each key cycle be r so that $d_{max} = d_{user} = rk_{user}$. It is easy to see that a node's maximum degree $d_{max} = d_{user} = rk_{user}$ is obtained in case any two pair of nodes intersect in exactly one local and/or global key. While combining with a suitable KPS, this optimality condition may be exploited. Observe that r is also assumed same, and not individual cycles (r_l and r_g) of local and global keys for simplicity.

4.2 Distribution of Local and Global Keys

WSG gives enormous flexibilities and can support wide range of designs; a particular hierarchical one meant for resource constraint IoT (for instance, WSN) networks is being discussed here. For a given user, let $k_{l_{user}}$:= number of keys preallocated for its local (in-cluster) connections and $k_{g_{user}}$:= keys for a node's links due to the global graph (say a KPS). Evidently, $k_{user} := k_{l_{user}} + k_{g_{user}}$. For practical applications (to be discussed in Sect. 6), we assume $k_{l_{user}} = k_{g_{user}} \Longrightarrow k_{user} = 2k_{l_{user}} = 2k_{g_{user}} (= k, \text{say})$. To overcome storage problem, we preallocate as many local keys $(\frac{k}{2})$ as global; so that an upper bound is $\frac{d_{max}}{2r}$ for each node.

Establishment of global keys using their unique *ids* is the essence of any key pre-distribution and leads to interesting combinatorial graph problems. Whereas, local keys need not be established as the number of nodes per clusters is orders less than that in entire network (see Sect. 5 below). Local keys can be used to secure key establishment of global keys (set out in Sect. 6). These keys can also form independent (hierarchical) key predistribution systems themselves; but best effects occur when

combined with keys of a global graph like a KPS. Our applications assume $r_l = 2$ during initial deployment, that is, local graphs for every cluster yield a pairwise key distribution.

4.3 Scalability: Addition of Nodes and/or CHs

Topology of most IoT networks is fast changing due to frequent node movements, addition, deletion and/or compromise. Our WSG model supports both deletion and addition of users. Deletion of users in our model is fairly simple–just delete remaining network's edges corresponding to all the keys exposed due to comprise of device(s).[4] Addition of users may increase r values of existing users. Generally r_g is fixed; so effectively the increment may occur in r_l. Employing $WSG - DB$ concept during scaling operations aids in enhancing our existing design as is briefed in Sect. 4.3.

Local graph induced by our $WSG - DB$ allows the network graph to expand with the addition of every extra vertex (node or CH). This is achieved by assigning all inherited global edges and required local edges needed for this new vertex. Addition of new nodes in existing clusters imply setting up new local links with existing nodes and their CH.

Injection of moderate number of ordinary users and barely a few super users may perhaps be managed by repeated use of same local key(s) to connect several users; thereby increasing r_l. Of course we assume that r_g is fixed for all ordinary users including the new comers. Further, assume that k_{user} is also fixed. Therefore the overall degree bound d_{max} of the network implies an upper bound on r_l. Though the global graph is restrictive in its growth (due to fixed $k_{g_{user}}, r_g$), the above scaling strategy results in better scalability for the combined model. Figure 2 explains the scenario graphically. This overcomes the problem of storage cost for devices even during scaling.

Enormous increment of nodes may lead to huge network growth. This can be tackled by new cluster formation via local graphs. These links are governed by the rules defined in Sect. 3. The incoming nodes and CH are injected with all required keys by the BS to connect with its concerned CH. Cluster-wise deployment may ensure proper cluster formation. In case of any 'misplaced node', a key rescheduling technique, set out in Algorithm 1 of Sect. 5, may be invoked to ensure secure communication of this node.

5 Deployment of Nodes and Cluster Heads

We suggest a cluster-wise deployment, i.e. deploy the nodes with their respective CH. Desired cluster formation should result in most cases. Standard methods of *challenge and response* [14, Sect. 2.1] using cluster wise broadcast keys may be adapted to check correctness of deployment, i.e., proper cluster formation. Evidently, instead of totally

[4] We have to rely on standard intrusion prevention system and/or traitor protocols like [15, 20] for updated information about compromised nodes to facilitate their deletion.

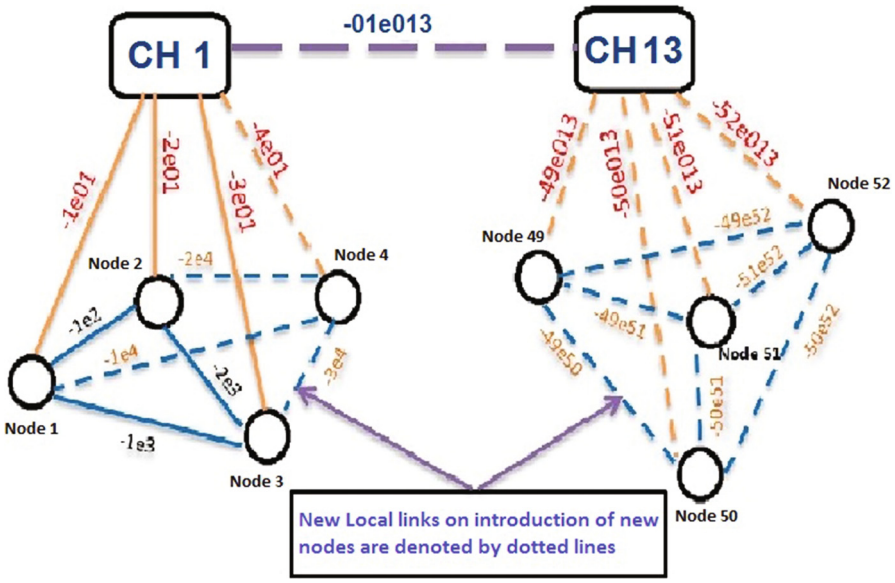

Fig. 2. Communication of (distant) Nodes N_i, N_j of different clusters using local keys

random deployment, we are proposing *group-wise* or *locally* random deployment of the nodes and respective CH according to the local graph. However their exact positions are still unknown. This implies that *local keys for correctly deployed nodes need not be established* and can be directly utilized; for instance an application may be to establish global keys (with large cycles r_g). However, in an unlikely event of a node falling out of their cluster, we adopt the below described key rescheduling technique:

BS generates and preallocates in each node and CH a *global re-scheduling key* ($\mathbf{rsk_{user}}$). They are used for the specific purpose of cluster formation during initial deployment and subsequent key refreshment as described in Algorithm 1. These keys are temporary and recursive in the sense that: (i) a fresh set of such keys are generated and transferred to all existing as well as in-coming nodes and CH using existing $\mathbf{rsk_{user}}$ keys that are (ii) thrashed soon afterwards during every round of key establishment. As such, there is exactly one such key in every node at any given point of time barring short-lived periods of key refreshment when there are exactly two such. Correspondingly there are $O(N)rsk_{user}$ keys in the BS (since number of CHs is orders less than number of nodes = N). We do not require such keys to be share between CHs and nodes.

Recursive use of rsk_{user} keys expands the local graphs; thereby profound network scaling occurs (see Sect. 4.3). Further, their use adapts our model to *dynamic environment*. For any significant movement of an user (from its initial cluster into another), we perform a fresh 'key rescheduling' process by treating this node as 'misplaced node'.[5]

[5] These processes will be detailed in extended version of this work.

Remark 1. At times, there may be more than one CH in the communication radius of a (misplaced) node. If one of these CHs can properly respond to the *challenge and response* test, this node is in its desired cluster. Otherwise, this is a 'misplaced node'. It uses Global Positioning System (GPS) to find out the closest CH and treats it as the 'new' CH. In case two or more CHs are at minimum distance from this 'misplaced node', we choose any of them (with lesser cluster size) to be its (new) CH.

6 Application to Low-Cost Networks

One standard application of our WSG model can yield a hierarchical key predistribution scheme for a WSN. Any existing (distributed) KPS can play the role of our (underlying) 'global graph' whose security gets enhanced by the application of 'local graph'. To this end, we observe that owing to large network size, keys were predistributed and later established in any existing KPS following the footsteps of the pioneering work [14]. Key predistribution process of our global graph can be treated in a similar manner with secure key establishment due to local keys.[6] We assume that cluster size is small with all nodes being neighbors. Our combined WSG design achieves improvements in:

Each user i sends a challenge text to their respective CH A (see Remark 1) encrypted using (intended) preloaded local (broadcast) key K_o;
if *CH A is able to correctly decrypt the challenge,* **then**
    CH A concludes K_o is shared local key with user i;
    and broadcasts: "User i is correctly placed.";
end
else if *CH A is unable to respond correctly to the challenge,* **then**
    CH A broadcasts this 'location error of user i';
    CH A_0, intended CH of user i receives this broadcast by CH A (like others);
    CH A_0 verifies information by *challenge and response* and confirms to BS;
    CH A_0 uses the link $-0A_0e0A$, $-0A_0e00BS$ to securely transmit K_o to CH A and BS (for escrow);
    Meanwhile, on hearing CH A's broadcast user i encrypts K_o with the key rsk_i that it uniquely shares with the BS;
    This encryption is routed to the BS via CH A (CH A is unable to decrypt);
    BS decrypts this encrypted K_o using rsk_i;
    BS generates a fresh of key K_n for CH A and user i;
    Sends K_n to CH A and user i after encrypting with rsk_A and rsk_i respectively;
    User i and CH A computes fresh shared (local) key as: $K_n \oplus K_o$;
end
A fresh set of 'global rescheduling keys' rsk_{user} are generated;
All the old rsk_{user} keys are revoked immediately afterwards.;

Algorithm 1. Key Rescheduling Protocol For 'Misplaced Nodes'.

[6] Represent global links as (*lower node no.*)(k_i)(*higher node no.*) for $1 \leq i \leq \nu$; $k_1, k_2, k_3, \cdots k_\nu$ are all the keys of selected KPS. This automatically captures the (regular) degree ($r_{KPS} = r_g$) of concerned KPS. Refer to [18, Sect. 2] for this definition of r_{KPS}, where it is denoted as r.

Key Establishment Protocol (KEP) of any existing KPS uses unencrypted transmission of (global) key *ids*. This process reveals the network (key) graph to an eavesdropper and induces *smart attacks*. Introduction of the '*local keys*' facilitates transmission of encrypted key ids during '*(global) key establishment phases*' (of our combined KPS); and so, restricts an attacker from equating the *unencrypted key ids*. As a result, an attacker need to break the underlying cryptosystem meant for the local graphs to trace the '*(global) key sharing graph*' of combined networks; unlike previous works. Hence, this novel key establishment technique results in eradication of the pertinent weakness of '*selective node attacks*' or '*smart attacks*', generically prevalent in most existing KPS [5–11, 13, 14, 16–19, 21, 22, 24, 28, 29].[7]

Message Exchange of any existing KPS involves the following steps:

- for neighboring sensors that share a common key (traced during key establishment), a message to be transmitted is encrypt with that common key.[8] Sender node then transmits this encrypted message via wireless channels. Only those receivers who posses the shared keys can decrypt this encryption to recover original text.
- direct encrypted communication is forbidden for sensors that do share any common key or are not in communication radius of each other. An alternate (path key) strategy of routing though other nodes is suggested. This brings in extra complexity of tracing optimized secure path, which is a major challenge for any KPS.

Our WSG based combined model outperforms existing KPS due to supplementary cryptosystem arising from the local graphs. '*Local keys*' may provide unique direct connectivity between nodes and CH of the same cluster; specially during initial deployment. This results in *ideal security* in terms of key distribution; that is, here the system's security depends solely on the underlying (KPS) cryptosystem. Local keys can independently links two nodes i and j of different clusters by use three local links as below:

- local link $-(ie0A)$ between the sender node i and its CH A;
- local link $-(0Ae0B)$ between respective CH A, B of sender and receiver nodes;
- local link $-je0B$ between the receiver node and its CH B.[9]

In case 'misplaced nodes' do not have any shared global key with its (new) neighbors, above process ensures that it is still securely connected to the network. This problem that every node, even misplaced ones, remain securely connected to the network is a challenging one and not many KPS provide adequate solution like we just did.

---

[7] Of course the use of local keys here requires proper cluster formation to ensure desired inter- cluster connectivity. One plausible way to obtain the desired cluster formation is to deploy the nodes and their Cluster Heads in a locally (uniform) random or group-wise random fashion. This assures proper cluster formation in most cases. In a rare event of 'misplaced node', we propose implementation of Key Rescheduling Protocol, described in Algorithm 1.

[8] In the event of (same set of) multiple keys shared between a pair of nodes, a standard method [11] is to concatenate all of these keys and use hash of this concatenated key.

[9] These communications make use of (fixed) publicly available addresses (like MAC or I.P. or email ids) of users (here nodes). Observe that these primary addresses are independent of the created secondary *node ids* [24, 26, 27] used during (global) key establishment.

Local connectivity defined above suffers from 'one point' attacks on the CH, though such an attack may be practically harder to mount on a handful of CH. Previous works generally assume considerable trust on CH and even nodes as message meant for distant nodes are routed via intermediate CH/nodes who can see these message in clear text. To be on safer side, we use an alternate trick of recursive encryption-decryption (maintaining orders) for distant nodes that possess a shared global key; described below:

- a sender (node i) encrypts the message twice; (i) first with the global key shared with intended recipient(s); and (ii) second with local key shared with its parent CH A. Sends the doubly encrypted message to its CH A using the link $-ie0A$ (remember recipient(s) are beyond communication range).
- Sender's parent CH A opens the outer encryption and puts on another encryption using the key shared with recipient's parent CH B. Sends this double encrypted message to recipient's CH B via $-0Ae0B$.
- Receiver's parent CH B opens the outer encryption and puts on another encryption using local key shared with recipient node j and sends to recipient via link $-0Be\,j$.
- Intended recipient node j has both the required local and global keys to decrypt the repeated encryption. Recursive decryption of this doubly encrypted message by: (i) first by the shared local key with its parent CH B and (ii) then the global key shared with sender node i reveals the clear text message.

Figure 2 represents these inter-cluster communications pictorially where Node i communicates with Node j with the KPS link $ie\,j$ (positive sign). This is the only step in our work that involves double encryption and results in highly enhanced network security.

7 Resiliency of Combined Model: Theoretical Analysis

Though most works concerning hierarchy in WSN restricts an attacker from compromising special nodes (CH), we think this assumption is rather strong. We give more power to an adversary. Consider a weaker assumption that an adversary can comprise CH but such captures are relatively harder as compared to compromising nodes. From this section onwards, the symbol 's' will represent number (no.) of nodes compromised, and the symbol 't' is reserved for number of CHs captured in penultimate tier (just above the level of ordinary nodes). So our attack models assumes $s >> t$.

Resiliency of existing KPS are measured by a standard metric '*fail(s)*' that denotes the ratio of links broken after compromise of s nodes to the total links in remaining network. Formally:

$$fail(s) = \frac{\text{number of links broken of non}-\text{compromised nodes due to capture of s nodes}}{\text{total number of links in after compromise of s nodes}}.$$

Evidently, most KPS [6, 7, 16–19, 21] try to minimize this '*fail(s)*' values for their respective schemes. We canonically extend this definition to $fail(s,t)$ as below:

Definition 1. *For the combined (Weighted Signed Graph) system, define fail(s,t) to be*

$$= \frac{number\ of\ global\ links\ broken\ due\ to\ capture\ of\ s\ nodes\ and\ t\ (CH)}{total\ number\ of\ KPS\ links\ in\ after\ compromise\ of\ s\ nodes}.$$

Remark 2. We do not consider links broken in our newly constructed local graphs. Focus is on global (KPS) links since the corresponding keys are shared between many nodes; whereas local keys are uniquely shared (at least initially). Details of scaling effects on resilience of local graphs will be presented in extended version of our work.

Though it is difficult to have an estimate of the exact value of *fail*(*s,t*) when '*s*' nodes and '*t*' CHs of a WSN are captured, we can compute an estimated upper bound of *fail*(*s,t*) of the combined design for a particular chosen KPS scheme.

Theorem 1. *Suppose s nodes of the chosen* KPS *t out of the original c CHs in the last tier of the combined network are captured. Then an upper bound of* fail(s,t) *is*

$$\frac{t}{c} \times fail(s),$$

where fail(s) *is the resiliency of chosen KPS due to the capture of s nodes.*

Proof. Special nodes (like CH for WSN) of the local graphs enable repeated encryption of messages using unique local keys (at least initially). Messages being transmitted always remain encrypted and only the outer encryption of these double encrypted on messages are decrypted and re-encrypted (by different keys). Removal of a CH means that the nodes of that cluster operate with only global keys; so that, extra security due to our WSG model gets negated for nodes of this cluster. Of course, capture of all c CHs at the penultimate level of the hierarchy eradicates every additional security imparted due to the local graphs of WSG model. This (latter) unlikely event reduces the security of this compromised combined system to that of underlying KPS.

Under the standard assumption that nodes are uniformly distributed under their CH in individual clusters, we conclude resilience upon capture of a CH, $fail(s,t) = \frac{fail(s,t)}{c}$. So when t CHs out of total c CHs at penultimate level are compromised, resiliency metric $fail(s,t) = \frac{fail(s,t)}{c}$. Thus, our desired result is achieved. □

7.1 Simulation Results: Comparative Study

We select *TD*(*k, p*) deign of Lee and Stinson [18] with $k = p$ as underlying KPS of our combined model; i.e., KPS based on *TD*(*k, p*) designs our global graph. From here on, by combined protocol, we shall refer to this combination of KPS [18] as global graph and pairwise local graphs (due to negative signs). This combined system has been used to conducted simulations with $s = 100, 200, \ldots, 1000$ and $t = \frac{s}{25}$ under hypothetical conditions that replicate real life scenarios. Results obtained were compared with some prominent existing schemes [11, 13, 14, 18, 25, 28]. Each network is assumed to have roughly $N \approx 10000$ nodes and stated connectivity probability p_c. These results

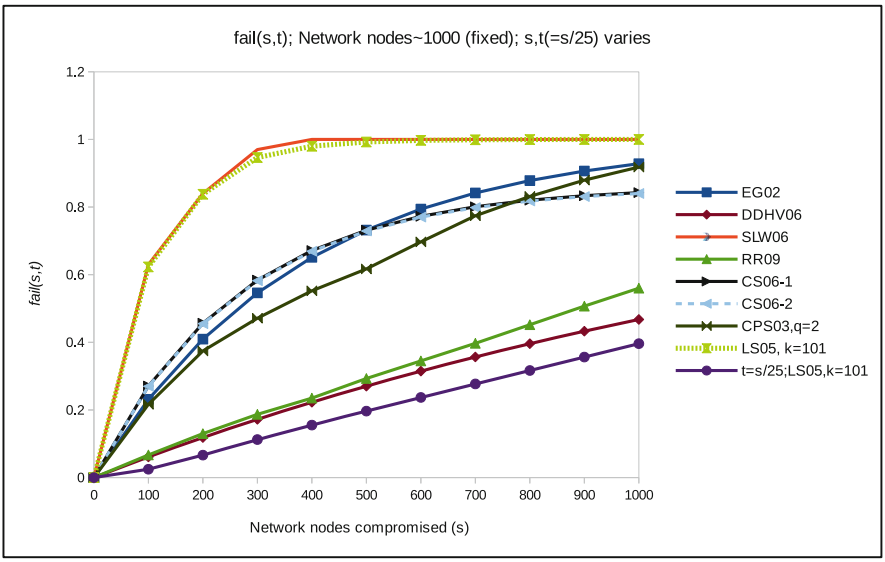

Fig. 3. fail(s,t) comparison for almost equal sized (10000 nodes) networks

presented in Fig. 3 help in visualizing the improvements achieved when our combined scheme is compared with KPS having following set of parameters:

1. Lee and Stinson (LS05) (distributed) scheme [18]: $p = 101$, $k = 101$, $p_c = 0.99$ and $N = 10201$;
2. Proposed combined scheme: $p = 101$, $k_{TD} = 202$, $t = \frac{s}{25}$, $p_c = 1$ and $N = 11040$;
3. Chakrabarti and Seberry (hierarchical) schemes [10] with 25 nodes compromised per cluster ($s_{cluster} = 25$) and CH compromised = (4% of nodes compromised of entire network) $t = 4, 8, \ldots, 40$:
   (a) CS06-1 where both tiers are based on symmetric BIBD [7] (construction extended over F_{p^z}); parameter: $p^z = 11$, 9 for node and CH levels respectively. Therefore, key-rings of nodes: = $k_{nodes} = 12$ and key-rings of CH: = $k_{CH} = 10$ and $N = 133 * 91 = 12103$;
   (b) CS06-2 uses [18] for lower tier, extended symmetric BIBD [7] for upper; parameter: $p^z = 11$, 9 for node and CH levels respectively. Therefore, $k_{nodes} = 11$ with $k_{CH} = 10$ and $N = 121 * 91 = 11011$;
4. Simonova et al. (SLW06) (location aware) scheme [28]: $k = 16$, $p = 11$, $m = 2$, $N = 12100$ over $TD(k, p)$ KPS [18] with $k = 4$, $p11$, so that $p_c = 0.363$;
5. Ruj and Roy (RR09) (hierarchical) scheme [25]: $n = 143$, $k = 12$, $N = 16093$.
6. Eschenauer and Gligor (EG02) (distributed) scheme [14]: $k = 263$, $p_c = 0.5$, $N = 10000$.
7. Chan et al. (CPS03) q-composite (distributed) scheme [11]: $q = 2$, $k = 263$, $p_c = 0.5$, $N = 10000$.
8. Du et al. (DDHV06) (location aware) scheme [13]: $k = 67$, $p_c = 0.5$ and $N = 10000$;

8 Conclusion and Future Work

This paper proposes a universal design to model any hierarchical graph. This design is based on Weighted Signed Graph (WSG). Such a model is particularly useful in designing key management schemes for low cost networks. As an application, we select any popular key predistribution scheme (KPS) that are particularly useful in resource starved environment of WSN–a prototype of IoT. Comparative study of a fixed KPS as the global (inter cluster) graph with a pairwise local (intra cluster) graph establishes the superior performance of our scheme with prominent existing schemes.

Due to page limits, we leave detailed study of scalability (thanks to local/cluster graph) of combined network for the extended version of this paper. We shall analyze the side effects of scalability on resiliency for fully connected resultant networks.

References

1. Bag, S.: A new key predistribution scheme for grid-group deployment of wireless sensor networks. Ad Hoc Sens. Wirel. Netw. **27**(3–4), 313–329 (2015)
2. Bag, S., Dhar, A., Sarkar, P.: 100% connectivity for location aware code based KPD in Clustered WSN: Merging Blocks. In: Information Security Conference (ISC) 2012, Passau, Germany, pp. 136–150 (2012)
3. Bag, S., Roy, B.K.: A new key predistribution scheme for general and grid-group deployment of wireless sensor networks. EURASIP J. Wirel. Commun. Networking **2013**, 145 (2013)
4. Bag, S., Saha, A., Sarkar, P.: Highly resilient key predistribution scheme using transversal designs and reed muller codes for wireless sensor network. In: Wyld, D.C., Wozniak, M., Chaki, N., Meghanathan, N., Nagamalai, D. (eds.) CNSA 2011. CCIS, vol. 196, pp. 344–355. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22540-6_33
5. Banihashemian, S., Ghaemi Bafghi, A., Yaghmaee Moghaddam, M.H.: Centralized key management scheme in wireless sensor networks. Wirel. Pers. Commun. **60**(3), 463–474 (2011)
6. Bose, M., Dey, A., Mukerjee, R.: Key predistribution schemes for distributed sensor networks via block designs. Des. Codes Cryptogr. **67**(1), 111–136 (2013)
7. Çamtepe, S.A., Yener, B.: Combinatorial design of key distribution mechanisms for wireless sensor networks. In: ESORICS 2004, French Riviera, France, pp. 293–308, 13–15 September 2004
8. Chakrabarti, D., Maitra, S., Roy, B.: A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 89–103. Springer, Heidelberg (2005). doi:10.1007/11556992_7
9. Chakrabarti, D., Maitra, S., Roy, B.K.: A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design. Int. J. Inf. Sec. **5**(2), 105–114 (2006)
10. Chakrabarti, D., Seberry, J.: Combinatorial structures for design of wireless sensor networks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 365–374. Springer, Heidelberg (2006). doi:10.1007/11767480_25
11. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy, pp. 197–213. IEEE Computer Society (2003)

12. Dhar, A., Sarkar, P.: Full communication in a wireless sensor network by merging blocks of a key predistribution using reed solomon codes. In: CCSAE, pp. 389–400 (2011)
13. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A key predistribution scheme for sensor networks using deployment knowledge. IEEE Trans. Dependable Sec. Comput. **3**(1), 62–77 (2006)
14. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: ACM Conference on Computer and Communications Security, pp. 41–47 (2002)
15. Fiat, A., Tassa, T.: Dynamic traitor tracing. J. Cryptology **14**(3), 211–223 (2001)
16. Henry, K., Paterson, M.B., Stinson, D.R.: Practical approaches to varying network size in combinatorial key predistribution schemes. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 89–117. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43414-7_5
17. Lee, J., Stinson, D.R.: On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. ACM Trans. Inf. Syst. Secur. **11** (2), 1–35 (2008)
18. Lee, J., Stinson, D.R.: A combinatorial approach to key predistribution for distributed sensor networks. In: IEEE Wireless Communications and Networking Conference WCNC 2005, New Orleans, USA, pp. 1200–1205, 13–17 March 2005. Invited Paper
19. Martin, K.M., Paterson, M.B., Stinson, D.R.: Key predistribution for homogeneous wireless sensor networks with group deployment of nodes. TOSN **7**(2) (2010)
20. Newman, R.: Computer Security: Protecting Digital Resources. Jones & Bartlett Learning, Sudbury (2009)
21. Paterson, M.B., Stinson, D.R.: A unified approach to combinatorial key predistribution schemes for sensor networks. Des. Codes Cryptogr. **71**(3), 433–457 (2014)
22. Pietro, R.D., Mancini, L.V., Mei, A.: Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. Wirel. Netw. **12**(6), 709–721 (2006)
23. Ruj, S., Pal, A.: Preferential attachment model with degree bound and its application to key predistribution in WSN. In: 30th IEEE International Conference on Advanced Information Networking and Applications, AINA 2016, Crans-Montana, Switzerland, 23–25 March 2016, pp. 677–683 (2016)
24. Ruj, S., Roy, B.: Key predistribution using partially balanced designs in wireless sensor networks. In: Stojmenovic, I., Thulasiram, R.K., Yang, L.T., Jia, W., Guo, M., Mello, R.F. (eds.) ISPA 2007. LNCS, vol. 4742, pp. 431–445. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74742-0_40
25. Ruj, S., Roy, B.K.: Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks. TOSN **6**(1), 4:1–4:28 (2009)
26. Sarkar, P., Saha, A.: Security enhanced communication in wireless sensor networks using reed-muller codes and partially balanced incomplete block designs. JoC **2**(1), 23–30 (2011)
27. Sarkar, P., Saha, A., Chowdhury, M.U.: Secure connectivity model in Wireless Sensor Networks (WSN) using first order Reed-Muller codes. In: MASS, pp. 507–512 (2010)
28. Simonova, K., Ling, A.C.H., Wang, X.S.: Location-aware key predistribution scheme for wide area wireless sensor networks. In: ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2006), Alexandria, VA, USA, 30 October 2006, pp. 157–168 (2006)
29. Zhou, L., Ni, J., Ravishankar, C.V.: Supporting secure communication and data collection in mobile sensor networks. In: 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2006, 23–29 April 2006, Barcelona, Catalunya, Spain (2006)