

A Byzantine Resilient Multi-path Key Establishment Scheme and Its Robustness Analysis for Sensor Networks

Dijiang Huang, Deep Medhi
Computer Science & Electrical Engineering Department
University of Missouri – Kansas City
Kansas City, Missouri 64110, USA.
{dhuang, dmedhi}@umkc.edu

Abstract

Sensor networks are composed of a large number of low power sensor devices. For secure communication among sensors, secret keys must be established between them. Random key predistribution and pairwise key establishment schemes have been proposed for key management in large-scale sensor networks. In these schemes, after being deployed, sensors set up pairwise keys via preinstalled keys. The key establishment schemes are vulnerable to Byzantine attacks, i.e., packet dropping or altering. To counter these attacks, we propose a Byzantine resilient multi-path key establishment scheme that uses the Reed-Solomon error-correct coding scheme to improve resilience to Byzantine attacks. Our proposed scheme can tolerate at most t faulty key paths, where $t = (n - k)/2$ when (n, k) Reed-Solomon error-correct coding scheme is used. In addition, by using the Reed-solomon coding scheme, sensors can identify the faulty paths with minimal communication overhead.

1. Introduction

Sensor networks are composed of a large number of low power sensor devices. For secure communication among sensors, secret keys must be established between them. Recently, several pairwise key schemes have been proposed for large-scale sensor networks. These schemes randomly select a set of keys from a key pool and install the keys in the memory of each sensor. After deployment, the sensors can set up pairwise keys by using the preinstalled keys.

In order to improve the resilience to security attacks on key management schemes for a large-scale sensor network, several pairwise key management schemes have been proposed. Based on the formation of the key pool, we broadly classify the proposed schemes into two groups: Purely Random Key Predistribution (P-RKP) schemes and Structured

Key-pool Random Key Predistribution Scheme (SK-RKP) schemes.

The P-RKP scheme was first proposed by Eschenauer and Gligor [5], and we call it the *basic scheme*. The proposals that follow the *basic scheme* improve the *basic scheme* in five aspects: 1) shared keys threshold: q -composite scheme [2] to improve the resilience against node capture attack, in which attackers can capture sensors and derive the preinstalled key information. The compromised keys can be also used among uncompromised sensors [2].; 2) key pool structure: SK-RKP scheme [9, 4] to improve the resilience to node capture attack; 3) location awareness: key predistribution and sensors' deployment are based on known sensors' deployment information [3, 10, 7] (also target at improving the resilience to node capture attack); 4) shared keys discovery [12, 11]: the one-way function schemes have been proposed to reduce the communication overhead during the key discovery phase and improve the resilience to node fabrication attack, in which attackers can fabricate new nodes based on information derived from the captured sensors [7]; 5) path-key establishment protocol: multi-path key establishment [2, 19] to prevent a few compromised sensors from knowing the established pairwise keys.

Rabin [13] proposed using dispersal of information for security and fault tolerance; Tsirigos and Haas [18] proposed using multi-path coding schemes to solved the path failure, such as packet dropping. However, when *Byzantine* node modify the data, the proposed schemes fails to recover the original message. Furthermore, the diversity coding cannot detect faulty path.

We can see from the above discussion that significant progress has been made in regard to guarding various attacks, yet *Byzantine* attacks (discussed below) on the path-key establishment protocol are the hardest ones to guard against. All of the previously discussed schemes are vulnerable to *Byzantine attack*, in which a faulty sensor (captured

by attackers) can (a) reveal the *indirect keys* set up via the key establishment protocol (We use *direct key* to represent a preinstalled key and *indirect key* to represent a pairwise key established via *direct keys*); (b) stop forwarding the *indirect keys* to prevent the sensors from establishing the *indirect keys*; (c) or cheat the receivers by altering the forwarded keys. The cheating attack can prevent the pair of key establishment nodes from deriving the same key. Before we go further, we first describe Byzantine attack.

1.1. Byzantine Attack

The *Byzantine* problem, also referred to as *Byzantine Generals* problem [8], can be expressed abstractly in terms of a group of generals of the *Byzantine* army camped with their troops around an enemy city. Communicating only by messengers, generals must agree upon a common battle plan. However, messengers may be captured by enemy and they may deliver wrong messages among generals. The problem is to find an algorithm to ensure that the loyal generals will reach the right message.

Here, we consider any pair of nodes that want to set up *indirect keys* as *Byzantine Generals* and the intermediate nodes that help set up *indirect keys* as messengers. The *Byzantine* attack is defined as the messengers sniff, drop, or alter transmitted messages between *Byzantine Generals*.

In *sniffing* attacks, a malicious node just behaves like normal node, but it will reveal all transmitted information to attackers. In *stop forwarding* attack, a malicious node drops the received key establishment packets. In this way, it can prevent the *indirect key* from being established via the malicious node. In *cheating* attacks, a malicious node alters the received key establishment packets. In this way, it can prevent the key establishment pairs from establishing the correct key.

1.2. Guarding Against Byzantine Attack

To safeguard the *indirect key*, multiple key-path schemes have been proposed in [2] and [19] to prevent the faulty sensors from deriving *indirect keys*. In [2], multiple *physically* link-disjoint paths between two nodes are used to set up an *indirect key*. When two nodes u and v want to set up an *indirect key* via multiple (say $j > 1$) link-disjoint paths, the source node, e.g., node u , selects j secrets, s_1, \dots, s_j , and sends each of the secrets onto a unique key establishment path. To secure a secret message, say s_1 , via a key establishment path, say $u \rightarrow x \rightarrow v$, following key establishment steps are performed:

$$u \rightarrow x : \{s_1\}_{k_{ux}}; x \rightarrow v : \{s_1\}_{k_{xv}}$$

where k_{ux} and k_{xv} are *direct keys* shared between pair (u, x) and pair (x, v) , respectively. Upon receiving all the

secrets, node v just simply uses bitwise *XOR* operation to derive the *indirect key*, i.e.,

$$indirect_key = s_1 \oplus \dots \oplus s_j. \quad (1)$$

In [19], multiple *logical* link-disjoint paths between two nodes are used to set up an *indirect key*. A *logical* path means there exists a key sharing relations among source, destination, and intermediate nodes along the key establishment path. For example, source node u share t_1 *direct keys* with intermediate node x and node x shares t_2 *direct keys* with destination node v (note that u and v do not share a *direct key*). Since a *direct key* can be only used for one *logical* path, there can be $z_x = \min(t_1, t_2)$ key establishment paths between u and v via intermediate node x . The secrets selection and transmission proposed in [19] is similar to the one described in [2]. The difference is the use of *physical* or *logical* key establishment paths in corresponding proposed schemes.

Both proposed multi-path key establishment schemes are efficient to guard against outsider's node capture attacks and *Byzantine* attacks by passively learning the forwarded messages. However, they are vulnerable to active *Byzantine* attacks, i.e., an attacker can stop forwarding the secrets or alter the forwarded secrets which can prevent the receiver from deriving the right *indirect key*.

1.3. Contribution of the work & Organization

In this paper, we propose a new multiple node-disjoint paths key establishment scheme. This scheme is based on error-correct coding scheme – Reed-Solomon Codes [15]. A sensor applies Reed-Solomon encoding scheme to partition an *indirect key* into multiple codewords. Each codeword is transmitted via a different node-disjoint path. The receiver applies Reed-Solomon decoding scheme to identify the faulty key establishment paths and then recover the original *indirect key*.

In a sensor network, sensors share the same communication channel, thus the physical node-disjoint paths may not be possible. However, messages are protected by pairwise keys; only legitimate sensors can decrypt the information. In this way, we can form multiple virtual node-disjoint paths through shared channels. The main benefits of proposed scheme are three folds:

- The proposed scheme is resilient to t faulty paths. If (n, k) Reed-Solomon codes is used, $t = (n - k)/2$.
- The receiver can identify faulty key establishment paths.
- No interactive communications are required to identify the faulty key establishment paths, which is communication efficient.

preinstalled secrets for each sensor generator polynomial $g(x)$, $2t$ roots $\alpha_1, \dots, \alpha_{2t}$, where $n - k = 2t$, see Equation (2)	
key establishment procedure, (n, k) RS codes, multi-path scheme	
sender u	receiver v
<ol style="list-style-type: none"> 1. generates p node-disjoint paths between u and v, see [1] 2. generates key message polynomial $m(x)$, see Algorithm 3 3. creates k codewords $m'_i, i = 1, \dots, k$, see Algorithm 3 4. uses source routing to send at most t codewords on each path where $t = (n - k)/2$ 	<ol style="list-style-type: none"> 1. uses majority rule to eliminate bad codeword(s) 2. composes received key polynomial $r(x)$ 3. uses Equation (5) to identify faulty path(s) 4. uses Forney's algorithm to derive error polynomial $e(x)$ 5. recovers the original message polynomial, see (8)
properties of proposed multi-path key establishment scheme	
<ol style="list-style-type: none"> 1. resilient to $t = (n - k)/2$ faulty paths when (n, k) Reed-Solomon codes is used. 2. receiver can identify the faulty key establishment paths. 3. no interactive communications are required, thus is communication efficient. 4. Reed-Solomon error correcting codes are computationally efficient: in the order of $O(n \log^2 n)$, see [16]. 	

Table 1. t -faulty resilient multi-path key establishment scheme

Note that the proposed scheme is not restricted only to sensor network and key management. It also applies to general multi-path data transmission to guard against *Byzantine* attacks.

The rest of the paper is organized as follows: in section 2, our proposed Reed-Solomon error correcting codes based multi-path key establishment scheme is discussed in details; we present the robustness analysis of our schemes in terms of insider passive/active attacks in section 3; finally, we give conclusion and future work in section 4.

2. Node-disjoint Multi-path Key Establishment Schemes

The multi-path key establishment schemes described in [2, 19] are vulnerable to *Byzantine* attacks. One of the reasons is that both these schemes use multiple link-disjoint paths to set up *indirect keys*. Their schemes are originally proposed to mitigate node capture attacks. Since a *direct key* can be preinstalled in multiple nodes by using the key predistribution schemes proposed in [5, 2], an attacker can capture nodes and derive the preinstalled keys that may be used among uncompromised nodes. Thus, using multiple link-disjoint paths to establish *indirect keys* can help to mitigate the node capture attack¹. However, attackers can perform more sophisticated attacks, such as node fabrication attacks [7] and *Byzantine* attacks. The attacker can fabricate sensors based on captured nodes and implant malicious codes in the fabricated sensors. These fabricated sensors can be deployed in the sensor network that can perform malicious functions, such as sniffing, stop forwarding, and cheating. Link-disjoint multi-path key establishment scheme cannot mitigate these attacks, since the malicious nodes can serve as proxy to set up *indirect keys*

and they can perform malicious functions without being detected.

Another reason that the previously proposed multi-path key establishment schemes are vulnerable to *Byzantine* attacks is the proposed key recovery scheme (i.e., bitwise XOR multiple secrets, see Equation (1)); it is vulnerable to packet dropping or packet altering. For example, only one bit altering or packet dropped in one path can prevent the receiver from correctly deriving the *indirect key*. Moreover, there is no way for the receiver to identify the problem source. Zhu et al. [19] proposed using a (k, n) threshold secret sharing scheme [17]. Such a threshold secret sharing scheme, based on polynomial interpolation, allows a node receiving any k (out of the n) shares to recover an *indirect key*, while no information about the *indirect key* can be determined with less than k shares. In other words, this scheme allows malicious nodes on $n - k$ disjoint key paths to drop packets and the receiver still can recover the *indirect key*. Although, the threshold secret sharing approach can mitigate the stop forwarding attacks, it can do little on cheating attacks. The original threshold secret sharing scheme proposed by Shamir [17] cannot detect and identify the cheaters.

In order to counter stop forwarding and cheating attacks, we propose a new multi-path pairwise key establishment scheme. Our proposed scheme employs multiple node-disjoint paths and Reed-Solomon error correcting codes [15] to mitigate stop forwarding and cheating attacks. The properties and operational procedures of the proposed scheme are shown in Table 1. In the following sections, we will discuss the proposed scheme in detail.

2.1. Multi-path Key Establishment Schemes

In general, p -node-disjoint paths are obtained from the knowledge of $(p - 1)$ -node-disjoint paths by applying the

¹ A link is protected by a *direct key*.

shortest path algorithm in a modified graph. The modified graph is obtained by replacing the edges of the $(p-1)$ -node-disjoint paths with arcs directed towards the source node, and making these arcs negative; in addition, the nodes, except for the endpoint vertices of the $(p-1)$ disjoint paths, are split in accordance with the node-splitting method. Note that the p -node-disjoint paths algorithms have been extensively studied. The detail description of the $p(> 2)$ node-disjoint shortest path algorithm is out scope of this paper. For details, please see [1].

In this section, our focus is to analyze the number of available node-disjoint paths for a node in large-scale sensor networks. In our analysis, we utilized the sensor deployment method proposed in [7]. We use n' to denote the average number of direct communication neighbors of a sensor and p_1 to denote the probability that two neighbors share a *direct key*. Thus the maximum number of node-disjoint paths $\max_number_paths \leq n'p_1$. In [6], we have derived the probability $p_r(h, n')$ that a node can reach any of its n' neighbors within h hops. Due to the requirement that multiple paths are node-disjoint, once a node is selected by a path, it cannot be used by other paths. The following algorithm is used to compute the number of paths within h hops with the probability $p_r(H, n') \geq T$, where T is a probability threshold and H is the hop count threshold.

Algorithm 1 (Number of paths)

- Initialize $H, n',$ and T ; set $h = 2$ and $P = 0$
1. if $h \geq H$ && $p_r(h, n') < T$
 Stop;
 else if $p_r(h, n') \geq T$
 $n' = n' - h$; $P = P + 1$;
 else
 $h = h + 1$;
 2. goto 1;
-

Based on Algorithm 1, we plot the average number of paths between two nodes in Figure 1. We analyze the case where the number of neighbors range from 25 to 50. We notice that, with the increasing number of neighbors, the number of paths between a pair of nodes increases.

2.2. Multi-path Encoding

The Reed-Solomon codes (RS codes) are nonbinary cyclic codes with code symbols from a Galois field [15]. RS codes have been widely used in many applications from compact discs and digital TV to spacecraft and satellite.

Briefly, RS codes are defined as follows. Let α be a primitive element in the Galois Field, $GF(2^q)$. For any positive integer $t \leq 2^q - 1$, there exists a t -symbol-error-correcting

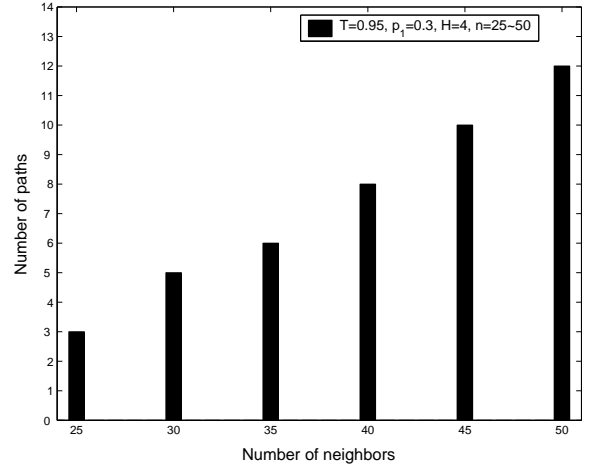


Figure 1. Number of paths to a neighbors ($n = 25 \sim 50$, average number of neighbors; $H = 4$, maximum number of hops; $T = 0.95$, probability threshold to evaluate a viable path; $p_1 = 0.3$, probability a node shares a *direct key* with one of its neighbors.

RS code with symbols from $GF(2^q)$ with following parameters:

$$n = 2^q - 1$$

and

$$n - k = 2t$$

where n is the total number of code symbols in the encoded block, t is the symbol-error correcting capability of the code, and $n - k = 2t$ is the number of parity symbols. A publicly known generator polynomial is of the form:

$$g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{2t}) \quad (2)$$

$$= g_0 + g_1x + g_2x^2 + \cdots + g_{2t-1}x^{2t-1} + x^{2t}$$

where $g_i \in GF(2^q)$ and $g(x)$ has $\alpha, \alpha^2, \dots, \alpha^{2t}$ as roots.

Using the RS encoding algorithm (see Algorithm 3 in Appendix A), we can derive the codeword polynomial $c(x)$:

$$c(x) = b(x) + x^{2t}m(x) \quad (3)$$

where

$$b(x) = x^{2t}m(x) \mod g(x) \quad (4)$$

and where $b(x)$ is the parity polynomial, $m(x)$ is the message polynomial.

Every element in $GF(2^q)$ can be represented uniquely by a binary q -tuple, called a q -bit byte. Suppose an (n, k) RS code with symbols from $GF(2^q)$ is used for encoding binary data. A message of kq bits is first divided into k q -bit bytes. Each q -bit byte is regarded as a symbol in $GF(2^q)$.

The k -byte message is then encoded into n -byte codeword based on the RS encoding rule. By doing this, we actually expand a RS code with symbols from $GF(2^q)$ into a binary (nq, kq) linear code, called a binary RS code. Binary RS codes are very effective in correcting bursts of bit errors as long as no more than t q -bit bytes are affected.

If the length of an *indirect key* is kq , we can divide kq bits into k q -bit segments. For example, in (15, 9), 3-symbol-error correcting RS code over $GF(2^4)$, $q = 4, k = 9, n = 15$. The key length is $kq = 36$ and the parity length is $(n - k)q = 24$. Thus, we can divide the key into $k = 9$ codewords. Each codeword contains $q = 4$ bits key information and 24 bits parity information, and the total length of a codeword is 28 bits. Note that the 24 bits parity code is used to check the entire 36 key message. In above example, we partition a kq -bit key message into k q -bit bytes. In order to increase the resilience to *Byzantine* attacks, each codeword can be transmitted via a node-disjoint path. The following algorithm is used to partition keys into k codes and the k codes are transmitted via p node-disjoint paths:

Algorithm 2 (Path-encoding)

Goal: node u sets up an indirect key M with node v

Initial: (n, k) t -symbol-error correcting RS code over $GF(2^q)$, size of an indirect key M is kq

1. *create message polynomial*
 $m(x) = m_0 + m_1x + \dots + m_{k-1}x^{k-1}$,
where m_i is a q bits code
2. *compute $b(x) = x^{2t}m(x) \bmod g(x)$*
where $b(x) = b_0 + b_1x + \dots + b_{2t-1}x^{2t-1}$
 $b = b_0 || b_1 || \dots || b_{2t-1}$ *is the parity code*
3. *compute p node-disjoint shortest paths,*
where $2t < p \leq k$
4. *create codewords $m'_i = (m_i || b)$, $i = 1, \dots, k$,*
and send at most t codewords on a path

* $||$ is concatenation operator

In Algorithm 2, node u wants to set up an *indirect key* with node v . Node u randomly picks up an *indirect key* $M = (m_0 || m_1 || \dots || m_{k-1})$ and creates the message polynomial $m(x)$. By using the RS encoding algorithm (see Algorithm 3 in Appendix A), node u derives parity polynomial $b(x)$ and corresponding parity code b . It then computes p -node-disjoint shortest paths, where $2t < p \leq k$ is to guarantee no more than t codewords are delivered via the same path. Finally, node u sends at most t codewords on each path. We assume $k, p > 2t$, thus we can use majority rule to rule out the compromised parity code b transmitted via the compromised paths. In this way, we can recover the *indirect key* when at most t paths are compromised.

2.3. Multi-path Decoding

If malicious nodes alter the parity code b , we can use the majority rule to rule out the compromised codeword. Thus, the attacker may want to alter forwarded codeword m_i instead of parity code b . From Equation (3) and Equation (4), we know the roots of $g(x)$ must also be the roots of $c(x)$. Since the received codeword $r(x) = c(x) + e(x)$, where $e(x) = \sum_{j=0}^{n-1} e_j x^j$ is the error polynomial, $r(x)$ evaluated at each of the roots of $g(x)$ should yield zero only when it is a valid codeword. Any errors will result in one or more of the computations yielding a nonzero result. The computation of a syndrome symbol can be described as follows:

$$\mathcal{S}_i = r(x)|_{x=\alpha^i} = r(\alpha^i) \quad i = 1, \dots, 2t \quad (5)$$

If there exists v errors, where $0 \leq v \leq t$, in the unknown locations j_1, j_2, \dots, j_v , then

$$e(x) = e_{j_1}x^{j_1} + \dots + e_{j_v}x^{j_v} \quad (6)$$

Define the error values to be $Y_l = e_{j_l}$, where $l = 1, 2, \dots, v$. And the error locators to be $X_l = \alpha^{j_l}$, where $l = 1, 2, \dots, v$. We can utilize Forney's algorithm[14] to derive the error values Y_l . Thus, we can derive the error correcting polynomial:

$$e(x) = \sum_{l=1}^v Y_l x^{j_l} \quad (7)$$

Then, we can recover $c(x)$ as

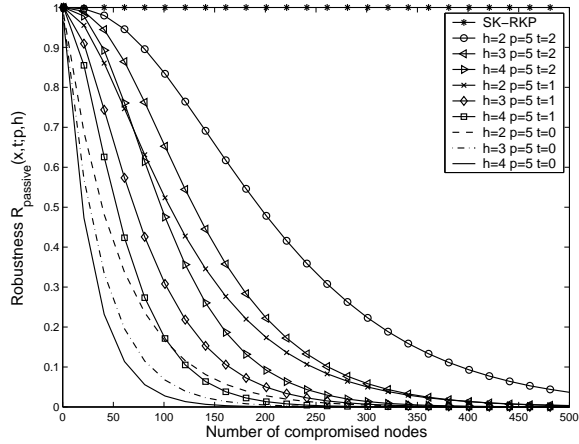
$$c(x) = r(x) - e(x). \quad (8)$$

Finally, applying Equation (3), we can derive the key message polynomial $m(x)$ and then derive the *indirect key* $M = (m_0 || \dots || m_{k-1})$.

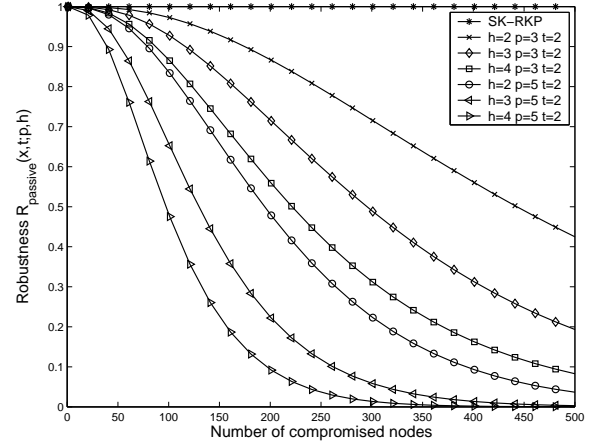
3. Robustness Analysis

In this section, we analyze the robustness of our proposed multi-path key establishment scheme under *Byzantine* attacks. The attacker can actively attack *indirect keys* establishment by replicating captured sensors and deploy the replicated sensors in sensor networks. In order to derive *indirect keys* established among uncompromised nodes, the attacker first inspects the *key graph*² topology by passively sniffing the traffic and quickly deploy the replicated sensors in the sensor network to create a shortcut of the *key graph* used by uncompromised nodes. These replicated sensors can serve as intermediate nodes in the *key paths*³. Then,

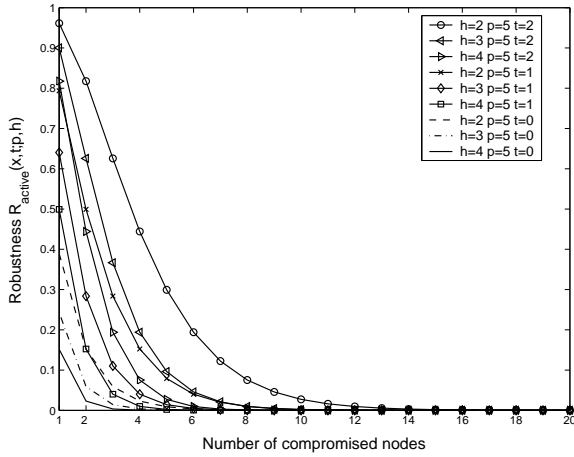
2 A *key graph* maintained by node i is defined as $G_i = (V_i, E_i)$ where, $V_i = \{j | j \in W_i \vee j = i\}$, $E_i = \{e_{jk} | j, k \in V_i \wedge k \in W_j \wedge j \in W_k \wedge jSk\}$, \mathcal{S} is a relation defined between two nodes if they share one *direct key*.



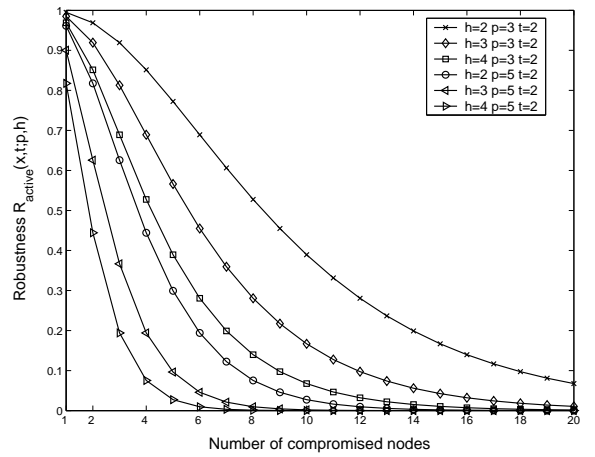
(a) Passive path-key establishment attack to P-RKP and SK-RKP schemes (same number of paths with different path-length and resilient factor $-t$)



(b) Passive path-key establishment attack to P-RKP and SK-RKP schemes (different number of paths and path-length with the same resilient factor $-t$)



(c) Active path-key establishment attack to P-RKP and SK-RKP schemes (same number of paths with different path-length and resilient factor $-t$)



(d) Active path-key establishment attack to P-RKP and SK-RKP schemes (different number of paths and path-length with the same resilient factor $-t$)

Figure 2. Attacks to path-key establishment (passive vs. active, $h = 2, 3, 4; p = 3, 5; t = 1, 2$)

they can derive the *indirect keys* established via these *key paths*.

As we have discussed, single path key establishment scheme is vulnerable to *Byzantine* attacks. Several multipath key establishment schemes have been proposed. In [2, 19], the authors proposed to divide *indirect* key into multiple pieces and then send them via multiple paths; in the following analysis, we call such schemes as *Byzantine* deficient schemes. We define the robustness $R(x, \cdot; \cdot, \cdot)$ as the probability of an *indirect* key is not compromised due to x

³ A *key path* between node A and B is defined as a sequence of nodes $A, N_1, N_2, \dots, N_i, B$, such that, each pair of nodes $(A, N_1), (N_1, N_2), \dots, (N_{i-1}, N_i), (N_i, B)$ has one *direct* key. The *length* of the *key path* is the number of pairs of nodes in it.

compromised nodes.

3.1. Resilience Against Passive Key Establishment Attacks

Suppose that there are x compromised nodes that collude by sharing their key sets. The probability that a key is not selected by a sensor is $1 - m/P$ and is not selected by x sensors is $(1 - m/P)^x$, where P is the total size of the key pool, m is number of keys selected for a sensor. Thus a key is selected by at least one of x sensors is $1 - (1 - m/P)^x$. We assume that the *direct keys* used on a *key path* are all different; thus, all *direct keys* on a *key path* with h hops are not selected by x sensors is $[(1 - m/P)^x]^h$. Hence, the proba-

bility that at least one of *direct keys* on a h hop path is compromised is:

$$p_{passive}(x, h) = 1 - (1 - m/P)^{xh}. \quad (9)$$

If all *key paths* have the same length h , we define robustness as:

$$\begin{aligned} & R_{passive}(x, t; p, h) \\ &= \sum_{i=0}^t \binom{p}{i} (p_{passive}(x, h))^i (1 - p_{passive}(x, h))^{p-i} \end{aligned} \quad (10)$$

In *Byzantine* deficient multi-path schemes, $t = 0$. For our proposed multi-key path key establishment scheme (which we call it as *Byzantine* resilient scheme), we have $t > 0$.

In Figure 2(a) and 2(b), we notice that using different key predistribution schemes, such as the SK-RKP key distribution scheme, the resilience to passive path-key establishment attacks can be increased dramatically compared to *Byzantine* deficient schemes. For sensor-class attackers, the insider nodes must be physically located on all *key paths* to derive the *indirect keys*, since a unique key can be derived between a pair of sensors. It requires that at least p insider nodes are deployed in the sensor network and each of the insider nodes is located on one of the p -disjointed *key paths*. Figure 2(a) and 2(b) show that (i) the P-RKP scheme is still vulnerable to *passive key establishment attack* and the SK-RKP scheme is perfectly resilient to this type of attacks unless the sensors involved in the *indirect key* establishment procedure are compromised; (ii) shorter path length exhibit better *Byzantine* resilience; (iii) using RS codes, if t is closer to the number of path p , we obtain the most gain in robustness from our scheme (see Figure 2(b)).

3.2. Resilience Against Active Key Establishment Attacks

For the SK-RKP scheme, $p_1 = 1 - \frac{\binom{\omega}{\tau} \binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^2}$ is the probability that two nodes share at least one key, where ω is the number of key space, τ is the number of selected key spaces for a sensor. For details, see [9, 4]. For the P-RKP scheme, we have $\omega = P, \tau = m$; for details, see [5]. Thus, $1 - p_1^2$ is the probability that a node does not share *direct keys* with at least one of two uncompromised nodes; thus, $(1 - p_1^2)^x$ is the probability that all x compromised nodes do not share *direct keys* with at least one of two uncompromised nodes; for a h -hop *key path*, $[(1 - p_1^2)^x]^h$ is the probability that all x compromised nodes do not share at least one of two uncompromised nodes on any hop of this *key path*. Thus, for a *key path* with h hops, the probability that at least one of x compromised nodes can create a short cut on a *key path* is:

$$p_{active}(x, h) = 1 - [(1 - p_1^2)^x]^h \quad (11)$$

If all *key paths* have the same length h , we have

$$\begin{aligned} & R_{active}(x, t; p, h) \\ &= \sum_{i=0}^t \binom{p}{i} (p_{active}(x, h))^i (1 - p_{active}(x, h))^{p-i} \end{aligned} \quad (12)$$

Note that *Byzantine* deficient multi-path schemes, $t = 0$.

Figure 2(c) and 2(d) show that both P-RKP and SK-RKP schemes are vulnerable to active path-key attacks. Comparing with the passive path-key establishment attack shown in Figure 2(a) and 2(b), the robustness of the PKE protocol decreases dramatically. Similar to the discussion in the case of passive attacks, we notice that increase in threshold t will increase the resilience to *Byzantine* attacks. We note that Figure 2(c) and 2(d) show the *worst case* of our proposed scheme. In reality, the source node u will arbitrarily select p node-disjoint paths to set up *indirect keys*. The malicious nodes may not be physically located on the selected key setup path. While in Figure 2(c), we address the worst case of the studied multi-path PKE schemes, i.e., we assume that a path is compromised as long as we can find a compromised node that can create a short-cut on any possible key path and this compromised node is physically located in the compromised key path.

4. Conclusion

In this paper, we propose a multi-path pairwise key establishment scheme to counter *Byzantine* attacks, i.e., attacks due to packet dropping and cheating. In our approach, we apply $p(\geq 2)$ node-disjoint paths key establishment scheme and embed the Reed-Solomon error correcting coding scheme. Our proposed scheme is resilient to $t = (n - k)/2$ faulty paths when (n, k) Reed-Solomon codes is used. The receiver can identify the faulty key establishment paths and no interactive communications are required, and is communication efficient.

Our proposed scheme can tolerate up to t faulty paths between the communication pairs. In our future work, we plan to study how to improve the resilience to *Byzantine* attacks with minimal interactive communications involved.

Acknowledgement

The authors would like to thank anonymous reviewers for their valuable comments.

References

- [1] R. Bhandari. *Survivable Networks – Algorithms for Diverse Routing*. Kluwer Academic Publishers, 1999.

- [2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of 2003 Symposium on Security and Privacy*, pages 197–215, Los Alamitos, CA, 11–14 2003. IEEE Computer Society.
- [3] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings of IEEE Information Communications Conference (INFOCOM)*, March 2004.
- [4] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 42–51, October 2003.
- [5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of 9th ACM Conference on Computer and Communication Security (CCS-02)*, pages 41–47, November 2002.
- [6] D. Huang, M. Mehta, D. Medhi, and L. Harn. Modeling pairwise key establishment for random key predistribution in large-scale sensor networks. *submitted for publication and available at http://conrel.sice.umkc.edu/HRP/modelling_pairwise_key_establishment_schemes-v2.pdf*, 2004.
- [7] D. Huang, M. Mehta, D. Medhi, and H. Lein. Location-aware key management scheme for wireless sensor networks. In *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pages 29–42, October 2004.
- [8] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
- [9] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 52–61, October 2003.
- [10] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (CCS'03)*, pages 72 – 82, 2003.
- [11] M. Mehta, D. Huang, and L. Harn. A practical scheme for random key predistribution and shared-key discovery in sensor networks. In *Proceedings of 24th IEEE International Performance Computing and Communications Conference*, 2004.
- [12] R. D. Pietro, L. V. Mancini, and A. Mei. Efficient and resilient key discovery based on pseudo-random key pre-deployment. In *Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, April 2004.
- [13] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the Association for Computing Machinery*, 36(2):335–348, 1989.
- [14] I. S. Reed and X. Chen. *Error-Control Coding for Data Networks*. Kluwer Academic Publishers, 1999.
- [15] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *SIAM Journal of Applied Math*, 8:300–304, 1960.
- [16] D. V. Sarwate. On the complexity of decoding goppa codes. *IEEE Transactions on Information Theory*, 23(4):515–516, 1977.
- [17] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- [18] A. Tsirigos and Z. J. Haas. Analysis of multipath routing part i: The effect on the packet delivery ratio. *IEEE Transactions on Wireless Communications*, 3(1):138–146, 2004.
- [19] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In *Proceedings of 11th IEEE International Conference on Network Protocols (ICNP)*, November 2003.

Appendices A and B can be found in the Reed and Chen's book "Error-Control Coding for Data Networks"[14]. We have reproduced them here for the ease of following the paper.

A. Encoding of Reed-Solomon Codes

Algorithm 3 (Encoding of RS Codes)

-
1. let $m(x) = m_0 + m_1x + \dots + m_{k-1}x^{k-1}$ be the message polynomial to be encoded where $m_i \in GF(2^q)$ and $k = n - 2t$
 2. Dividing $x^{2t}m(x)$ by $g(x)$, we have $x^{2t}m(x) = a(x)g(x) + b(x)$ where $b(x) = b_0 + b_1x + \dots + b_{2t-1}x^{2t-1}$ $b(x)$ is the parity check polynomial
Then $c(x) = b(x) + x^{2t}$ is the codeword polynomial for the message $m(x)$
-

B. Decoding of Reed-Solomon Codes

Algorithm 4 (Decoding of RS Codes)

-
1. $c(x) = c_0 + c_1x + \dots + c_{n-1}x^{n-1}$
 $r(x) = r_0 + r_1x + \dots + r_{n-1}x^{n-1}$
 $e(x) = e_0 + e_1x + \dots + e_{n-1}x^{n-1}$
where $c_i, r_i, e_i \in GF(2^q)$,
 $r(x)$ is received codeword,
and $e(x) = r(x) - c(x)$ is the error polynomial where $e_i = r_i - c_i$ is a symbol in $GF(2^q)$
 2. Suppose $e(x)$ has v errors at the locations then, $e(x) = e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \dots + e_{j_v}x^{j_v}$
The error-location numbers are $X_{j_1} = \alpha^{j_1}, X_{j_2} = \alpha^{j_2}, \dots, X_{j_v} = \alpha^{j_v}$
Using Forney's algorithm[‡], the error values are $Y_l = e_{j_l}, l = 1, \dots, v$
 3. To recover the original message $c(x)$ we have $c(x) = r(x) - e(x)$
-

[‡]: readers can refer to [14] for details of Forney's algorithm.