Cooperative secret delivery in wireless sensor networks

Jing Deng*

Department of Computer Science,
University of North Carolina at Greensboro
Greensboro, NC 27412, USA
E-mail: jing.deng@uncg.edu
*Corresponding author

Yunghsiang S. Han

Department of Electrical Engineering,
National Taiwan University of Science and Technology,
#43, Sec.4, Keelung Rd., Taipei, Taiwan
E-mail: yshan@mail.ntust.edu.tw

Abstract: In the technical literature, many random key pre-distribution techniques have been proposed to secure wireless sensor networks (WSNs). Such techniques only establish keys for some pairs of physically connected sensors. On the contrary, in many WSN applications, a source node must securely communicate with all of its neighbours. In this work, we address the issue of delivering secret link keys to each of the source's neighbours in WSNs. We propose a scheme called cooperative secret delivery (CSD) that finds bridge nodes to deliver keys to the insecure neighbours, which share no prior keys with the source. The novelty of our scheme is to deliver multiple keys through these bridge nodes and regular paths instead of multi-hop secure paths. Since the secret link keys are delivered cooperatively by the bridge nodes in the CSD scheme, secret disclosure probability is significantly lower compared to other delivery techniques. We perform extensive theoretical analysis and performance evaluation on the CSD scheme and demonstrate its salient features.

Keywords: WSNs; wireless sensor networks; key establishment; symmetric encryption; multiple path; routing; fault tolerance.

Reference to this paper should be made as follows: Deng, J. and Han, Y.S. (2013) 'Cooperative secret delivery in wireless sensor networks', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 14, No. 4, pp.226–237.

Biographical notes: Jing Deng is an Associate Professor in the Department of Computer Science at UNC Greensboro. He visited the Department of Electrical Engineering at Princeton University and the Department of Electrical and Computer Engineering, WINLAB at Rutgers University in Fall of 2005. He was with Syracuse University and the University of New Orleans and Syracuse University between 2002 to 2008. He received his PhD Degree from Cornell University in 2002. He is an associate editor of IEEE Transactions on Vehicular Technology. His research focuses include wireless network security, information assurance, mobile ad hoc networks, and wireless sensor networks.

Yunghsiang S. Han received PhD Degree from the School of Computer and Information Science at Syracuse University in 1993. He was, from 1993 to 1997, with Hua Fan University, with National Chi Nan University from 1997 to 2004, and with National Taipei University from 2004 to 2010. From August 2010, he is with the Department of Electrical Engineering at National Taiwan University of Science and Technology. He has published several highly cited works on wireless sensor networks and serves as the editors of several international journals. He was the winner of the Syracuse University Doctoral Prize and a Fellow of IEEE.

1 Introduction

Wireless sensor networks (WSNs) are formed by large number of wireless micro-sensors that can sense the environment and send data towards information sinks autonomously (Akyildiz et al., 2002). WSNs have attracted significant interests from the research community due to their potentials in a wide range of applications such as environmental sensing, battlefield sensing and hazard leak detection. When these miniature sensor devices are deployed to the environment with potential eavesdroppers, information delivery must be protected with keys. Due to the lack of trusted servers nearby (for public/private key schemes), secret key schemes are considered to be more viable to protect such local communications. The question is how to deliver such secret keys to the communicating nodes.

In 2002, Eschenauer and Gligor proposed a random key pre-distribution technique. In this technique, each sensor pre-loads a subset of keys (key ring) from a large key pool before deployment. Some secure connectivity will then be in place after deployment. It has been shown that when the size of the key ring and the size of the key pool are chosen carefully, the neighbouring sensors will have a non-negligible probability of sharing common keys. With such shared keys, the neighbouring nodes can establish secret link key securely (Eschenauer and Gligor, 2002). The scheme has been extended and investigated by researchers in several research groups (Chan et al., 2003; Du et al., 2005; Liu and Ning, 2003; Huang et al., 2007; Huang and Medhi, 2007; Fu et al., 2008; Deng and Han, 2008; Liu et al., 2009; Mittal and Novales, 2010). However, other local links are still disconnected on the security plane. The lack of such secure links can be thought as intentional. This is because a higher local connectivity (on the security plane) would mean higher vulnerability and lower network resilience, as shown in Eschenauer and Gligor (2002) and Du et al. (2005).

In this work, we focus on the issue of delivering secret link keys from a source to multiple neighbours that are disconnected from the source on the security plane. We call such neighbours of the source as *insecure neighbours*. Eschenauer and Gligor suggested to use the secure multi-hop path scheme to deliver such secrets (Eschenauer and Gligor, 2002). In their scheme, the secret is disclosed to all the nodes on the path. If any of these nodes is compromised, the secret is disclosed to the adversary. Instead, we propose a new technique called cooperative secret delivery (CSD) that identifies several bridge nodes to help deliver secret link keys to these neighbours. The novelty of our scheme is to deliver multiple keys with the use of these bridge nodes and regular routing paths instead of *multi-hop secure paths*, each of which is formed by a sequence of nodes sharing keys with the direct neighbours. Since the secret link keys are delivered cooperatively by the bridge nodes in the CSD scheme, key disclosure probability is significantly lower compared to other delivery techniques.

We expect our scheme to be implemented in networks where the source node needs to securely communicate with all its physical neighbours. Disclosure of any information sent from the source poses a security threat. An example is the cluster-based WSNs where the cluster head needs to communicate with the sensors in its neighbourhood. Even though similar techniques have been used in other areas such as routing, there has never been any similar approach in secret delivery in key pre-distribution.

Our paper is organised as follows: in Section 2, we present the multiple secret key delivery problem and our CSD scheme design. We analyse the performance of the CSD scheme in communication cost and secret disclosure probability in Section 3. An evaluation of the proposed CSD scheme is presented in Section 4. Section 5 summarises related work and the state-of-the-art in the field. We summarise our work in Section 6.

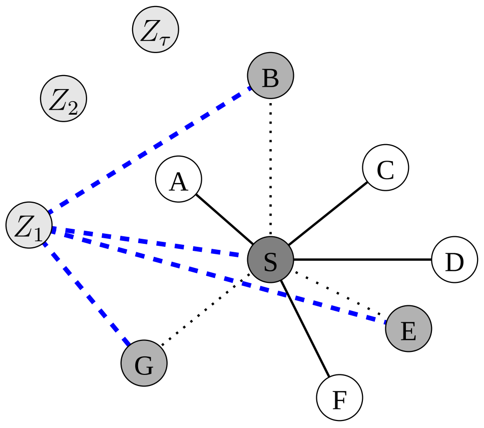
2 The CSD scheme

2.1 Problem formulation

Figure 1 illustrates the problem of delivering multiple secret link keys to neighbouring nodes. Node S has several physical neighbours, which include nodes A, B, C, D, E, F and G, clockwise. The solid lines in Figure 1 represent both security and physical connectivity. The dotted lines represent only physical connectivity. Therefore, links S-B, S-E and S-G are disconnected on the security plane. Nodes B, E and G are termed 'insecure neighbours', $\mathcal{N}_{in}$ of node S.[1]

In order to secure the data transmission between node S and all its neighbours with symmetric encryption/decryption technique, we need to establish a secret link key for each of the physical links connecting node S. In Figure 1, the secret link key between node S and nodes A, C, D and F can be set up easily with the shared keys. However, it is more challenging for node S to establish secret link keys with nodes B, E and G.

Figure 1 Illustration of the security connectivity around node S. Solid lines connect nodes that are direct neighbours and share at least one common key. Dotted lines connect physical neighbours but they do not share any common key. Nodes $Z_1, Z_2, \cdots Z_\tau$ are the bridge nodes, which share keys with node S and some of the insecure neighbours. Such security connectivity is represented with dashed lines. The connectivities of node $Z_2, \cdots Z_\tau$ are omitted for clarity (see online version for colours)



Problem statement (Delivery of multiple secret link keys): When random key pre-distribution techniques are employed in WSNs, it is possible that sensor nodes do not share keys with several neighbouring nodes. In order to establish secure communication for these links, secret link keys must be delivered to these neighbours. How should these secret link keys be delivered efficiently and securely?

Note that some existing delivery techniques, e.g., the multi-hop secure path technique (Eschenauer and Gligor, 2002; Chan et al., 2003; Mishra et al., 2012) and the Babel scheme (Deng and Han, 2007), can deliver these secret link keys from the source to the insecure neighbours. However, some of these schemes can be easily disrupted by a single compromised node (e.g., Chan et al. (2003)). Others may suffer from Denial-of-Service (DoS) attacks (e.g., the Babel scheme in Deng and Han (2007)). Instead, an efficient scheme that is resilient towards node compromises and eavesdropping should be designed. We further assume the network is not sparse such that network partitions appear (Li et al., 2012).

2.2 The CSD scheme through an example

In this work, we propose the CSD scheme to deliver multiple secret link keys towards the insecure neighbours. Our main idea is to find several bridge nodes that share keys with the source node and some of its insecure neighbours. These nodes are termed *bridge nodes*, which will be employed to deliver secret keys. All nodes in between only serve as passive routers without knowing the secrets that they help to forward. We will show that this approach improves efficiency, lowers the secret disclosure probability as well as provides protection against eavesdropping and DoS attacks.

We explain the CSD scheme through the example shown in Figure 1. Formal presentations including algorithms will be presented in Section 2.3. Denote the set of keys stored on node i as K_i, as explained in Eschenauer and Gligor (2002). We use the notations given in Table 1 in our example and in pseudo-codes.

Table 1 The notations used in algorithms

λ	the number of keys carried by each node
S	source node
$\mathcal{N}_{in}$	set of insecure neighbours of S
m	size of the set $\mathcal{N}_{in}$
τ	number of bridge nodes found and used
Z_i	the i-th bridge node, $i \in \{1, 2, \ldots, \tau\}$
$K_{i,a}$	keys carried by node i, $i \in \{S\} \cup \mathcal{N}_{in}$, $a \in \{1, 2, \ldots, \lambda\}$
$\|\|$	Concatenation operation
$\{msg\}_K$	encryption of message msg using key K
$N_{i,a}$	Nonce generated by node i for its ath key, $K_{i,a}$
$C_{i,a}$	Encrypted challenge by node i for its ath key, $K_{i,a}$
$M_{i,a}$	Message with the encrypted challenge by node i for its ath key, $K_{i,a}$
$N'_{i,a}$	Response generated by a potential bridge node towards $N_{i,a}$
$C'_{i,a}$	Encrypted response generated by a bridge node for key $K_{i,a}$
$M'_{i,a}$	Message with the encrypted response by a bridge towards node i ($K_{i,a}$)
TTL	time-to-live, the maximum number of hops for a request message to travel

Node S collects the Message Authentication Codes (MACs) of a challenge message based on each of the keys in K_S, K_B, K_E, and K_G.[2] These information are flooded in the network. While flooding may introduce vulnerabilities for DoS attackers, we argue that many network functions use flooding and, therefore, mitigation techniques should be in place and are out of the scope of this paper. Each overhearing node compares the MACs of the message based on its carried keys and those on the overheard message. Only those nodes sharing a key with the source and at least ϵ portion of the insecure neighbours will respond. As an example, we assume that node Z_1 satisfies the above condition, i.e., node Z_1 shares at least one key with node S and at least ϵm of the insecure nodes (note that m represents the number of insecure neighbours of node S). The shared keys could be different or the same among such nodes. It will respond and volunteer to serve as a bridge by replying to node S. The reply message includes its response to each of the challenges with the shared keys.

Every node that does not satisfy the shared key criteria forwards the message by attaching its ID to the end of the message (for path record). Controlled flooding is employed to propagate the message, i.e., it may only travel up to a certain number of hops, termed time-to-live (TTL). The message is discarded when it has been forwarded TTL times.

Let us assume that node Z_1 shares keys with nodes S, B, and G, but not with node E. Note that node Z_1 may be physically multi-hop away from the source node. When the reply from node Z_1 arrives at node S, it sends the reply to the insecure neighbours, i.e., nodes B and G, for verification. The corresponding insecure neighbours validate the responses and ensure that the bridge node does share keys as claimed. After the insecure neighbours verify such claims, they notify node S, which will prepare secret components to be delivered towards nodes B and G through node Z_1. Such components will be encrypted with the shared key between nodes S and Z_1.

Upon receiving the message with the encrypted secret components, node Z_1 decrypts the message and re-encrypts the secret components with the shared keys with nodes B and G, respectively. For example, the secret component between nodes S and B will be encrypted at node Z_1 with the common key between nodes Z_1 and B. These re-encrypted secret components are then returned to node S. Node S forwards the encrypted secret components to nodes B and G, which decrypt the secret link key components.

2.3 The CSD scheme

Assume that τ of these bridge nodes exist and denote them as node Z_1, Z_2, ... Z_τ. We further assume that bridge node Z_j, $j \in \{1, 2, \cdots \tau\}$ shares keys with node S and n_j nodes in $\mathcal{N}_{in}$ and denote them as $i_{j,1}, i_{j,2}, \cdots i_{j,n_j}$. The details of the CSD scheme are explained in the following:

Node S sends all the challenges from each $i \in \mathcal{N}_{in}$ to the entire neighbourhood. Assuming that each node Z_j, $j \in \{1, 2, \cdots \tau\}$ shares a key with node S and at least ϵ portion of nodes in $\mathcal{N}_{in}$, it submits responses to the nodes in $\mathcal{N}_{in}$ that it shares a key with. In particular, it submits a response to nodes $i_{j,1}, i_{j,2}, \cdots i_{j,n_j}$.

When these responses are received by node S, it distributes them to each node in $\mathcal{N}_{in}$, which will verify the response and report results to node S. Node S then prepares a matrix of codes to be transmitted through these bridge nodes towards $\mathcal{N}_{in}$. We denote this matrix as M and its dimension is $\tau \times m$, where $m = |\mathcal{N}_{in}|$. Specifically, $M(j, i)$ represents the codes to be sent to Z_j towards $i \in \mathcal{N}_{in}$. Note that only when Z_j and $i \in$

$\mathcal{N}_{in}$ shares a key and Z_j shares a key with node S, the element $M(j,i)$ is encoded. Otherwise, it is empty. The preparation of matrix M is basically an encoding process, which we will explain in Section 2.5.

Node S concatenates all non-empty elements on row j and encrypts the result with the shared key between itself and node Z_j. This encrypted information is transmitted to node Z_j, which will decrypt and separate the elements. Z_j then encrypts the elements with the shared key with the corresponding receiver of each of the elements, $i \in \mathcal{N}_{in}$. The encrypted results are then concatenated and encrypted with the shared key between node Z_j and S and are sent back to node S.[3] Node S decrypts the secret and distributes each encrypted element to their corresponding receiver, $i \in \mathcal{N}_{in}$. The new shared secret between node S and node $i \in \mathcal{N}_{in}$ will be computed as

$$k(S,i) = F(I(i)), \tag{1}$$

where Z_j and i share a key, $I(i)$ contains all non-empty elements in the column corresponding to i, and $F(.)$ is the decoding function. A simple example for function $F(.)$ is exclusive OR (XOR) of all components (see Section 2.5 for details).

The pseudo-code of bridge node discovery in the CSD scheme is shown through Algorithms 1–3. In particular, Algorithm 1 prepares a broadcast message looking for bridge nodes. Algorithm 2 explains how each node receiving such a broadcast message should proceed. Algorithm 3 verifies the responses from the potential bridge nodes.

Algorithm 1 Pseudo-code of challenge message preparation

```
1:  for each $i \in \{S\} \cup \mathcal{N}_{in}$ do
2:      for each $j \in \{1, 2, \ldots, \lambda\}$ do
3:          Node $i$ generates a nonce $N_{i,j}$
4:          Node $i$ computes the challenge message $C_{i,j} = \{i||j||N_{i,j}\}_{K_{i,j}}$
5:          Node $i$ prepares message $M_{i,j} = \{i||C_{i,j}\}$
6:          Node $i$ sends all $M_{i,j}$ to node $S$
7:      end for
8:  end for
9:  Node $S$ prepares $\mathcal{MSG} = S||TTL|| \bigcup_{i \in \{S\} \cup \mathcal{N}_{in}, j \in \{1,2,\ldots,\lambda\}} M_{i,j}$
10: $S$ broadcasts $\mathcal{MSG}$
```

The pseudo-code of actual secret delivery in the CSD scheme is straightforward after the preparation of matrix M, as explained above. We omit the algorithm details due to page limit.

Note that it might be tempting to include key indices in message $M_{i,a}$ to improve processing speed at the potential bridge nodes. This would allow them to compare the key indices with their own instead of trying all keys for decryption. However, this information would also empower the adversaries to selectively compromise some of the nodes.

Algorithm 2 Pseudo-code for any node, e.g., node z, to process a received $\mathcal{MSG}$.

```
1:  Node $z$ decrypts $C_{S,j}$, $j \in \{1, 2, \ldots, \lambda\}$, using all of its keys.
2:  if Node $z$ has one key that decrypts $C_{S,j}$, $j \in \{1, 2, \ldots, \lambda\}$, successfully then
                                        ▷ Node $z$ shares a key with node $S$
3:      $n \leftarrow 0$
4:      for each $i \in \mathcal{N}_{in}$ do
5:          if Node $z$ has one key that decrypts $C_{i,j}$, $j \in \{1, 2, \ldots, \lambda\}$, successfully then
                                        ▷ Node $z$ shares a key with node $i$
6:              $n \leftarrow n + 1$
7:          end if
8:      end for
9:      if $n \geq \epsilon m$ then
                                        ▷ Node $z$ can serve as a bridge node
10:         for each shared key between $z$ and $i \in \{S\} \cup \mathcal{N}_{in}$ do
11:             Node $z$ decrypts $C_{i,a}$ in $M_{i,j}$ into $N_{i,j}$ using the shared key, $K_{i,j}$
12:             Node $z$ generates response $N'_{i,j}$ (e.g., $N'_{i,j} = N_{i,j} + 1$)
13:             Node $z$ computes $C'_{i,j} = \{z||K_{i,j}||N'_{i,j}\}_{K_{i,j}}$
14:             Node $z$ prepares $M'_{i,j} = \{z||C'_{i,j}\}$
15:         end for
16:         Node $z$ prepares $\mathcal{MSG}' = z|| \left\{ \bigcup_{\text{all } i \in \{S\} \cup \mathcal{N}_{in} \text{ sharing keys with } z} M'_{i,j} \right\}_{K_{S,j}}$
17:         Node $z$ sends $\mathcal{MSG}'$ to node S, using the list of attached IDs as routing information.
18:     end if
19: end if
20: TTL ← TTL-1
21: if $TTL > 0$ then
22:     $z$ adds its ID to the end of $\mathcal{MSG}$ and forwards $\mathcal{MSG}$
23: end if
```

2.4 *Discussions of the CSD scheme*

Encrypted Transmission: All message transmissions between nodes S and Z_j, $j \in \{1, 2, \cdots \tau\}$ are encrypted with the shared key between nodes S and Z_j, except the first broadcast message.[4] This will protect the transmitted information from being disclosed to a third party. Even though node Z_j can send all information to nodes B, E and G directly, transmission through node S is preferred. This has two benefits: reducing transmission overhead by putting all information into one large packet; eliminating the need to find routes between node Z_j and nodes B, E and G. Note that this may introduce extra energy consumption on the source node. However, since such a delivery does not occur frequently, the battery drain should not be a concern.

Algorithm 3 Pseudo-code to verify bridge nodes. Message $\mathcal{MSG}'$ is received at node S.

```
1: $S$ decrypts $\mathcal{MSG}'$ using all its keys to find $K_{S,j}$, the key shared between $S$ and $z$
2: $S$ decrypts $M'_{S,j}$ from node $z$ and checks whether response is valid.
                                        ▷ E.g., is $N'_{S,j} = N_{S,j} + 1$?
3: if response is valid then
4:     $S$ forwards $M'_{i,j}$ to the corresponding $i \in \mathcal{N}_{in}$
5:     for each $i \in \mathcal{N}_{in}$ that receives $M'_{i,j}$ do
6:         Node $i$ decrypts $C'_{i,j}$ in $M'_{i,j}$ using all its keys to find the shared key with node $z$.
7:         Node $i$ checks whether response is valid.
8:         if response is valid then
9:             Node $i$ sends confirmation to $S$
10:        end if
11:    end for
12:    if $S$ receives confirmation for each $M'_{i,j}$ in $\mathcal{MSG}$ then
13:        $S$ accepts $z$ as a bridge node
14:    else
15:        $S$ rejects $z$ as a bridge node
16:    end if
17: end if
```

Selection of ϵ and availability of bridge nodes: The selection of ϵ affects the performance of the CSD scheme. A large ϵ may find very few or zero number of bridge nodes; a small ϵ increases the workload of the source node and the transmission overhead due to extra packet headers. In an extreme, when $\epsilon m = 1$ and there is one insecure neighbour, the scheme is similar to the scheme in Li et al. (2005) with $k = 1$. When ϵ is small, bridge nodes can be found in the close neighbourhood of node S, as we have relaxed the requirement of asking a bridge node to share keys with node S and every node in $\mathcal{N}_{in}$. We will investigate the effect of ϵ on τ in Section 4.

Communication cost: Because of the less restrictive selection criteria as compared to the common bridge nodes in the Babel scheme (Deng and Han, 2007), it is more likely to identify bridge nodes that are close to the source and its insecure neighbours. Therefore, the communication cost of searching for bridge nodes and sending secret key information are low. Combining secret key components of different nodes into one message before sending them to a bridge node will further reduce communication overhead caused by packet headers and trailers. In addition, we argue that our proposed scheme will only be used when a source node needs to send secret to multiple insecure neighbours, which usually occur in a key initialisation process. Such an infrequent usage limits the extra transmission overhead caused by our scheme. Communication cost will be further investigated in Section 4.

Packet transmission failures: Wireless communication may experience transient transmission/reception failures. Therefore, the bridge node searching packets or replies may fail to reach their destination. In order to alleviate such failures, retransmissions of the bridge node search packets should be sent. The number of retransmissions and timing depends on the TTL value of each query and the chance of transmission failure. Note that such retransmissions will naturally increase transmission overhead and secret delivery delay.

Defense against eavesdroppers: The intention of the eavesdroppers is to obtain the secret key information. The CSD scheme makes eavesdropping rather difficult due to two mechanisms that it employs: secret components are delivered to bridge nodes through regular paths instead of any multi-hop secure paths, in which all routers get to know the contents that they forward; the receiver (insecure neighbour) XOR'es all the received key components into a single secret link key. Because of the XOR procedure, the eavesdroppers must compromise all the bridge nodes that are helping to deliver the key components to a particular insecure neighbour in order to re-construct the secret link key. Compromising any of the regular routers will not help (except that they would compromise more keys from the key space).

Defense against DoS attacks: DoS attackers can modify or interrupt secret component deliveries. Interestingly, the CSD scheme is able to identify and ignore any modified secret components. The procedure can be explained as follows: after receiving the secret components from the bridge nodes, an insecure neighbour performs a challenge-response check on each of the components with the source. This will make sure that none of the secret components has been modified. If any secret component is suspected to have been modified, the source and the insecure neighbour will simply ignore it in the XOR process, i.e., they will only XOR other key components but not this one. It is clear that this technique works unless the attackers compromised all the bridge nodes. This check is done by a coding scheme given in Section 2.5.

2.5 *Coding schemes*

There are different ways to prepare code matrix $M(j, i)$ and the combination function F in equation (1). We provide a technique as follows.

In designing the coding scheme, we need to consider several important factors: capability of resisting DoS attacks (message modification and/or message dropping), capability of resisting eavesdroppers and transmission overhead. The technique that we present below satisfies the first two requirements and it has a tunable transmission overhead. Due to the combined transmission of several elements in M, our secret delivery scheme is inherently efficient with respect to packet headers and underlying communication overhead.

In our scheme, all elements in M are randomly generated. The F function in equation (1) takes the form of XOR and has a unique form to screen and eliminate any modified elements received at node $i \in \mathcal{N}_{in}$. The screening process starts with XOR'ing all received elements received at node i and use challenge-and-response technique to ensure all elements are received correctly. This challenge-and-response takes place between node S and node i. When the response from node i fails the challenge from node S, node i understands that some elements must have been received incorrectly. A procedure to identify modified elements starts:

A straightforward method to identify the modified keys is to perform challenge-and-response for each key received by node i. Even though this method can identify all modified keys, it has a high communication overhead and long delay. In practice, there may be only a few modified keys. In this case, a better procedure based on coding theory can be performed. In the following, we demonstrate how to use a Hamming code to identify one modified key. When $m = 7$, there are only three checks to perform to identify the modified key instead of 7 checks for the straightforward method. When m is larger, the saving is more significant. For example, when $m = 15$, there are only four checks required instead of 15 to perform in the straightforward method. The test procedure for $m = 7$ is demonstrated as follows. Node i and node S both XOR the keys received from nodes Z_4, Z_5, Z_6, and Z_7. Then they check whether the resultant keys match. If they do, then set $s_1 = 0$; otherwise, set $s_1 = 1$. Both nodes perform the same procedure for keys received from Z_2, Z_3, Z_6 and Z_7, determining the value of s_2, and keys received from Z_1, Z_3, Z_5 and Z_7, determining the value of s_3. Then the value of s_1, s_2, s_3 is the binary representation of the index of common bridge that sent the modified key.

For example, if $s_1 = 1,\ \ s_2 = 1,\ \ s_3 = 0$, Z_6 sent the modified key to node i. Actually, the rule to determine the values of s_i, $1 \leq i \leq 3$ comes from the parity-check matrix H of the $(7, 4)$ Hamming code as follows:

$$H = \begin{bmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{bmatrix}.$$

In general, for $2^{r-1} - 1 < m \leq 2^r$, a parity-check matrix of the $(2^r - 1, 2^r - 1 - r)$ Hamming code (or its shorten version) is required. The ith column of the parity-check matrix H can be arranged as the trans-pose of the binary representation of i. For example, when $i = 6$, the sixth column of H is $(6)_2^T = (1\ 1\ 0)^T$. Arranged in this way, the decimal representation of $(s_1, s_2, \cdots, s_r)$ is the index of the erroneous key. The way to calculate s_i is to XOR the keys received from nodes which are corresponding to non-zero elements in ith row of H.

Once the modified elements are identified and eliminated, both node S and node i will XOR the rest of the elements transmitted and received successfully and use the result as the new secret shared between themselves. If more than one key has been modified, the aforementioned procedure will fail to identify them. Node i should use the straightforward procedure to identify such modified keys with node S.

Note that binary testing can also be used to identify a single compromised bridge and it has the same complexity as the above scheme. For the scenarios with more than one compromised bridges, more powerful codes can be used, e.g., multiple-error correcting Bose, Chaudhuri, and Hocquenghem (BCH) codes (Reed and Chen, 1999). Fast decoding algorithms are available for the receiver to identify compromised bridges.

The above scheme also helps node S to identify a potential compromised bridge. When node S performs challenge-and-response with the next insecure node, it can eliminate the keys forwarded by the potential compromised bridge. Hence, for each compromised bridge, at most one insecure node must perform the procedure to identify modified elements.

3 Analysis of bridge availability

We will estimate the average number of bridge nodes in order to deliver secret link keys to the insecure nodes of the source node S. Our analysis is based on the following system model: assume that there are N_p keys in the key pool from which each node chooses its λ distinctive keys. Two nodes sharing more than one common key will be able to set up a secure channel. Assume there are N nodes in the sensor network. Suppose that there are $m \geq 1$ insecure neighbours of node S, the source node. Based on the definition of insecure neighbours, none of these m nodes shares a common key with node S. N_g is the number of physical neighbours of node S, and $N_n(TTL)$ is the number of nodes that are within TTL-hop from node S.

For presentation purposes, we define the following events:

- $\Pi_{S,m}$: the event that node S has m insecure neighbours
- Ψ_0: the event that a node shares no key with node S
- $\Omega_{\lambda-b}$: the event that a node picks $\lambda - b$ keys from a key pool of size $N_p - \lambda$, while another node picks λ keys from the same key pool, and they do not share any key
- $\Xi_{m,c}$: the event that a non-neighbouring node of node S shares keys with node S and exactly c out of m insecure neighbours of node S
- Δ_b: the event that a non-neighbouring node shares exactly b keys with node S
- $\Xi^*_{m,c}$: the event that a secure neighbour of node S shares keys with exactly c out of m insecure neighbours of node S
- Δ^*_b: the event that a secure neighbour of node S shares exactly b keys with node S.

When event $\Pi_{S,m}$ occurs, we know that there are $N_g - m$ neighbouring nodes sharing at least a common key with node S. For each non-neighbour node of node S, we compute its chance of becoming a bridge node as the following

$$q(\epsilon) = \sum_{m=1}^{N_g} \sum_{c=\lceil m\epsilon \rceil}^{m} \Pr(\Xi_{m,c} | \Pi_{S,m}) \cdot \Pr(\Pi_{S,m}), \tag{2}$$

where ϵ is the portion of insecure neighbours with which the potential bridge node must share keys before it can be a bridge and $\lceil \cdot \rceil$ is the ceiling function.

Similarly, there are $N_g - m$ secure neighbours of node S. We compute the chance that each of these becomes a bridge node as

$$q^*(\epsilon) = \sum_{m=1}^{N_g} \sum_{c=\lceil m\epsilon \rceil}^{m} \Pr(\Xi^*_{m,c} | \Pi_{S,m}) \cdot \Pr(\Pi_{S,m}), \tag{3}$$

where some of the terms have been revised to include a star, representing the secure neighbours.

We first focus on the non-neighbours. Obviously, the probability depends on the value of ϵ, or $\epsilon' = \lceil m\epsilon \rceil$. For each $c = \lceil m\epsilon \rceil, \cdots m$, we compute the chance that this node shares keys with exactly c of the insecure neighbours of node S as well as node S itself. Such a probability can be calculated as the conditional probability of different m.

And since there are $N_n(TTL)$ nodes within the TTL-hop region, N_g of which are neighbours of node S, the expected number of bridge nodes from non-neighbours is

$$\begin{aligned} N_1(\epsilon) &= (N_n(TTL) - N_g)q(\epsilon) \\ &= (N_n(TTL) - N_g) \cdot \\ &\quad \sum_{m=1}^{N_g} \sum_{c=\lceil m\epsilon \rceil}^{m} \Pr(\Xi_{m,c}|\Pi_{S,m}) \cdot \Pr(\Pi_{S,m}). \end{aligned} \tag{4}$$

Since there are $N_g - m$ secure neighbours (with probability $\Pr(\Pi_{S,m})$), the expected number of bridge nodes from secure neighbours is

$$\begin{aligned} N_1^*(\epsilon) = \sum_{m=1}^{N_g} &(N_g - m) \cdot \\ &\sum_{c=\lceil m\epsilon \rceil}^{m} \Pr(\Xi^*_{m,c}|\Pi_{S,m}) \cdot \Pr(\Pi_{S,m}) \end{aligned} \tag{5}$$

and the overall expected number of bridge nodes is

$$N_e(\epsilon) = N_1(\epsilon) + N_1^*(\epsilon). \tag{6}$$

Now we compute the two probabilities in equation (4).

First, we shall calculate $\Pr(\Pi_{S,m})$. Recall that Ψ_0 is the event that there is no shared key between node S and another node:

$$\Pr(\Psi_0) = \frac{\binom{N_p-\lambda}{\lambda}}{\binom{N_p}{\lambda}}. \tag{7}$$

Then we have

$$\Pr(\Pi_{S,m}) = \binom{N_g}{m} [\Pr(\Psi_0)]^m [1 - \Pr(\Psi_0)]^{N_g - m}. \tag{8}$$

The probability that a non-neighbouring node shares keys with exactly c insecure neighbours of node S and node S itself given that node S has m insecure neighbours can be computed as

$$\begin{aligned} \Pr(\Xi_{m,c}|\Pi_{S,m}) = \sum_{b=1}^{\lambda-1} &\Pr(\Xi_{m,c}|\Delta_b, \Pi_{S,m}) \cdot \\ &\Pr(\Delta_b|\Pi_{S,m}). \end{aligned} \tag{9}$$

The event of $\Xi_{m,c}$ given Δ_b and $\Pi_{S,m}$ means that the non-neighbouring node, e.g. node z, shares keys with exactly c insecure neighbours of node S, given that node S has m insecure neighbours and node S shares exactly b keys with node z. Then it can be estimated as

$$\begin{aligned} \Pr(\Xi_{m,c}|\Delta_b, \Pi_{S,m}) \approx \binom{m}{c} &[\Pr(\Omega_{\lambda-b})]^{m-c} \cdot \\ &[1 - \Pr(\Omega_{\lambda-b})]^c, \end{aligned} \tag{10}$$

where $\Pr(\Omega_{\lambda-b})$ represents the probability of event $\Omega_{\lambda-b}$ taking place, i.e., node z picks $\lambda - b$ keys from a key pool of size $N_p - \lambda$, one of the insecure neighbours picks λ keys from the same key pool, and they do not share any key. We have

$$\Pr(\Omega_{\lambda-b}) = \frac{\binom{N_p-\lambda-(\lambda-b)}{\lambda}}{\binom{N_p-\lambda}{\lambda}}. \tag{11}$$

The events Δ_b and $\Pi_{S,m}$ are mutually independent since it concerns different nodes (node z and the physical neighbours of node S). Therefore, $\Pr(\Delta_b|\Pi_{S,m})$ can be computed as follows:

$$\Pr(\Delta_b|\Pi_{S,m}) = \Pr(\Delta_b) = \frac{\binom{\lambda}{b}\binom{N_p-\lambda}{\lambda-b}}{\binom{N_p}{\lambda}}. \tag{12}$$

In order to calculate equation (3), we use a similar approach, except that some of the terms should be replaced with the those with stars.

The probability that a secure neighbour shares keys with exactly c insecure neighbours of node S given that node S has m insecure neighbours can be computed as

$$\begin{aligned} \Pr(\Xi^*_{m,c}|\Pi_{S,m}) = \sum_{b=1}^{\lambda-1} &\Pr(\Xi^*_{m,c}|\Delta^*_b, \Pi_{S,m}) \cdot \\ &\Pr(\Delta^*_b|\Pi_{S,m}). \end{aligned} \tag{13}$$

The event of $\Xi^*_{m,c}$ given Δ^*_b and $\Pi_{S,m}$ means that a secure neighbour shares keys with exactly c insecure neighbours of node S, given that node S has m insecure neighbours and node S shares exactly b keys with this secure neighbour. Then we have the estimation as

$$\begin{aligned} \Pr(\Xi^*_{m,c}|\Delta^*_b, \Pi_{S,m}) \approx \binom{m}{c} &[\Pr(\Omega_{\lambda-b})]^{m-c} \cdot \\ &[1 - \Pr(\Omega_{\lambda-b})]^c, \end{aligned} \tag{14}$$

where $\Pr(\Omega_{\lambda-b})$ is given by equation (11).

The event Δ^*_b given $\Pi_{S,m}$ means that, given that node S has m insecure neighbours, a secure neighbour shares exactly b keys with node S. We need to take care of the conditional probability where this secure neighbour shares at least one key with the source. Therefore, $\Pr(\Delta^*_b|\Pi_{S,m})$ can be computed as follows:

$$\Pr(\Delta^*_b|\Pi_{S,m}) = \frac{\Pr(\Delta_b)}{1 - \Pr(\Delta_0)} = \frac{\binom{\lambda}{b}\binom{N_p-\lambda}{\lambda-b}}{\binom{N_p}{\lambda} - \binom{N_p-\lambda}{\lambda}}. \tag{15}$$

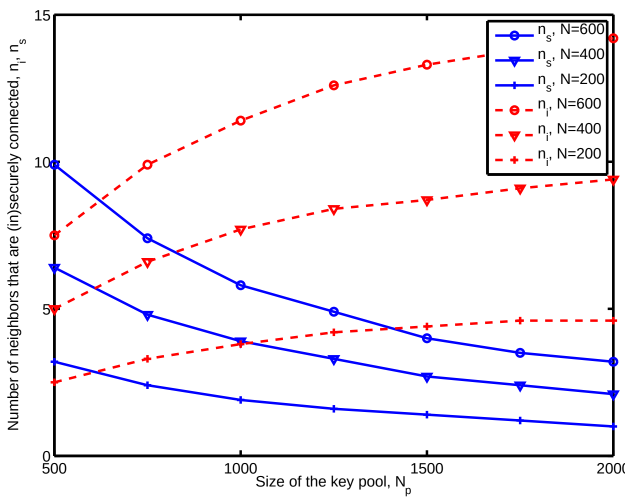
4 Performance evaluation

We used MATLAB in our simulations, in which N sensors are randomly deployed to an area of 1000 by 1000 m^2, i.e., each node's X and Y locations are chosen uniformly between 0 and 1000. Wireless transmission range is 100 m. Each node carries λ keys from a large key pool of size N_p. We chose one source randomly and tried to establish a secret link key to each of its security-disconnected neighbours. All results are the average of 20 random runs. In our simulations, we used a simplified circular connectivity model and an abstract of the physical and networking models. Furthermore, we focused on key sharing among different nodes. Therefore, simulations through NS2, OPNET or other more complex simulators are unnecessary at this stage.

4.1 Necessity and availability of bridge nodes

First of all, we demonstrate the number of secure/insecure neighbours from a source for different sizes of key pool, N_p, and node density, N, in Figure 2. As N_p increases, it is less likely to find secure neighbours and more neighbours of the source become insecure neighbours (decreasing n_s and increasing n_i). Based on the curves in Figure 2, n_i and n_s change proportionally with N. The increased number of insecure neighbours of the source require more bridge nodes to deliver the secret link keys.

Figure 2 Comparison of the number of secure/insecure neighbours of a source, n_s, n_i. $\lambda = 20$ (see online version for colours)

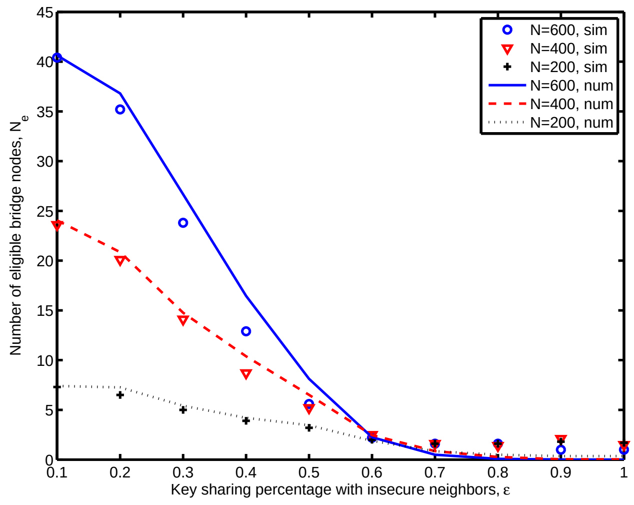


Next, we study the availability of bridge nodes, which can be used to forward the secret information from the source to the insecure neighbours. These nodes must share a key with the source and ϵ portion of the insecure neighbours of the source. We demonstrate the effects of ϵ on the number of eligible bridge nodes, N_e, in Figure 3.

Intuitively, N_e should decrease as ϵ increases because the requirement is more restrictive. This is confirmed in Figure 3, which also shows that the number of eligible bridge nodes increases as there are more nodes in the network (increasing N). For example, when $\epsilon = 0.4$, there are $N_e = 10, 7$, and 4 eligible bridge nodes with $N = 600, 400$ and 200 nodes in the network. Numerical results from analysis have been plotted in Figure 3 and matched well with simulation results.

In order to compare the number of eligible bridge nodes within different neighbourhood sizes, we present TTL's effect on N_e in Figure 4. As can be observed from Figure 4, N_e increases with TTL because more nodes are considered. For instance, when TTL is extended from 2 to 4, the number of eligible bridge nodes almost triples. Note that the total number of nodes within TTL-hops should roughly quadruple if TTL doubles because of the 2D region.

Figure 3 Number of eligible bridge nodes, N_e. ϵ represents the portion of insecure neighbours that must share keys with before they become eligible. These nodes must share a key with the source node as well. $N_p = 1000$, $\lambda = 20$ and $TTL = 3$ in these simulations, i.e., only the nodes within three-hops from the source were considered. Numerical results from analysis are plotted, labelled as 'num', and matched well with simulation results (see online version for colours)



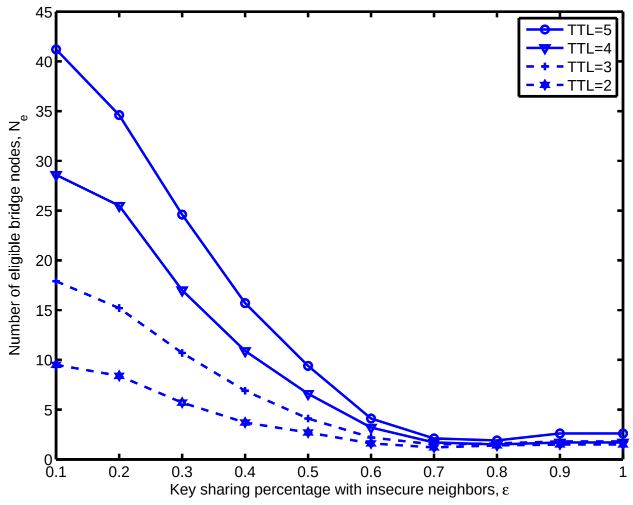
4.2 Transmission overhead

Since the transmission overhead of our scheme depends largely on how far the bridge nodes are away from the source node and the insecure neighbours, we recorded the total hop count from the source towards all the insecure neighbours and showed them in Figure 5. Based on Figure 5, the transmission cost is higher when we extend the region (increasing TTL) to search for bridge nodes and lower the key sharing requirement (decreasing ϵ). When $\epsilon > 0.7$, the difference in transmission costs for different ϵ is small.

4.3 Security evaluation

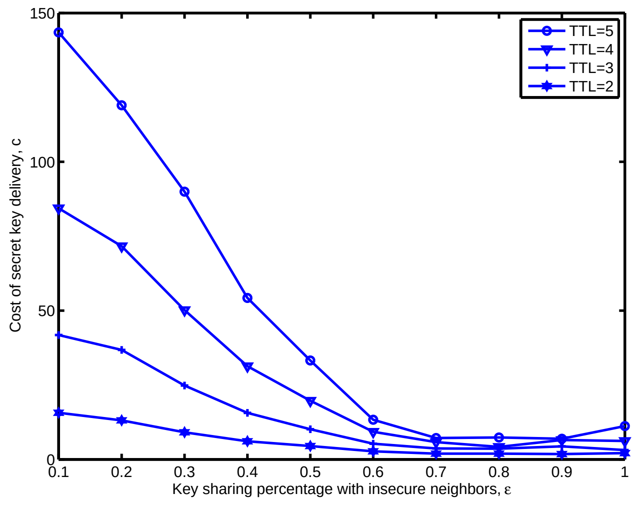
We present the secret disclosure probability of the CSD scheme under the condition of node compromise. In our simulations, we randomly picked x_c portion of the nodes as compromised nodes (except the source and the insecure neighbours). Once a node is compromised, all information on it is disclosed. Then we computed the probability of the secret delivery being compromised (or secret disclosed to any third party). *Secret disclosure probability* is defined as the chance of a secret link key of an insecure neighbour is disclosed to the adversary.

Figure 4 Number of eligible bridge nodes, N_e. ϵ represents the portion of insecure neighbours must share keys with before they become eligible. These nodes must share a key with the source node as well. $N_p = 1000$, $\lambda = 20$, and $N = 400$ (see online version for colours)



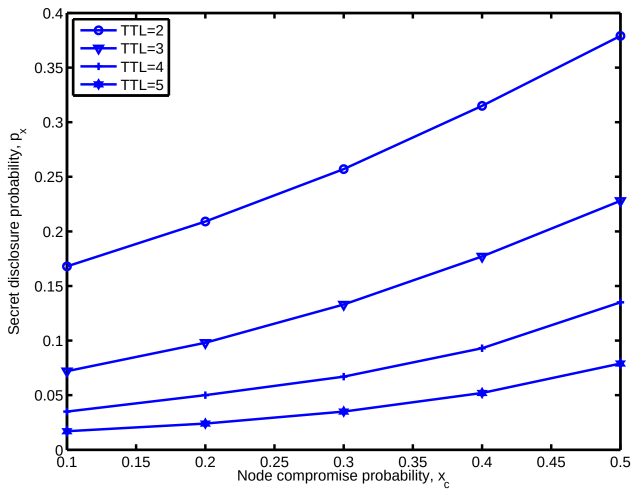
In Figure 6, the secret disclosure probability of the CSD scheme with different TTL values is shown and compared. As node compromise probability, x_c, increases, secret disclosure probability is higher. Since each insecure neighbour XOR'es all received components that are sent through the bridge nodes, the adversary needs to compromise all the bridge nodes helping to deliver keys towards this node in order to compromise the secret delivery. As a result, when there are more bridge nodes in use (due to an increased TTL), the chance of secret disclosure is lower.

Figure 5 Transmission cost of the CSD scheme in terms of number of hops. We calculate the total number of hops for all the bridge nodes and from the source to the insecure neighbours. $N = 400$, $\lambda = 20$, $N_p = 1000$ (see online version for colours)



We compared the secret disclosure probability of CSD and several related schemes in Figure 7. In CSD, the adversary needs to compromise all the bridge nodes that help forwarding key components. In SMSP scheme (Eschenauer and Gligor, 2002), compromising any node on the path will disclose the secret. In MMSP scheme (Chan et al., 2003), the adversary must compromise at least one node on each of the used paths. In the Babel scheme (Deng and Han, 2007) though, only one bridge node is used; therefore, the secret disclosure probability is simply x_c. In SPREAD (Liu et al., 2009), the adversary needs to compromise at least one node on each of all of the node-disjoint paths in use.

Figure 6 Secret disclosure probability for CSD with different TTL values. $N_p = 1000$, $\lambda = 20$, $N = 400$, $\epsilon = 0.5$ (see online version for colours)

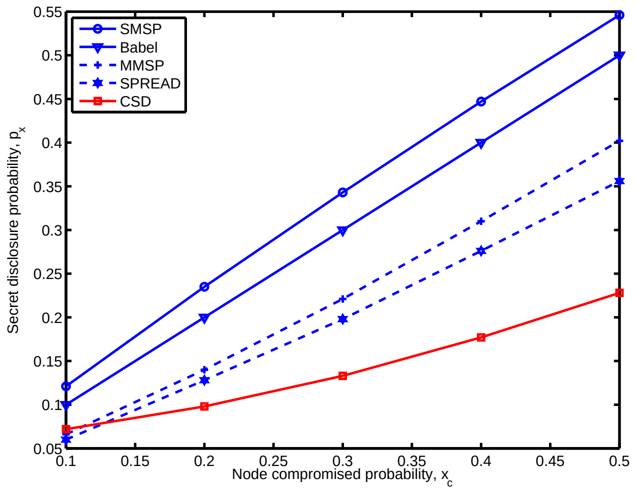


As can be observed from Figure 7, the CSD scheme enjoys a lower secret disclosure probability than SMSP, MMSP, the Babel scheme and the SPREAD scheme, except in a rather small compromised probability scenario. MMSP has a secret disclosure probability that is lower than Babel because the adversary needs to compromise multiple nodes in order to obtain the secret being delivered through multiple secured paths. The SPREAD scheme is better than MMSP with its node-disjoint secure paths. The better security of the CSD scheme comes from its diversified transmission through multiple bridge nodes. Furthermore, the routers in between do not get to know any secret that they help forwarding.

5 Related work

Besides the related work discussed above (Eschenauer and Gligor, 2002; Du et al., 2005; Liu and Ning, 2003), Chan et al. presented a technique to establish secure link keys for two neighbouring nodes if they do not share enough common keys (Chan et al., 2003). These two neighbours first identify all secure paths between themselves. Then one node generates a set of random keys (of the same size) for all the paths and send one key to the destination through each of the paths. After the destination receives all the keys, it XOR'es all of them to obtain the secret link key. The multi-path key reinforcement scheme significantly improves the protection of the secret link key from being disclosed to the adversary.

Figure 7 Secret disclosure probability for CSD and several related schemes. $N = 400$, $N_p = 1000$, $\lambda = 20$, $\epsilon = 0.5$, $TTL = 3$. We compare CSD with SMSP: single multi-hop secure path scheme (Eschenauer and Gligor, 2002); Babel (Deng and Han, 2007); MMSP: multiple multi-hop secure path scheme (Chan et al., 2003); and SPREAD (Liu et al., 2009) (see online version for colours)



Deterministic key pre-distribution schemes, location-assisted key pre-distribution schemes and group-based key pre-distribution schemes can help to improve or even assure local connectivity (Lee et al., 2005; Du et al., 2006). However, these schemes require either additional memory space or group/location information. Instead, our scheme complements the random key pre-distribution scheme after sensor deployment and helps to increase local security connectivity.

Li et al. proposed to use k intermediate nodes between two sensors to establish a link key (Li et al., 2005). Two methods were presented to identify such k nodes, which should be securely connected to both nodes. In one method, all common neighbours of these two nodes are checked and one neighbour securely connected to both the source and the destination is used; in the other method, the source node sends out a controlled flooding message to recruit k nodes, which may not be physical neighbours of these two nodes but each of them should still be securely connected to the source and the destination. In our paper, we focus on a different problem: efficient and secure secret delivery from one node to multiple neighbours. By coordinating the tasks of secret delivery towards several insecure neighbours and identifying best bridge nodes, we increase the chance of finding bridge nodes, which are only required to share keys with some of such insecure neighbours. Furthermore, with the use of Hamming code, our scheme protects the delivered secret from eavesdropping and secret modification by the compromised nodes in the network. The receiver can detect and correct such errors given that the number of errors is lower than a certain threshold.

In order to combat the problem of topology instability in wireless networks, a multi-path routing scheme was proposed and investigated in Tsirigos and Haas (2001). The scheme allows the sender to add extra overhead to each packet that is to be transmitted over multiple paths. The goal is to find the optimal way to fragment the packet into smaller blocks and deliver them over multiple paths. A secure routing protocol was proposed in Papadimitratos and Haas (2003) to send additional information to protect routing information from being dropped. Secure message transmission and secure single path were proposed to send message over unreliable mobile ad hoc networks (Papadimitratos and Haas, 2006).

Multipath delivery (Yilmaz et al., 2012) has been used by many works in the technical literature. Lou et al. used multiple paths to deliver multiple shares of a secret in their SPREAD scheme (Liu et al., 2009). Collision-aware multipath routing was proposed in Wang et al. (2009). Shu et al. took advantage of randomised dispersive routes to circumvent black holes formed by compromised nodes and DoS attackers (Shu et al., 2010). Xiong et al. investigated the energy consumption issue of single and multiple path delivery techniques (Xiong et al., 2010). A secure and reliable multipath routing protocol was proposed in Moustafa et al. (2011). Zhang et al. proposed a segment-based multi-path routing protocol to divide a primary route into multiple segments led by reliable nodes (Zhang et al., 2011). Chen et al. balanced multipaths in the proximity of the source/sink for multimedia traffic delivery (Chen et al., 2012). Lee et al. proposed a radio-disjoint geographic multipath scheme to avoid the collisions (Lee et al., 2012). Fault-tolerant and reliable multipath routing were investigated in Alwan and Agarwal (2010) and Challal et al. (2011).

Maximum distance separable codes have been used to establish link keys between neighbours without common keys through multiple paths (Huang and Medhi, 2005; Deng and Han, 2008). Different from these works, we focus on the delivery of multiple link keys from one source to multiple immediate physical neighbours.

Traynor et al. proposed to use a few more powerful sensors to achieve key establishment (Traynor et al., 2006). Since these sensors are expected to have tamper-resistant hardware, the keys on these sensors are considered safe. Miller and Vaidya proposed to use Bloom filter and Merkel trees to distribute key information, so that the chance of neighbouring nodes establishing a key can be close to 1 (Miller and Vaidya, 2006). Their scheme takes advantage of the multiple available channel and performs random switching among these channels in the initialisation process. Mobile keying robots were used in Tas and Tosun (2011) and triple key distribution using trades was presented in Ruj et al. (2012).

In Gupta et al. (2006), a new pairwise key establishment technique was introduced. The technique uses ‘friends’, which share keys with the receiver, in addition to ‘proxies’, each of which shares a key with the sender and the receiver. Each of the ‘friend’ nodes sends a part-key back to the sender, which will choose some of such part-keys and use a public function to generate a pairwise key between itself and the receiver. Since the ‘friend’ nodes only need to share key with the receiver, the number of such friend nodes is large and they are within shorter hops from the source node. The proxies will also send part keys but with encryption, since they share keys with the source. Because the friends do not share keys with the source node, the transmission of their part keys is open for eavesdropping and they have limited contribution towards the secrecy strength of the generated key.

A hop-by-hop authentication scheme was used to protect path key establishment from stop-forwarding and Byzantine attacks (Sheu and Cheng, 2007). It can further identify malicious nodes as well. With the use of hop-by-hop authentication, false data can be quickly detected, keeping attacked nodes from forwarding such fabricated packets. After finding n node-disjoint paths between the source and the destination, the secret path key is divided and transmitted through these paths. The division of the path key makes sure that, as long as $k < n$ out of n paths deliver the their partial keys, the destination will be able to recover the secret path key. On the contrary, we focus on the delivery of secret information (or keys) from a source node to several of the disconnected neighbours.

The Babel scheme (Deng and Han, 2007) searches for a common bridge node that shares key with the source and the neighbours. The common bridge node will be used to deliver secrets for all such insecure one-hop links. The main problem of such an approach is that the common bridge node may become an easy DoS target. Instead, our approach in this work is to find several bridge nodes to help with the delivery, and these bridge nodes only need to share keys with some of the neighbours, but not all of them.

6 Conclusions

We have proposed the CSD scheme that makes use of multiple bridge nodes within the close neighbourhood of the source and insecure neighbours. These bridge nodes cooperatively help to deliver secret link key information from the source towards the insecure neighbours. In the delivery process, no secret information is disclosed to any node en route. These routers simply forward the encrypted data. Even the bridge nodes only get to know partial information about the secrets that they help to deliver.

We have performed extensive theoretical analysis and performance evaluation on our proposed CSD scheme. It is shown that the CSD scheme enjoys a relatively low communication cost and low secret disclosure probability. The CSD scheme combats the eavesdropping attack with the extra protection strategies. The unique locations of the source and the insecure neighbours allow easy challenge-response integrity check.

Acknowledgements

This work was supported in part by UNCG new faculty grant and by the National Science Council of Taiwan, ROC, under grants NSC 99-2221-E-011-158-MY3 and NSC 96-2221-E-305-002-MY3. Y.S. Han's work was partially supported by his visit to LIVE lab at University of Texas at Austin.

References

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'A survey on sensor networks', *IEEE Communications Magazine*, pp.102–114.

Alwan, H. and Agarwal, A. (2010) 'Reliable fault- tolerant multipath routing protocol for wireless sensor networks', *Communications (QBSC), 2010 25th Biennial Symposium on*, pp.323–326.

Challal, Y., Ouadjaout, A., Lasla, N., Bagaa, M. and Hadjidj, A. (2011) 'Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks', *Journal of Network and Computer Applications*, Advanced Topics in Cloud Computing, Vol. 34, No. 4, pp.1380–1397.

Chan, H., Perrig, A. and Song, D. (2003) 'Random key predistribution schemes for sensor networks', *Proc. of IEEE Symposium on Security and Privacy*, Berkeley, California, pp.197–213,

Chen, M., Leung, V.C., Shu, L. and Chao, H-C. (2012) 'On multipath balancing and expanding for wireless multimedia sensor networks', *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 9, No. 2, pp.95–103.

Deng, J. and Han, Y.S. (2007) 'Babel: using a common bridge node to deliver multiple keys in wireless sensor networks', *Proc. of IEEE Global Telecommunications Conference - General Symposium (GLOBECOM '07)*, Washington, DC, USA, pp.161–165.

Deng, J. and Han, Y.S. (2008) 'Multi-path key establishment for wireless sensor networks using just enough redundancy transmission', *IEEE Transactions on Dependable and Secure Computing*, Vol. 5, No. 3, pp.177–190.

Du, W., Deng, J., Han, Y.S. and Varshney, P.K. (2006) 'A key predistribution scheme for sensor networks using deployment knowledge', *IEEE Transactions on Dependable and Secure Computing*, Vol. 3, No. 1, pp.62–77.

Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J. and Khalili, A. (2005) 'A pairwise key pre-distribution scheme for wireless sensor networks', *ACM Transactions on Information and System Security*, Vol. 8, No. 2, pp.228–258.

Eschenauer, L. and Gligor, V.D. (2002) 'A key-management scheme for distributed sensor networks', *Proc. of the 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, pp.41–47.

Fu, H., Kawamura, S., Zhang, M. and Zhang, L. (2008) 'Replication attack on random key pre-distribution schemes for wireless sensor networks', *Computer Communications*, Vol. 31, No. 4, pp.842–857.

Gupta, A., Kuri, J. and Nuggehalli, P. (2006) 'A new scheme for establishing pairwise keys for wireless sensor networks', *Distributed Computing and Networking*, Vol. 4308 of Lecture Notes in Computer Science (LNCS), Springer, pp.522–533.

Huang, D., Mehta, M., van de Liefvoort, A. and Medhi, D. (2007) 'Modeling pairwise key establishment for random key predistribution in large-scale sensor networks', *IEEE/ACM Trans. on Networking*, Vol. 15, No. 5, pp.1204–1215.

Huang, D. and Medhi, D. (2005) 'A byzantine resilient multi-path key establishment scheme and its robustness analysis for sensor networks', *Proc. of 19th IEEE International Parallel and Distributed Processing Symposium*, Colorado, USA, p.240b.

Huang, D. and Medhi, D. (2007) 'Secure pairwise key establishment in large-scale sensor networks: An area partitioning and multigroup key predistribution approach', *ACM Trans. on Sensor Networks*, August, Vol. 3, No. 3, Article No. 16.

Lee, J., Park, H., Oh, S., Yim, Y. and Kim, S-H. (2012) 'A radio-disjoint geographic multipath routing in wireless sensor networks', *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pp.803–809.

Lee, J. and Stinson, D.R. (2005) 'A combinatorial approach to key predistribution for distributed sensor networks', *Proc. of IEEE Wireless Communications and Networking Conference (WCNC '05)*, New Orleans, LA, USA, pp.1200–1205

Li, G., Ling, H. and Znati, T. (2005) 'Path key establishment using multiple secured paths in wireless sensor networks', *Proc. of CoNEXT '05*, Toulouse, France, pp.43–49.

Li, W-S., Tsai, C-W., Chen, M., Hsieh, W-S. and Yang, C-S. (2012) 'Threshold behavior of multi- path random key pre-distribution for sparse wireless sensor networks', *Mathematical and Computer Modelling*, No. 0, Available online March 2012.

Liu, D. and Ning, P. (2003) 'Establishing pairwise keys in distributed sensor networks', *Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, Washington, DC, USA, pp.52–61.

Liu, Z., Ma, J., Huang, Q. and Moon, S. (2009) 'Asymmetric key pre-distribution scheme for sensor networks', *IEEE Trans. on Wireless Communications*, Vol. 8, No. 3, pp.1366–1372.

Lou, W., Liu, W., Zhang, Y. and Fang, Y. (2009) 'Spread: improving network security by multipath routing in mobile ad hoc networks', *Wirel. Netw.*, Vol. 15, No. 3, pp.279–294.

Miller, M.J. and Vaidya, N.H. (2006) 'Leveraging channel diversity for key establishment in wireless sensor networks', *Proc. of the 25th Conference of the IEEE Communications Society (Infocom '06)*, Barcelona, Spain, pp.1–12

Mishra, A., Gyires, T. and Tang, Y. (2012) 'Towards a theoretically bounded path key establishment mechanism in wireless sensor networks', *Proc. of the Eleventh International Conference on Networks (ICN '12)*, Saint Gilles, Reunion Island, pp.147–152.

Mittal, N. and Novales, R. (2010) 'Cluster-based key predistribution using deployment knowledge', *IEEE Trans. on Dependable and Secure Computing*, Vol. 7, No. 3, pp.329–335.

Moustafa, M., Youssef, M. and El-Derini, M. (2011) 'Msr: A multipath secure reliable routing protocol for wsns', *Computer Systems and Applications (AICCSA), 2011 9th IEEE/ACS International Conference on*, pp.54–59.

Papadimitratos, P. and Haas, Z.J. (2003) 'Secure message transmission in mobile ad hoc networks', *Elsevier Ad Hoc Networks*, Vol. 1, No. 1, pp.193–209.

Papadimitratos, P. and Haas, Z.J. (2006) 'Secure data transmission in mobile ad hoc networks', *IEEE Journal on Selected Areas in Communications*, Vol. 24, No. 2, pp.343–356.

Reed, I.S. and Chen, X. (1999) *Error-Control Coding for Data Networks*, Kluwer Academic, Boston, MA.

Ruj, S., Nayak, A. and Stojmenovic, I. (2012) 'Pairwise and triple key distribution in wireless sensor networks with applications', *IEEE Transactions on Computers*, Vol. 99, No. 1, pp.1–1.

Sheu, J-P. and Cheng, J-C. (2007) 'Pair-wise path key establishment in wireless sensor networks', *Computer Communications*, Special Issue on Security on Wireless Ad Hoc and Sensor Networks, Vol. 30, Nos. 11–12, pp.2365–2374.

Shu, T., Krunz, M. and Liu, S. (2010) 'Secure data collection in wireless sensor networks using randomized dispersive routes', *Mobile Computing, IEEE Transactions on*, Vol. 9, No. 7, pp.941–954.

Tas, B. and Tosun, A.S. (2011) 'Mobile assisted key distribution in wireless sensor networks', *Proc. of IEEE International Conference on Communications (ICC '11)*, Kyoto, Japan, pp.1–6.

Traynor, P., Choi, H., Cao, G., Zhu, S. and LaPorta, T. (2006) 'Establishing pair-wise keys in heterogeneous sensor networks', *Proc. of the 25th Conference of the IEEE Communications Society (Infocom '06)*, Barcelona, Spain, pp.1–10

Tsirigos, A. and Haas, Z.J. (2001) 'Multipath routing in the presence of frequent topological changes', *IEEE Communication Magazine*, Vol. 39, No. 11, pp.132–138.

Wang, Z., Bulut, E. and Szymanski, B. (2009) 'Energy efficient collision aware multipath routing for wireless sensor networks', *Communications, 2009. ICC '09. IEEE International Conference on*, Dresden, Germany, pp.1–5.

Xiong, B., Lin, C., Ren, F. and Yan, W. (2010) 'Single path or multipath stochastic reliability in wireless sensor networks', *Communication Technology (ICCT), 2010 12th IEEE International Conference on*, Nanjing, P.R. China, pp.243–246.

Yilmaz, O., Demirci, S., Kaymak, Y., Ergun, S. and Yildirim, A. (2012) 'Shortest hop multipath algorithm for wireless sensor networks', *Computers & Mathematics with Applications*, Vol. 63, No. 1, pp.48–59.

Zhang, Z., Wang, Y. and Li, F. (2011) 'Reliable packets delivery over segment-based multi-path in wireless ad hoc networks', *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, Lanzhou, P.R. China, pp.300–306.

Notes

[1] Such neighbours are termed to-be-connected neighbours in Deng and Han (2007).

[2] We use the challenge-response technique to verify the common keys between the common bridge node and the source and the insecure neighbours. With the use of this technique, key and key index will not be disclosed. If the adversary is not expected to be capable of identifying keys from indices, key indices can be sent directly, greatly reducing the communication and computational cost of our scheme.

[3] While sending such encrypted shared secret components directly back to the nodes in $\mathcal{N}_{in}$ may be more efficient, the cost of identifying such new paths can eliminate any of such benefits. Therefore, we design our scheme to allow these nodes to send encrypted shared secret components back to node S, which will forward them to the nodes in $\mathcal{N}_{in}$.

[4] In fact, encryption of the first broadcast message is possible. Node S would just encrypt the same message with λ copies, each with one of its λ carried keys. Then only a node sharing at least a key with node S can decrypt the message. The computational and transmission cost would be higher though, since node S needs to encrypt the message λ times and broadcast a message that is λ times as large in size.