Multipath Key Establishment for Wireless Sensor Networks Using Just-Enough Redundancy Transmission

Jing Deng, *Member*, *IEEE*, and Yunghsiang S. Han, *Senior Member*, *IEEE*

Abstract—In random key predistribution techniques for wireless sensor networks, a relatively small number of keys are randomly chosen from a large key pool and are loaded on the sensors prior to deployment. After deployment, each sensor tries finding a common key shared by itself and each of its neighbors to establish a link key to protect the wireless communication between themselves. One intrinsic disadvantage of such techniques is that some neighboring sensors do not share any common key. In order to establish a link key among these neighbors, a multihop secure path may be used to deliver the secret. Unfortunately, the possibility of sensors being compromised on the path may render such an establishment process insecure. In this work, we propose and analyze the Just-Enough Redundancy Transmission (JERT) scheme that uses the powerful Maximum-Distance Separable (MDS) codes to address the problem. In the JERT scheme, the secret link key is encoded in (n, k) MDS code and transmitted through multiple multihop paths. To reduce the total information that needs to be transmitted, the redundant symbols of the MDS codes are transmitted only if the destination fails to decode the secret. The JERT scheme is demonstrated to be efficient and resilient against node capture. One salient feature of the JERT scheme is its flexibility of trading transmission for lower information disclosure.

Index Terms—Wireless sensor networks, key predistribution, security, secret link key, symmetric key, maximum-distance separable codes.

1 Introduction

Wireless Sensor Networks (WSNs) have attracted significant interest from the research community due to their potentials in a wide range of applications such as environmental sensing, battlefield sensing, and hazard leak detection. The security problem of these WSNs are important, as the sensors might be deployed to unfriendly areas. When any of the sensors is compromised or captured, the information on the sensor is disclosed to the adversary, and its operation may be controlled by the adversary.

In order to secure the communication between a pair of sensors, a unique key is needed. Since public/private (asymmetric) keys require significantly more computation than secret (symmetric) key techniques, the latter is preferred in WSNs [1], [2], [3], [4], [5], [6]. In addition to the computation cost, public/private key techniques are based on the existence of a Certificate Authority (CA), but such CAs may be unavailable in WSNs. The distribution of a secret key for every possible communication link is nontrivial due to the large number of sensors and the limited onboard memory size. To this trend, key predistribution techniques have been proposed and studied [1], [2], [3], [4]. These techniques allow the sensors to randomly pick a relatively small number of keys from a large key pool, and two neighboring nodes[1] then try finding a common key that is shared by themselves.

In some WSNs, some nodes need to communicate with each of their neighbors, such as the cluster heads in the cluster-based WSNs [7]. Therefore, in order to secure all these communication, a key needs to be shared by such a node with each of its neighbors. Due to the randomness of the key selection process in key predistribution, some communication links do not have any common key. In [2], a secret link key delivery technique using a multihop secure path was proposed: one of the neighboring nodes finds a multihop secure path toward the other node. Each pair of neighboring nodes on the secure path share at least a common key, which could be different throughout the path. Then, a secret link key is generated from the source node and sent toward the destination through the secure path. The small geographical separation between the source and the destination enables prompt acknowledgment and efficient secret verification.

Such a multihop secure path scheme works quite well when none of the nodes on the path is compromised and all sensor nodes forward the secret key honestly. However, the scheme has security problems if any of the nodes is compromised or captured by the adversary. Such a compromise affects the multihop secure path scheme in the following way: 1) since the secret link key is decrypted and reencrypted by each sensor on the path, it may be

• *J. Deng is with the Department of Computer Science, University of New Orleans, New Orleans, LA 70148. E-mail: jing@cs.uno.edu.*
• *Y.S. Han is with the Graduate Institute of Communication Engineering, National Taipei University, Taipei 237, Taiwan, R.O.C. E-mail: yshan@mail.ntpu.edu.tw.*

Manuscript received 2 July 2006; revised 15 May 2007; accepted 21 Aug. 2007; published online 24 Oct. 2007.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0094-0706.
Digital Object Identifier no. 10.1109/TDSC.2007.70233.

1. In this work, we use "sensors," "nodes," and "sensor nodes" interchangeably.

disclosed to the adversary and 2) the adversary can modify or drop the information passing through.

In this work, we address the problem of compromised sensors modifying and eavesdropping on the secret information on such multihop paths. We use the powerful Maximum-Distance Separable (MDS) codes to develop the Just-Enough Redundancy Transmission (JERT) scheme to provide protection for information delivery. In the JERT scheme, the secret link key is encoded in (n, k) MDS code and transmitted through multiple multihop paths. To reduce the total information that needs to be transmitted, the redundant symbols of the MDS codes are transmitted only if the destination fails to decode the secret. Since paths with different hop counts may be compromised with different probabilities, we further differentiate the number of symbols sent through these paths. One salient feature of the JERT scheme is its flexibility of trading transmission for lower information disclosure. Our analysis and simulation results show that the proposed technique is highly efficient and resilient against node capture.

This paper is organized as follows: In Section 2, we overview related work. The secret link key delivery problem is formulated in Section 3. In Sections 4 and 5, we present the JERT scheme and our analysis. The performance evaluation results are provided in Section 6. We summarize and conclude this work in Section 7.

2 Related Work

Eschenauer and Gligor [1] proposed a random key establishment technique for WSNs. In this technique, each sensor is preloaded a number of keys that are randomly selected from a large key pool. After deployment, two neighbors can establish a secure communication if they share a common key. Otherwise, they need to exchange a secret key via a multihop secure path. Chan et al. [2] extended the technique into q-composite random key establishment technique, which forces two neighbors to establish a secure communication only when they share q common keys, where $q \geq 2$. Based on [1], two similar random key predistribution techniques that used multispace key pool to improve network resilience and memory usage efficiency were developed independently in [3] and [4].

A multipath key reinforcement technique was proposed in [2] to enable two nodes to establish secure communication, even if they do not share enough common keys (with the use of the q-composite technique). These two neighbors first identify all secure paths between themselves. Then, one node generates a set of random numbers (of the same size) for all the paths and sends one number to the other node through each of the paths. After the destination receives all the numbers, it exclusive-ORs all of them to obtain the secret link key. The multipath key reinforcement scheme significantly improves the protection of the secret link key from being disclosed to the adversary. This scheme takes care of only information disclosure to the adversary but not information modification. It fails if any of the paths is compromised by the adversary and the number is modified or dropped.

In [8] and [9], combinatorial set was used to distribute keys to sensors prior to deployment. Such a deterministic combinatorial set technique allows each key in the key pool to be assigned to a constant number of sensor nodes. Therefore, the number of nodes that each sensor shares a common key is fixed.

A Secure Routing Protocol (SRP) was proposed in [10] to send additional information to protect routing information from being dropped. To combat the problem of topology instability in wireless networks, a multipath routing scheme was proposed and investigated in [11]. 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 of fragmenting the packet into smaller blocks and delivering them over multiple paths. The focus of [10] and [11] is on the problem of missing some of the messages but not on the modification of them.

In [12], an efficient information dispersal mechanism was developed to provide security, load balance, and fault tolerance for communication networks. The mechanism uses Reed-Solomon (RS) codes to recover link faults and to provide security. In this technique, all redundancy is sent along with the information symbols, increasing the transmission overhead significantly.

RS codes were used in a similar way for the key establishment of WSNs by Huang and Mehdi [13]. The technique was proposed to combat Byzantine attacks on the multihop paths. With the use of the (n, k) RS codes, the proposed scheme is resilient to $t = (n - k)/2$ faulty paths, which may drop or alter the information sent through. Furthermore, the receiver can identify faulty paths, as long as their number is not greater than t. In this scheme, $k(2t + 1)$ symbols need to be transmitted, limiting its applications of RS codes for large k or t.[2] Compared with [13], our scheme employs an efficient incremental information transmission technique that lowers the expected overall overhead significantly, and it can also be performed with RS codes of large k or large t. Extra symbols are transmitted only when they are necessary. We argue that with the source and the destination being direct neighbors, the cost of acknowledgment transmission is low. We further take advantage of the fact that different paths have different probabilities of being compromised or becoming faulty. Therefore, different amounts of symbols are sent through paths of different lengths in the JERT scheme.

MDS codes have been used in the Automatic-Repeat-Request (ARQ) protocols to reduce the transmission overhead on communication systems [16], [17]. In [16], RS codes were used in a type-2 hybrid ARQ protocol. In the first transmission, a relatively high rate RS code with fewer redundancy is used. When an additional transmission is needed, only the redundant symbols are sent. With such a technique, the overall code rate is reduced. This scheme increases the system throughput by reducing the transmission overhead. In [17], punctured MDS codes were used for the type-2 hybrid ARQ protocol, and a modified

2. The (n, k) codes are used to share secrets among n users, any k of which can recover the secret cooperatively. The Shamir scheme [14] is one of such schemes. As pointed out in [15], an RS code may be treated as a special (n, k) secret-sharing scheme with tamper-resistant capability, because it can correct errors. Also pointed out in the same work, the complexity of the decoding algorithm of the RS code is similar to that of the Shamir scheme.

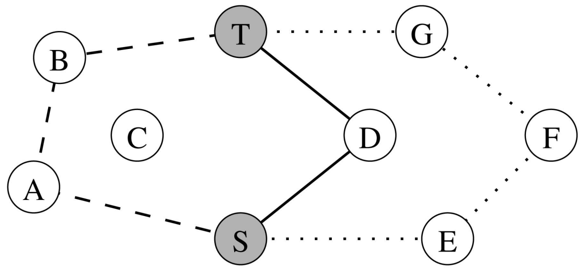


Fig. 1. Illustrations of multihop key establishment.

version (with fewer decoding operations) of the scheme proposed in [16] was presented.

3 Problem Formulation

We explain the link key establishment problem in WSNs in more detail in this section. The key predistribution schemes such as [1], [3], [4], and [18] provide memory-efficient and resilient ways of establishing secret link keys for *a fraction of* potential communication links. The rest of the communication links need to establish their secret keys by other means such as multihop delivery.

A few sensor nodes are shown in Fig. 1. Each line segment connecting two nodes represents that these two nodes share at least a common key. For example, nodes E and F share at least a key. Note that nodes S and T do not share any common key. Now, assume that nodes S and T, which are physical neighbors, need to establish a secure communication that requires a secret link key. As suggested in [1], [2], and [3], in order to establish a secret link key between nodes S and T, a multihop secure path may be used to deliver the secret key. For example, node D may be used to relay the secret key between nodes S and T. Since only one multihop path is used in the key delivery, we term it the Single-Path (SP) scheme.

The SP scheme may be summarized as follows: When node S needs to establish a secret link key K_{link} with node T, node S finds a path $S - N_1 - N_2 - \cdots - N_h - T$, where S shares a key with N_1, N_1 shares a key (could be different) with $N_2, \ldots$, and N_h shares a key with T. Then, node S encrypts K_{link} with the secret key shared by itself and node N_1 and sends the encrypted message to node N_1. Node N_1 decrypts K_{link} by using the common secret key shared with node S. Then, node N_1 sends K_{link} to node N_2 by using the same technique. This process continues until K_{link} reaches the destination, that is, node T. Some examples of such multihop paths in Fig. 1 are $S - D - T$, $S - A - B - T$, and $S - E - F - G - T$.

In general, random key predistribution schemes such as [1], [3], [4], and [18] may experience some communication links being exposed when some sensors are compromised. This is because some keys are simply reused by other communication links. In the multihop path link key establishment process, the secret link key K_{link} is decrypted and then reencrypted by each of the sensor nodes on the multihop path. If 1) any of these sensors is compromised during the WSN initialization process or 2) the adversary is able to decrypt the recorded information after it compromises sensor nodes later on, such a secret link key K_{link} is exposed. A compromised sensor may modify or drop the secret information passing through in the multihop path key delivery process. This leads to the following problem:

Problem statement. *In key predistribution schemes for WSNs, some neighboring sensors do not share any common key. Their secret link key needs to be established through multihop secure paths. However, when any of the sensors on the multihop secure path is compromised or captured by the adversary, the secret link key is disclosed. A compromised sensor may also modify or drop the key information passing through itself. What fault-tolerant mechanism should we use to send the secret link key between two physical neighbors efficiently and securely?*

Note that we work on the problem of sending secret link key information between two neighbors that do not share a common key after the key predistribution process. Other security provisions such as authentication and confidentiality may be provided once the secret key is delivered but are out of the scope of this work.

In Section 4, we introduce the powerful MDS codes and use them in our JERT scheme to solve this problem.

4 The Just-Enough Redundancy Transmission Scheme

4.1 Variable Definitions

For the sake of clarity and convenience for the readers, we list some major variables used throughout this paper:

- n: the number of total symbols of the MDS code.
- k: the number of information symbols of the MDS code ($k < n$).
- γ: the length of secret link key ($\gamma \leq k$).
- α: the threshold on the portion of information symbols being disclosed to the adversary.
- β: a primitive element in a finite field that can represent all nonzero elements in the finite field.
- t: the maximum number of errors (in symbols) that the MDS code can correct ($t = \lfloor (n-k)/2 \rfloor$).
- m: the number of available paths.
- p_{local}: the connectivity probability (on security plane).
- $\mathbf{c} = (c_0, c_1, \ldots, c_{k-1}, c_k, \ldots, c_{n-1})$: the code word of an (n, k) MDS code.
- e: the maximum number of transmissions that the JERT scheme performs.
- q_j: the fraction of the total symbols that are being transmitted in each round on the $j\text{th}$ path.
- $r_1, r_2, \ldots, r_e$: the number of extra symbols sent out by the source in each additional transmission.
- $r_0 = k$: the number of symbols sent out by the source in the first round of transmission.
- h_j: the hop count of the $j\text{th}$ path.
- x: the sensor node compromised probability.
- n_T: the number of total routers used by the JERT scheme.
- x_c: the number of compromised sensors on all of the multihop paths used by the JERT scheme.
- L: the maximum number of hops of all multihop paths used by the JERT scheme.
- m_x: the number of compromised paths among the m available multihop paths.

- $p_x^{(JERT)}$ and $p_x^{(SP)}$: the secret disclosure probabilities of the JERT scheme and the SP scheme, respectively.
- $\delta^{(JERT)}$ and $\delta^{(SP)}$: the expected numbers of transmitted symbols of the JERT scheme and the SP scheme, respectively.

4.2 Maximum-Distance Separable Codes

We first review the MDS codes [16], [17] that will be used in the JERT scheme. Let the Hamming distance between two vectors (code words) be the number of distinct positions between two vectors. An (n, k, d_{min}) MDS code is a linear block code whose minimum Hamming distance d_{min} between any pair of distinct code words must satisfy $d_{min} = n - k + 1$, where n is the code length, and k is the dimension of the code. Therefore, each code word in the $(n, k, n - k + 1)$ MDS code has exactly n symbols, among which there are k information symbols. Usually, the extra $n - k$ symbols are called parity checks or the redundancy of the code. Furthermore, an $(n, k, n - k + 1)$ MDS code will be able to recover any v errors if

$$v \leq \left\lfloor \frac{n-k}{2} \right\rfloor.$$

MDS codes are optimal in the sense that they provide the largest possible minimum Hamming distance between code words and hence can correct the most number of errors. The most famous family of MDS codes are RS codes. Efficient decoding algorithms for MDS codes have been studied extensively in [19] and [20]. In the following, we give a brief description of the encoder and the decoder of RS codes.

Let $GF(2^\tau)$ be the finite field of order 2^τ such that each element in $GF(2^\tau)$ can be represented by τ bits. An (n, k) RS code is a linear code, where each symbol is in $GF(2^\tau)$, with the following parameters:

$$n = 2^\tau - 1,$$

and

$$n - k = 2t,$$

where n is the total number of symbols in a code word, k is the total number of information symbols, and t is the symbol-error-correcting capability of the code. Let the sequence of k information symbols in $GF(2^\tau)$ be $\boldsymbol{m} = (m_0, m_1, \ldots, m_{k-1})$ and let $m(x)$ be the information polynomial of $\boldsymbol{m}$ represented as

$$m(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}.$$

The code word polynomial $c(x)$ corresponding to $m(x)$ can be encoded as

$$c(x) = m(x)g(x),$$

where $g(x)$ is a generator polynomial of the RS code. It is well known that $g(x)$ can be obtained as

$$\begin{aligned} g(x) &= (x + \beta)(x + \beta^2) \cdots (x + \beta^{2t}) \\ &= g_0 + g_1 x + g_2 x^2 + \cdots + g_{2t} x^{2t}, \end{aligned} \tag{1}$$

where β is a primitive element in $GF(2^\tau)$, and $g_i \in GF(2^\tau)$. Note that $g(x)$ has $\beta, \beta^2, \ldots, \beta^{2t}$ as roots.

Another way of encoding $c(x)$ is to use polynomial division as

$$c(x) = x^{2t} m(x) + p(x),$$

where

$$p(x) = x^{2t} m(x) \bmod g(x).$$

Since each symbol is represented by τ bits, an (n, k) RS code can be expanded to a $(\tau n, \tau k)$ binary linear block code. For example, a (1,023,255) RS code with 384 symbol-error-correcting capabilities is of $10 \times 255 = 2{,}550$ information bits. The computational cost of the encoder is roughly $k(n - k)$ additions and $k(n - k)$ multiplications.[3]

The decoding processes of RS codes are more complex. Let $r(x)$ be the received polynomial and let $r(x) = c(x) + e(x)$, where $e(x) = \sum_{j=0}^{n-1} e_j x^j$ is the error polynomial. Since $g(x)$ (and, hence, $c(x)$) has β, β^2, ..., β^{2t} as roots, the syndromes S_i can be calculated as

$$S_i = r(\beta^i) = e(\beta^i) = \sum_{j=0}^{n-1} e_j \beta^{ij} \text{ for } i = 1, \ldots, 2t. \tag{2}$$

Assume that $v \leq t$ errors occur in unknown locations $j_1, j_2, \ldots, j_v$ of the received polynomial. Then

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \cdots + e_{j_v} x^{j_v},$$

where e_{j_ℓ} is the value of the ℓth error, $\ell = 1, \cdots, v$. The decoding process finds all j_ℓ and e_{j_ℓ}. Instead of solving the set of the above $2t$ syndrome equations, an intermediate polynomial, called the error-locator polynomial, is introduced as

$$\Lambda(x) = \prod_{\ell=1}^{v} (1 - x\beta^{j_\ell}) = 1 + \Lambda_1 x + \cdots + \Lambda_v x^v.$$

The coefficients of the error-locator polynomial can be determined by the Berlekamp-Massy algorithm or euclidean algorithm that are of time complexity $O(t^2)$ [19], [20], [21]. Once all coefficients of the error-locator polynomial are found, β^{j_ℓ} can be determined by successive substitution through the Chien search [21]. Finally, e_{j_ℓ} can be calculated by the Forney formula [21].

Each of the RS decoding processes can be implemented in either hardware or software. Hardware implementations with moderate/high speed but small/large hardware have been proposed [19], [20], [21]. Software implementation of the RS decoding process can be programmed on a general-purpose processor [21]. The first step in the decoding of an RS code is to compute the $2t$ syndromes. Combining with the Horner rule, this step requires $(n - 1)t$ additions and nt multiplications. Finding the coefficients of the error-locator polynomial requires roughly $2t^2$ additions and $2t^2$ multiplications. In the worst case, the Chien search needs to substitute n field elements into the error-location polynomial of degree t to determine its roots. This requires

3. Addition and multiplication are performed on $GF(2^\tau)$, which can be implemented by a table lookup method.

nt multiplications and nt additions in the software implementation. The computational complexity of the Forney formula calculation, which is the final step of the decoding process, is similar to that of finding the coefficients of the error-locator polynomial. In total, $(2n-1)t + 4t^2$ additions and $2nt + 4t^2$ multiplications are needed to complete the decoding process.

When the lookup table technique is implemented, the additions and multiplications on $GF(2^\tau)$ have roughly the same complexity. Let β be a primitive element of $GF(2^\tau)$ and all elements in this field can be expressed as powers of β. Each multiplication calculation takes a modular addition of two exponents on $2^\tau - 1$. The addition calculation is implemented by the Zech logarithms [22] and takes one subtraction, one modular addition on $2^\tau - 1$, and one memory access. The memory usage is then $\tau 2^\tau$ bits to implement the Zech logarithms. For $GF(2^{10})$ used in our simulations, the memory required is only 1.25 Kbytes.

MDS codes have several nice properties that make them very useful. Two of such properties are given as follows without proof:

Property 1. *Punctured (shortened) MDS codes are MDS.*

A code is *punctured* when some parity symbols are deleted from each code word in the code. Similarly, a code is *shortened* when some information symbols are deleted from each code word in the code. For instance, an $(n, k, n-k+1)$ MDS code can be punctured (shortened) to an $(n-j, k, n-j-k+1)$ $(n-j, k-j, n-k+1)$ MDS code by deleting j corresponding parity (information) symbols from each code word.

Property 2. *Any k-coordinates of an MDS code can be used as information symbols.*

According to this property, by knowing any k symbols of a code word in an MDS code, we can recover other $n-k$ symbols for this code word.

4.3 The Just-Enough Redundancy Transmission Scheme

Let $\boldsymbol{c} = (c_0, c_1, \ldots, c_{k-1}, c_k, \ldots, c_{n-1})$ be a code word of an (n, k) MDS code over $GF(2^\tau)$, where $c_i \in GF(2^\tau)$, $0 \le i \le n-1$, is a symbol. Here, $c_0, \ldots, c_{k-1}$ are the information that the source needs to send to the destination. Assume that the secret link key generated by the source is of length $\gamma \le k$ and can be generated by a function f of these information if the destination decodes the code word $\boldsymbol{c}$ correctly. The design goal of our scheme is to tolerate up to e compromised paths and to send as few symbols as possible. Since an (n, k) MDS code can recover up to $(n-k)/2$ errors, n should satisfy

$$n \ge k + 2v, \tag{3}$$

where v is the number of errors occurring on a code word, and we have implicitly assumed that $n-k$ is even.

In this work, we assume that the costs of all one-hop transmissions are the same. Therefore, the transmission overhead or cost of sending a packet through one path is proportional to its hop count. The unit of such a cost is $[\text{symbol} \cdot \text{hop}]$. We neglect other extra costs such as MAC-layer headers and physical-layer headers. We argue that our transmission overhead analysis will hold, as long as the number of symbols being sent is large compared to these extra costs.

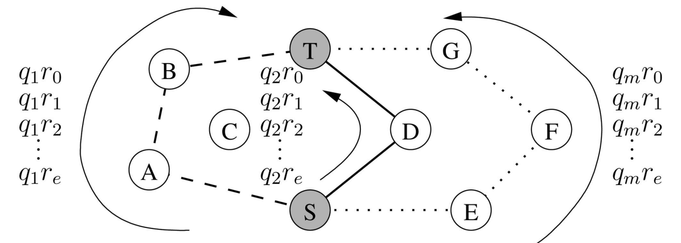


Fig. 2. Illustration of the JERT scheme. Assume that there are m node-disjoint multiple paths. The MDS code is separated into $r_0 = k, r_1, r_2, \ldots, r_e$. Each path j sends q_jr_i symbols in the ith round until the receiver decodes the transmitted secret successfully, or failure takes place when i reaches e.

In order to minimize such a transmission overhead and to simultaneously maximize the security protection of the scheme, the sender should send different numbers of symbols through different paths according to their hop counts. Assume that the sender transmits q_j fraction of the total symbols that are being transmitted toward the destination in one round through path j, where $1 \le j \le m$, and $\sum_{j=1}^{m} q_j = 1$. The values of q_j, $1 \le j \le m$, affects the performance of our scheme. We will determine q_j in Section 5.1.

A brief outline of the JERT scheme is given as follows (see Fig. 2): Let $r_0 = k$ be the number of symbols sent by the source in the original transmission. If the destination receives all information correctly and regenerates the secret key sent by the source, our scheme terminates. Otherwise, the source sends r_1 extra symbols in its first additional transmission. If the destination succeeds in the secret key regeneration, our scheme stops. Otherwise, the same process continues until the n symbols are exhausted.

Let $r_1, r_2, \ldots, r_e$ be the numbers of extra symbols sent out by the source in each additional transmission (the values of r_i, $1 \le i \le e$, will be determined in Section 5.2). Note that in our scheme, the source only needs to send out up to $\sum_{i=0}^{t_p} r_i$ symbols when t_p paths are compromised.[4] Let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{\ell-1})$ be the corresponding received vector at the destination when the source has sent out ℓ symbols up to now.

The JERT scheme is given as follows:

1. The source first encodes k symbols, $\boldsymbol{c}_0 = (c_0, c_1, \ldots, c_{k-1})$, into a code word with n symbols, $\boldsymbol{c} = (c_0, c_1, \ldots, c_{k-1}, c_k, \ldots, c_{n-1})$,[5] where $r_0 = k$. Initialize $i = 0$, $b = 0$, and $s = r_i - 1$.
2. The source transmits q_jr_i symbols specified by b and s in $\boldsymbol{c}$, that is, $\boldsymbol{c}_i = (c_b, c_{b+1}, \ldots, c_s)$, along

4. This is always true when $q_j = 1/m$ for all $1 \le j \le m$. For other values of q_j, t_p is the average result.

5. Note that the first k symbols of the code word $\boldsymbol{c}$ do not have to be the same as $\boldsymbol{c}_0$, as our notation suggests. When they are different, the destination or an adversary needs to use a decoding function to regenerate the information symbols. This increases the operational complexity but slightly improves the system security.

path j for $1 \leq j \leq m$. If $q_j r_i$ is not an integer, a round-off value will be used instead.

3. Assume that the destination receives all symbols from the m paths as $\boldsymbol{y}_i = (y_b, y_{b+1}, \dots, y_s)$.[6] The destination appends $\boldsymbol{y}_i$ to all the previously received symbols to form a longer code word. Then, it tries decoding this code word in order to obtain the k symbols. If the decode process fails due to more than i errors, then go to step 4 directly; otherwise, it verifies this result with the source through the challenge-response technique (recall that the source and the destination are direct physical neighbors). If the regenerated secret link key is verified, the transmission of the secret link key has succeeded; otherwise, go to step 4.
4. If $i = e$, then the key establishment fails due to too many compromised paths. Otherwise, the destination asks for another round of additional transmission.
5. The source sets $i = i + 1$, $b = s + 1$, and $s = s + r_i$ and repeats step 2.

Therefore, compared with [16] and [17], which have only one additional transmission, the JERT scheme sends multiple retransmissions when necessary.

Note that the JERT scheme does not need to know the identification of the compromised paths to decode the secret successfully. The MDS decoder at the receiver will automatically correct any modification by the compromised nodes in these paths and request for additional transmission when necessary.

4.4 Multihop Paths

Before we present our analysis of the JERT scheme, we discuss the path selection process and its effect on the performance of the JERT scheme. In this work, we assume that a source node identifies m multihop paths between itself and the destination. Such m paths could be chosen from the node-disjoint paths, in which none of the multihop paths shares any common node other than the source and the destination [23], [24], [25]. Another option is to allow the source node to randomly select among all available paths. The result is that some paths may have common nodes, and thus, the security performance worsens. The benefit of such a selection technique is that it does not rely on the availability of node-disjoint multihop paths and eliminates the cost of identifying such paths.[7]

As suggested by Chan et al. [2], it is always beneficial to choose short multihop paths instead of long multihop paths. As the length of a multihop path increases, the possibility of path compromise is higher. As we limit ourselves to short multihop paths, however, the number of available paths may be limited. We evaluate the values of m under various network conditions and the effect of such m paths on the security performance of our scheme in Section 6. The proposed JERT scheme works with any set of multihop paths and node-disjoint multihop paths.

6. The source can notify the destination regarding the number of transmitted symbols over plaintext (recall that the source and the destination nodes are direct physical neighbors). Therefore, the event of symbols being dropped is similar to that of symbols being modified along the multihop paths.

7. The procedure of establishing such secure paths is out of the scope of this paper. See [23], [24], and [25] for more details.

5 Analysis

In this section, we derive formulas for the fractions of symbols to be transmitted through each path q_j, $1 \leq j \leq m$, and the number of additional symbols to be transmitted in each round r_i, $0 \leq i \leq e$, for the JERT scheme. We will also investigate three performance aspects of the JERT scheme and the SP scheme, which sends the secret link key through a single multihop path: secret information disclosure, transmission overhead, and computation overhead.

5.1 Selection of $q_1, q_2, \dots, q_m$

Due to the lack of the knowledge of which paths may be compromised, the source node has the best option of making sure that the expected number of symbols compromised on each path is more or less the same.

Let h_j be the hop count of path j, $1 \leq j \leq m$, and x the probability of nodes being compromised. Then, the probability that path j is compromised, given that the source and the destination are not compromised,[8] is

$$\begin{aligned}
&Pr[\text{at least one router in between is compromised}|\\
&\quad \text{the source and the destination are not compromised}]\\
&= Pr[\text{at least one router in between is compromised}]\\
&= 1 - Pr[\text{none of the routers is compromised}]\\
&= 1 - (1-x)^{h_j - 1},
\end{aligned}$$

where $1 \leq j \leq m$.

The expected number of symbols being compromised for path j is

$$q_j \cdot [1 - (1-x)^{h_j-1}],$$

and the source node needs to make sure that

$$q_j \cdot [1 - (1-x)^{h_j-1}] = C, \text{ for all } j. \tag{4}$$

Since $\sum_{j=1}^{m} q_j = 1$, the constant C should satisfy

$$C = \frac{1}{\sum_{i=1}^{m} \frac{1}{1-(1-x)^{h_i-1}}}.$$

When x is small, $1 - (1-x)^{h_j-1}$ may be approximated as

$$1 - (1-x)^{h_j-1} \approx 1 - [1 - (h_j - 1)x] = (h_j - 1)x.$$

Therefore, C becomes

$$C \approx \frac{x}{\sum_{i=1}^{m} \frac{1}{h_i - 1}},$$

and we have

$$q_j \approx \frac{C}{(h_j - 1)x} = \frac{\frac{1}{h_j-1}}{\sum_{i=1}^{m} \frac{1}{h_i-1}}. \tag{5}$$

Thus, we have derived a closed form for q_j, $1 \leq j \leq m$, that is unrelated to x when $x \ll 1$.

8. A compromised source or destination makes the key exchange meaningless.

5.2 Selections of $r_1, r_2, \dots, r_e$

The values of $r_1, r_2, \dots, r_e$ can be determined as follows: In order to reduce the total number of symbols to be transmitted, in each transmission, we should add as little redundancy as possible, which can correct errors due to one more compromised path.

In general, when one round of the JERT scheme fails, the next round of transmission should provide the receiver *just-enough* symbols so that it can correct the errors due to one more compromised path. In our JERT scheme, however, each path transmits different amounts of symbols. Although this provides better security performance, such a transmission technique makes the determination of $r_1, r_2, \dots, r_e$ rather complex.

In this section, however, we determine $r_1, r_2, \dots, r_e$ with the help of the average number of the symbols transmitted on all routes. Thus, we assume that in the next round of transmission, each route carries the same amount of information. This makes our analysis tractable. In Section 6, we will show the effectiveness of our simplified model.

Let q_{avg} be the average value of q_j, $1 \le j \le m$. Since $\sum_{j=1}^{m} q_j = 1$, we have

$$q_{avg} = \frac{1}{m}. \tag{6}$$

In the following, we derive a set of $r_1, r_2, \dots, r_e$ such that when up to $i-1$ rounds of transmission fail to allow the receiver to regenerate the secret link key, the ith round of symbol transmission corrects the errors caused by one more path, $0 < i \le e$.

Since $q_{avg} = 1/m$, by noticing that r_1 symbols are added in order to correct the errors caused by one compromised path, we can determine r_1 as

$$\left(\frac{r_1}{m} + \frac{k}{m}\right) \le \frac{r_1}{2}. \tag{7}$$

The left side of (7) is the total number of errors introduced by the compromised path. The right side of (7) is the error correction capability due to the transmission of the additional r_1 symbols.

Taking the smallest integer that satisfies the above inequality, we have

$$r_1 = \left\lceil \frac{2k}{m-2} \right\rceil. \tag{8}$$

In general, the value of r_ℓ, where $1 \le \ell \le e$, must satisfy the following:

$$\frac{\ell}{m}\left(k + \sum_{i=1}^{\ell} r_i\right) \le \frac{1}{2}\sum_{i=1}^{\ell} r_i. \tag{9}$$

In order to reduce the total number of symbols transmitted, we choose the smallest r_ℓ that satisfies the above inequality. Therefore,

$$\begin{aligned} r_\ell &= \left\lceil \frac{2\ell k}{m-2\ell} - \sum_{i=1}^{\ell-1} r_i \right\rceil \\ &= \left\lceil \frac{2\ell k}{m-2\ell} \right\rceil - \sum_{i=1}^{\ell-1} r_i. \end{aligned} \tag{10}$$

Based on (10), when there are ℓ compromised paths between the source and the destination, the total number of additional symbols that should be transmitted is

$$\sum_{i=1}^{\ell} r_i = \left\lceil \frac{2\ell k}{m-2\ell} \right\rceil. \tag{11}$$

With the help of (11), we can rearrange (10) as

$$r_\ell = \left\lceil \frac{2\ell k}{m-2\ell} \right\rceil - \left\lceil \frac{2(\ell-1)k}{m-2(\ell-1)} \right\rceil, \tag{12}$$

when $1 \le \ell \le e$.

Since e is the maximum number of compromised paths that can be tolerated by the JERT scheme, the set of $r_1, r_2, \dots, r_e$ should satisfy:

$$\sum_{i=1}^{e} r_i \le n - k,$$

which leads to (based on (11))

$$\left\lceil \frac{2ek}{m-2e} \right\rceil \le n - k.$$

Therefore, e should satisfy

$$e < \frac{n-k}{n} \cdot \frac{m}{2}. \tag{13}$$

As a point of reference, when $n = 1{,}023$, $k = 255$, and $m = 15$, the maximum value of e is 5 due to (13). The array r_i, $0 \le i \le e$, is {255 40 53 77 122 218}.

5.3 Information Disclosure

We start our discussion on the security performance of information disclosure by presenting the attack model. An adversary takes control of some compromised sensors. The adversary may control the sensors to perform one of the following activities:

1. *Observing sensors*. The compromised sensors may silently observe and record all the symbols that are passing through, but they do not modify these symbols.
2. *Modifying sensors*. The compromised sensors may modify or drop the symbols that are passing through.
3. *Hybrid*. A combination of the above two categories of sensors. Some sensors observe and record the secret informations. Others modify or drop the secret information.

An analysis of the third category of actions would be quite complex and is out of the scope of this paper. We focus on the first two categories of actions instead. In this section, we study the security problem caused by the observing sensors.

Recall that x is node compromised probability, $0 \le x < 1$. Let $\Pr[P_{s_2, s_3, \dots, s_L}]$ be the probability of event S that the source and the destination find s_2 paths with length (hop count) 2, s_3 paths with length $3, \dots,$ and s_L with the maximum length L (note that all these paths are secure multihop paths). Such a probability is dependent on the algorithm to search for the node-disjoint paths and will be discussed in

Section 6. Since all found paths are node disjoint and $m = s_2 + \cdots + s_L$, the total number of routers is

$$n_T = \sum_{i=2}^{L} (i-1)s_i. \tag{14}$$

We denote B_{x_c} as the event that x_c out of the n_T routers are compromised. Based on an independent compromised probability x, the probability of event B_{x_c} taking place is

$$Pr(B_{x_c}) = \binom{n_T}{x_c} x^{x_c}(1-x)^{n_T - x_c}. \tag{15}$$

Let A_{m_x} be the event that m_x among the m available paths are compromised by the adversary.[9] Next, we evaluate $\Pr[A_{m_x}|B_{x_c} \cap S]$. Let $D_{n_2,\ldots,n_L}$ be the event that the m_x compromised paths contain n_2 paths among the s_2 paths with length $2,\ldots$ and the n_L paths among the s_L paths with length L. When $x_c < m_x$, $\Pr[A_{m_x}|B_{x_c} \cap S]$ is zero, since it is impossible to have more compromised paths than compromised sensors when all paths are node disjoint. When $x_c \geq m_x$,

$$\Pr[A_{m_x}|B_{x_c} \cap S] = \sum_{\substack{n_2+n_3+\cdots \\ +n_L=m_x}} \prod_{i=2}^{L} \binom{s_i}{n_i} \Pr[D_{n_2,\ldots,n_L}|B_{x_c} \cap S], \tag{16}$$

where $0 \leq n_i \leq s_i$ for $2 \leq i \leq L$.

Now, we need to derive $\Pr[D_{n_2,\ldots,n_L}|B_{x_c} \cap S]$. Assume that $n_{j_1}, n_{j_2}, \ldots, n_{j_g}$, $0 \leq g \leq L-1$ are the nonzero terms of $n_2, \ldots, n_L$. Therefore, there are n_{j_1} paths of length j_1 compromised, n_{j_2} paths of length j_2 compromised, $\ldots, n_{j_g}$ paths of length j_g compromised. Then, we need to consider all possibilities to select x_c compromised nodes from $x_n = \sum_{i=1}^{g} n_{j_i}(j_i - 1)$ nodes on the m_x compromised paths. Since all these m_x paths are compromised, at least one node on each path must be compromised. Hence,

$$\Pr[D_{n_2,\ldots,n_L}|B_{x_c} \cap S] = \begin{cases} \displaystyle\sum_{\substack{y_{1,1}+\cdots+y_{1,n_{j_1}} \\ +y_{2,1}+\cdots+y_{2,n_{j_2}}+\cdots \\ +y_{g,1}+\cdots+y_{g,n_{j_g}}=x_c}} \frac{\prod_{i=1}^{g}\prod_{\ell=1}^{n_{j_i}} \binom{j_i-1}{y_{i,\ell}}}{\binom{n_T}{x_c}}, & \text{if } x_n \geq x_c, \\ 0, & \text{otherwise}, \end{cases} \tag{17}$$

where $1 \leq y_{i,\ell} \leq j_i - 1$ for $1 \leq i \leq g$.

We now derive the *secret disclosure probability* p_x, which is defined as the probability of disclosing enough symbols to the adversary so that it can obtain the key with relative ease when the node-compromised probability is x.

In the SP scheme, the γ symbols are transmitted through one randomly chosen path among the m available paths. If the SP scheme selects a compromised path to transmit, then all γ symbols are revealed. Hence, when there are m_x compromised paths, the secret disclosure probability is m_x/m. The overall secret disclosure probability can be calculated as

$$p_x^{(SP)} = \sum_S \left\{ \Pr[P_{s_2,s_3,\ldots,s_L}] \sum_{x_c=1}^{n_T} \left[\binom{n_T}{x_c} x^{x_c} \cdot (1-x)^{n_T-x_c} \sum_{m_x=1}^{m} \left(\Pr[A_{m_x}|B_{x_c} \cap S] \frac{m_x}{m} \right) \right] \right\}, \tag{18}$$

where n_T is given by (14).

When the JERT scheme is used, the source transmits only $r_0 = k$ symbols, and the destination gets all of these symbols successfully, because no information is modified. Such k symbols are transmitted through the m paths with q_j fraction for path j, $1 \leq j \leq m$, where q_j is given by (5). For fair comparison between the JERT scheme and the SP scheme, we define the secret disclosure probability of the JERT scheme as the probability of at least αk symbols being disclosed to the adversary, where $0 < \alpha \leq 1$. Therefore, the secret disclosure probability can be calculated as

$$p_x^{(JERT)} = \sum_S \Pr[P_{s_2,s_3,\ldots,s_L}] \left\{ \sum_{x_c=1}^{n_T} \binom{n_T}{x_c} x^{x_c}(1-x)^{n_T-x_c} \left[\sum_{m_x=1}^{m} \Pr[A_{m_x} \cap E_\alpha | B_{x_c} \cap S] \right] \right\}, \tag{19}$$

where E_α is the event that the fraction of symbols transmitted through the compromised paths are greater than or equal to α and is given by

$$\frac{\sum_{i=2}^{L} n_i q_i}{\sum_{i=2}^{L} s_i q_i} = \frac{\sum_{i=2}^{L} \frac{n_i}{i-1}}{\sum_{i=2}^{L} \frac{s_i}{i-1}} \geq \alpha. \tag{20}$$

$\Pr[A_{m_x} \cap E_\alpha | B_{x_c} \cap S]$ is given as

$$\Pr[A_{m_x} \cap E_\alpha | B_{x_c} \cap S] = \sum_{\substack{n_2+n_3+\cdots \\ +n_L=m_x \\ E_\alpha}} \prod_{i=2}^{L} \binom{s_i}{n_i} \Pr[(A_{m_x} \cap D_{n_2,\ldots,n_L}|B_{x_c} \cap S], \tag{21}$$

where $0 \leq n_i \leq s_i$ for $2 \leq i \leq L$. Note the additional condition of E_α in (21) compared to (16).

The value of α depends largely on how the γ symbols of the secret link key information are encoded into the k symbols. Therefore, it depends on the selection of function f in the scheme.

5.4 Transmission Overhead

In this section, we reuse some of the analysis in Section 5.3 to evaluate the transmission overhead (cost) of the JERT and the SP schemes. We assume that all compromised nodes modify the symbols passing through in our analysis in this section.

For the SP scheme, only the $\frac{m_x}{m}$ term in (18) needs to be modified in order to derive the transmission overhead. When there are m_x out of m paths that are compromised, the chance of successful transmission in the first round is $\frac{m-m_x}{m}$. The chance of successful transmission in the second round is $\frac{m-m_x}{m-1} \cdot \frac{m_x}{m}$ (recall that the sender randomly

9. We use a different variable from e in order to distinguish the two different kinds of compromises: information modification and information disclosure.

picks a path other than the failed path). The process continues until either the transmission becomes successful or all paths are found to be compromised. The expected extra number of transmitted symbols, given m and m_x, is then

$$\begin{aligned}\delta^{(SP)}(m,m_x) &= [1-\mathbf{1}(m_x<m)]\cdot\gamma\cdot(m-1)\\ &\quad+\mathbf{1}(m_x<m)\cdot\gamma\\ &\quad\cdot\sum_{i=1}^{m_x+1}\left[(i-1)\cdot\frac{m-m_x}{m-(i-1)}\cdot\prod_{j=0}^{j=i-2}\frac{m_x-j}{m-j}\right],\end{aligned} \tag{22}$$

where $\mathbf{1}(m_x<m)$ returns 1 when the condition $m_x<m$ is true; otherwise, it returns 0.

Thus, the expected number of transmitted symbols of the SP scheme is

$$\begin{aligned}\delta^{(SP)} &= \gamma+\sum_{S}\Pr[P_{s_2,s_3,\dots,s_L}]\Bigg\{\sum_{x_c=1}^{n_T}\binom{n_T}{x_c}x^{x_c}(1-x)^{n_T-x_c}\\ &\quad\cdot\left[\sum_{m_x=1}^{m}\Pr[A_{m_x}|B_{x_c}\cap S]\delta^{(SP)}(m,m_x)\right]\Bigg\},\end{aligned} \tag{23}$$

where n_T is given by (14), $\Pr[A_{m_x}|B_{x_c}\cap S]$ is given by (16), and $\delta^{(SP)}(m,m_x)$ is given by (22).

For the JERT scheme, a failed ith, $1\le i<e$, transmission leads to additional r_i symbols to be transmitted. In order to simplify our analysis, we assume that when a path is compromised, the average number of symbols is modified (instead of the q_j fraction of symbols transmitted through the path in the current round). Such a simplification makes our analysis tractable while maintaining its validity.

When there are m_x paths compromised, the JERT scheme sends a total of $\sum_{i=0}^{\min(m_x,e)}r_i$ symbols. Therefore, the expected extra number of transmitted symbols of the JERT scheme, given that m_x out of m paths are compromised, can be expressed as

$$\delta^{(JERT)}(m,m_x)=\sum_{i=1}^{\min(m_x,e)}r_i, \tag{24}$$

where e is determined in (13).

The expected number of transmitted symbols of the JERT scheme is

$$\begin{aligned}\delta^{(JERT)} &= r_0+\sum_{S}\Pr[P_{s_2,s_3,\dots,s_L}]\Bigg\{\sum_{x_c=1}^{n_T}\binom{n_T}{x_c}x^{x_c}(1-x)^{n_T-x_c}\\ &\quad\left[\sum_{m_x=1}^{m}\Pr[A_{m_x}|B_{x_c}\cap S]\delta^{(JERT)}(m,m_x)\right]\Bigg\}.\end{aligned} \tag{25}$$

5.5 Computation Cost

In the proposed JERT scheme, punctured MDS codes are used to transmit extra symbols. Thus, the error-erasure decoding algorithm must be implemented in order to decode the received vector efficiently [21], [26], [27]. Two extra decoding steps are needed for error-erasure decoding compared with the error-only decoding: the calculation of the erasure-locator polynomial and the computation of the Forney syndrome polynomial. For punctured codes, the calculation of the erasure-locator polynomial can be performed in advance such that no computation is needed. The Forney syndrome is obtained by multiplying the erasure-locator polynomial by the syndrome and then modularizing it by x^{2t}. For an (n',k) punctured MDS code with (n,k) mother code, the computation takes about $(n-n')(n+n'-2k-3)/2$ multiplications and additions. Determining the values of errors via the Forney formula is now more complex. One additional step is the error-locator polynomial multiplying the erasure-locator polynomial. The computations of the whole procedure to determine the values of errors is roughly $3(n-k+n-n')(n'-k)/8+(n-n')(n'-k)/2$ additions and multiplications. In total, for the punctured code, the total number of calculations is then

$$\begin{aligned}&4\Bigg[(n'-1)\frac{(n-k)}{2}+(n-n')\frac{(n+n'-2k-3)}{2}\\ &\quad+2\left(\frac{(n'-k)}{2}\right)^2+n'\frac{(n'-k)}{2}\\ &\quad+3(n-k+n-n')\frac{(n'-k)}{8}+(n-n')\frac{(n'-k)}{2}\Bigg]\\ &\approx[2n(n-k)+2(n-n')(n'-k)+4(n-k)(n'-k)].\end{aligned} \tag{26}$$

In some applications, the available memory could be limited. Then, a direct software implementation on additions and multiplications is preferred. The complexity of additions is to take bitwise exclusive-ORs on two τ-bit vectors, and performing a multiplication is equivalent to performing τ additions. Thus, the total number of calculations is then

$$(\tau+1)\left[\frac{n(n-k)}{2}+\frac{(n-n')(n'-k)}{2}+(n-k)(n'-k)\right].$$

It has been observed that the energy cost of transmitting 1 Kbit to a distance of 100 m is approximately 3 J. By contrast, a general-purpose processor with 100-MIPS/W power could efficiently execute 3 million instructions for the same amount of energy [28]. Based on these numbers, we can convert the extra computation cost of the JERT scheme into equivalent symbol transmissions. Assume that the JERT scheme performs $t_p<e$ round of extra transmissions. Using (26), we calculate the extra computation cost as equivalent to

$$\begin{aligned}&\frac{10^{-3}(\tau+1)}{3\tau}\sum_{i=1}^{t_p}\Bigg[\frac{n(n-k)}{2}\\ &\quad+\frac{(n-n')(n'-k)}{2}+(n-k)(n'-k)\Bigg]\end{aligned} \tag{27}$$

extra symbol transmissions, where $n'=\sum_{j=0}^{i}r_j$, and r_j is the number of symbols of the jth round of extra transmission by the JERT scheme.

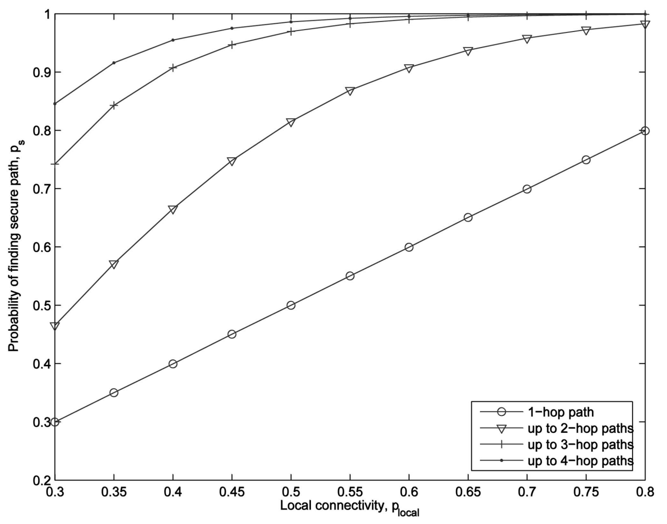


Fig. 3. Probability of finding secure paths between two physical neighbors with up to one-hop, two-hop, three-hop, and four-hop paths.

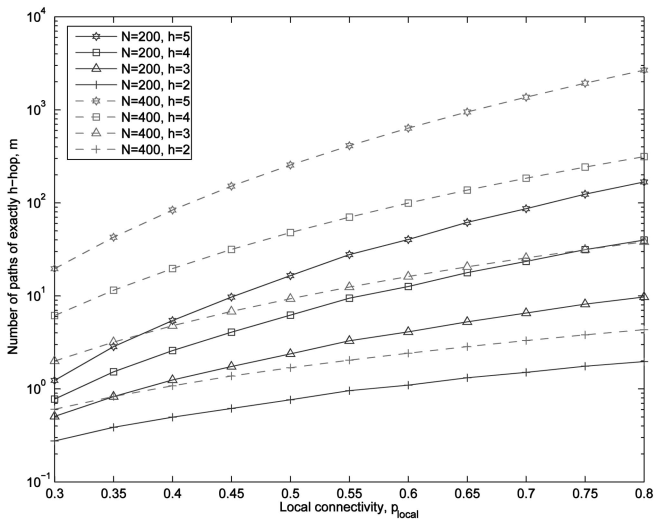


Fig. 4. Number of paths with exactly h hops between the source and the destination.

6 Performance Evaluation

Simulations have been performed in Matlab to evaluate the efficiency of the proposed scheme. Unless specified otherwise, our simulations were set up with the following parameters: We randomly place $N = 400$ nodes on a square area of 1,000 m by 1,000 m. The radio transceiver range is 100 m. The MDS code is assumed to be $(n, k) = (1{,}023{,}255)$. We investigate the performance of the JERT scheme and other related schemes. These schemes include the Incremental Redundancy Transmission (IRT) scheme,[10] the SP scheme, the H-M scheme proposed by Huang and Mehdi [13], and the scheme proposed by Chan et al. [2]. In our discussions, we vary γ for the SP scheme instead of changing (n, k) for the JERT scheme.

6.1 Path Availabilities

In Fig. 3, we demonstrate the need for multihop paths to connect two physical neighbors. In this figure, we present the probability of connecting two physical neighbors p_s with up to one-hop, two-hop, three-hop, and four-hop paths for increasing the local connectivity p_{local}. When we only include one-hop paths, $p_s = p_{local}$. As we include paths with more hops, the connectivity probability increases and approaches 1 when p_{local} increases. For example, when $p_{local} = 0.5$, up to two-hop paths can connect about 80 percent of the physical neighbors. However, we can connect about 97 percent and 99 percent of the physical neighbors when we use up to three-hop and four-hop paths, respectively.

In Fig. 4, we show the number of paths with secure connections that are exactly h hops from a source to a destination (assuming that they do not share a common key). The average number of paths is presented, corresponding to various p_{local}. We also present the number of paths for a similar network with half the nodes $(N = 200)$ for comparison purposes. As shown in Fig. 4, the number of available paths increases with the local connectivity p_{local}. When the node density increases, there are more paths as well. The number of h-hop paths also increases with h. Note that these paths may have common nodes other than the source and the destination.

We show the number of node-disjoint paths in Fig. 5. Note that we chose two-hop paths first and then eliminated all other paths with nodes that have appeared in the previously counted paths. The total available node-disjoint paths are rather limited, as shown in Fig. 5. One interesting observation based on Fig. 5 is that the number of exactly h-hop node-disjoint paths decreases as h increases after 3. This could be due to our path selection process and the larger number of nodes needed in h-hop node-disjoint paths in the neighborhood of the source and the destination when h is larger.

10. The IRT scheme is an early version of the JERT scheme [29]. The IRT scheme uses all two-hop and three-hop paths, including those with common routers. Another major difference between the IRT and the JERT schemes is that the JERT scheme sends different fractions of symbols through different paths, whereas the IRT scheme always sends $q_j = 1/m$ for all available paths.

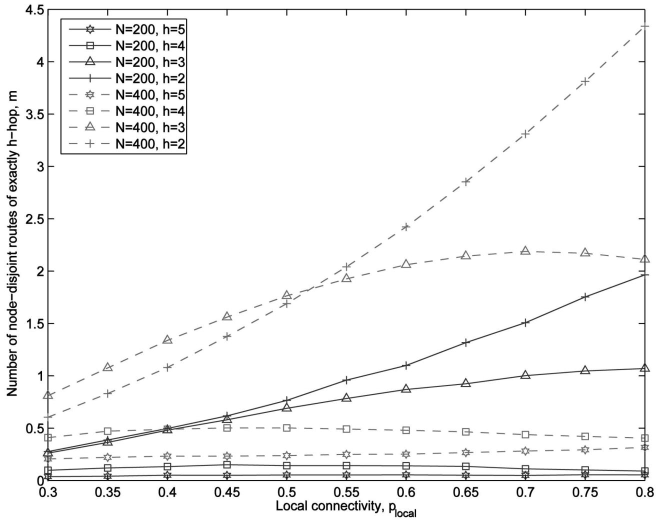


Fig. 5. Number of node-disjoint paths with exactly h hops between the source and the destination.

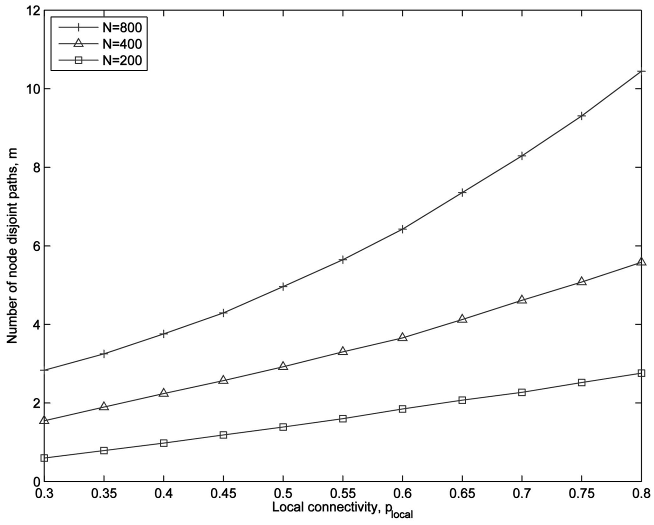


Fig. 6. Total number of node-disjoint routes.

The total number of node-disjoint routes from a source toward a destination is shown in Fig. 6. Based on this figure, it can be concluded that the available node-disjoint paths increases with p_{local} and N. An example of these results is that when $p_{local} = 0.6$ and $N = 800$, there are roughly six node-disjoint paths. Note that the JERT scheme uses all the available node-disjoint routes from a source toward a destination. When the number of such routes is larger, the JERT scheme can tolerate more compromised routes.

In order to gain more insight on the neighbors of the source node serving in the node-disjoint multihop secure paths, we investigated the probability of secure neighbors serving in the transmission paths P_t and showed the results in Fig. 7. Since the insecure neighbors will not serve in the transmission paths, we only count the secure neighbors. As shown in Fig. 7, P_t increases with p_{local}. This can be explained by the better chances of finding other secure neighbors toward the destination. As N increases, there are more nodes in a neighborhood as well, leading to an increasing P_t. Therefore, under normal circumstances, the serving probability is from 0.4 to 0.7, depending on the node density and p_{local}. When nodes have a relatively large number of secure neighbors, they will be able to find enough node-disjoint multihop paths.

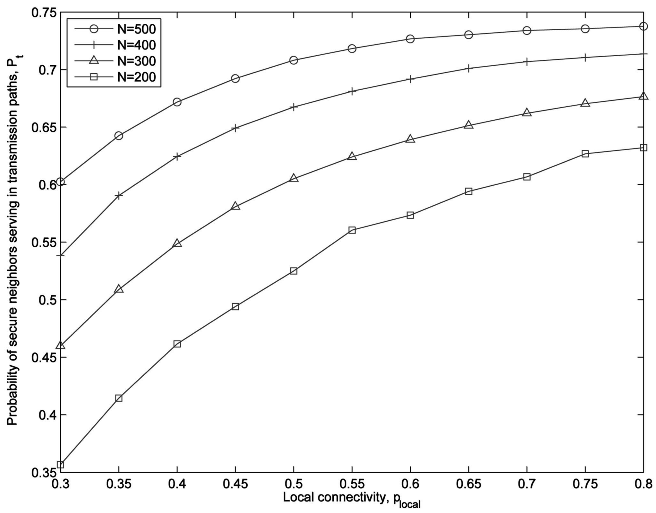


Fig. 7. Probability of secure neighbors serving in the node-disjoint transmission paths.

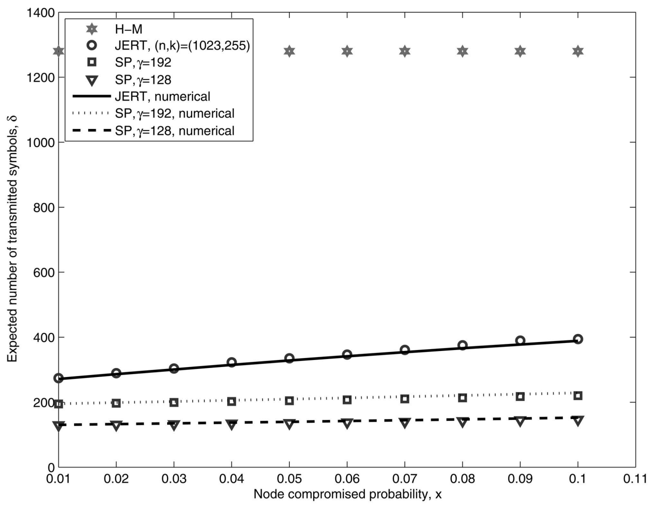


Fig. 8. Transmission overhead of the JERT, SP, and H-M [13] schemes. Numerical results based on (22) and (25) match well with our simulation results.

6.2 Transmission Overhead

When there are compromised nodes on the paths used to deliver the secret link key information and these compromised nodes modify the passing information, extra symbols need to be transmitted. Fig. 8 shows the expected number of symbols that need to be transmitted in order to allow the destination to regenerate the secret key. We use all the available node-disjoint paths in the JERT scheme. The SP scheme randomly chooses one out of these paths to send the secret link key. The number of transmitted symbols in the SP scheme lowers as γ decreases. Therefore, the relative transmission cost of the JERT scheme, as compared with the SP scheme, increases as γ decreases. Note that there is a slight increase in the number of transmitted symbols in the JERT and the SP schemes as the node-compromised probability x increases. This is due to the higher probability of the used paths being compromised, and retransmissions may be needed.

For comparison purposes, the number of symbols that are transmitted in the H-M scheme [13] is also shown in Fig. 8. Since an MDS code of $(n, k) = (1{,}023{,}255)$ was used, the H-M scheme sends $255 \times (4 + 1) = 1{,}275$ symbols while only tolerating up to two compromised paths. The JERT scheme with (1,023,255) code, however, can tolerate up to five compromised paths when the total number of paths is 12. Although it ensures the detection and correction of up to a higher number of faulty paths, the H-M scheme operates with a higher transmission cost. Note that the difference in cost of the JERT and H-M schemes changes with the selection of n and k.

Numerical results based on (22) and (25) are also presented in Fig. 8. These results match well with our simulation results.

In Fig. 9, we compare the extra symbol transmission of the JERT scheme and a related scheme suggested by

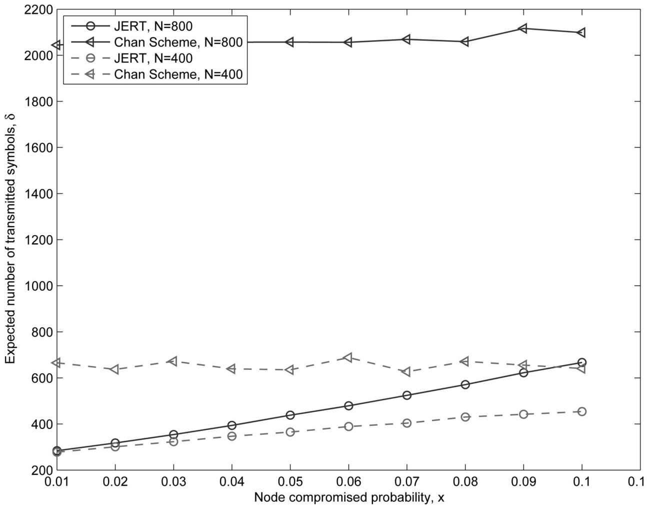


Fig. 9. Transmission overhead of the JERT scheme and a related scheme from [2]. The additional computation of the JERT scheme has been converted to an equivalent symbol transmission and included in this figure.

Chan et al. [2]. In this scheme, a reinforcement technique is used to make sure that the adversary needs to compromise all paths in order to obtain the key. For fair comparison, the extra symbol transmission of the scheme is calculated based on sending keys through $e + 1$ paths. Note that JERT performs additional transmission when it is necessary such that the computational cost of JERT depends on the number of compromised paths between the source and the destination. In this figure, the extra computational cost of JERT has been converted to equivalent extra symbol transmissions and added to the results (see (27)). According to this figure, the extra symbol transmission of the JERT scheme is much smaller than that of the scheme given in [2].

6.3 Security Performance

The flexibility of the secret-disclosure probability of the JERT scheme is presented in Fig. 10. In this figure, we vary the value of α and show the secret-disclosure probability p_x for different node-compromised probabilities x. It can be observed that the JERT scheme has a much lower p_x than the SP scheme when $\alpha < 1$. As α increases within the range between 0 and 1, the p_x value is smaller. For a fixed γ, α may be lowered by increasing k and n. Therefore, the JERT scheme provides a nice property of flexibility: a predefined threshold of the probability of secret key disclosure can be guaranteed by varying (n, k). The numerical results of (18) and (19) are compared with the simulation results in Fig. 10 as well. The numerical results (curves) match with the simulation results (symbols) quite well.

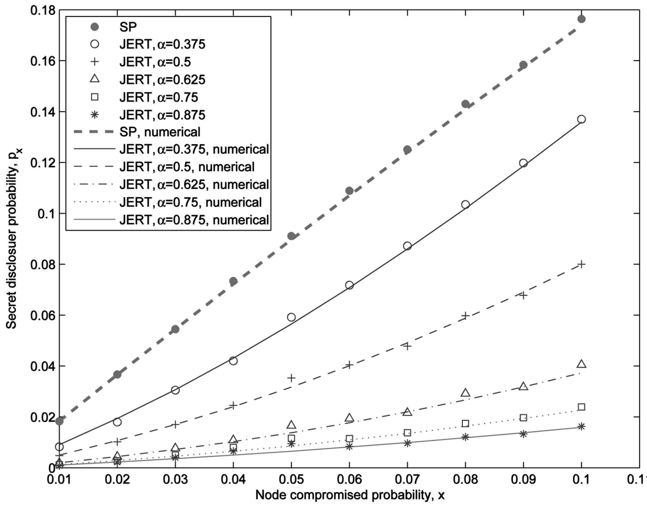


Fig. 10. The flexible secret-disclosure probability of the JERT scheme $(N = 400, p_{local} = 0.5)$. Numerical results are compared with simulation results. All cases with $m \geq 1$ paths are studied.

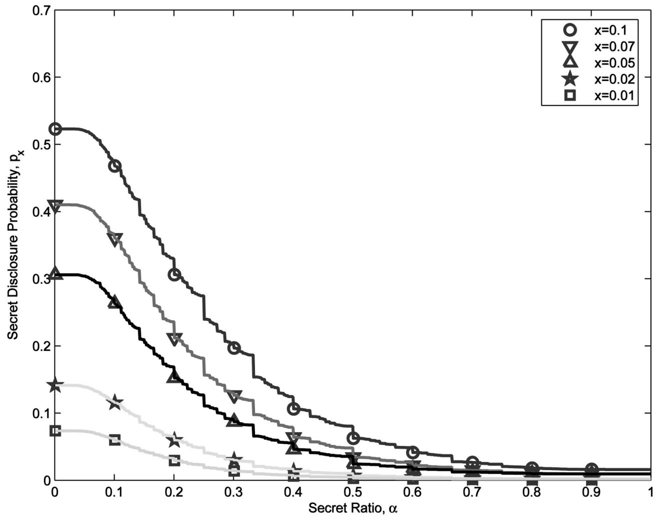


Fig. 11. The secret-disclosure probability as a function of secret ratio α for different compromised probabilities x for the JERT scheme.

We present the secret disclosure probability from another angle in Fig. 11, where we show p_x as a function of α for different x. The stepped curves shown in Fig. 11 suggest that p_x has some abrupt changes when α varies. This is because each path sends an integer number of symbols, and when one path is compromised, all these symbols are disclosed. Thus, the ratio of disclosed symbols is not continuous when α varies. When this value is compared with a secret ratio, step functions may appear.

In Fig. 12, we compare the expected number of transmitted symbols in the JERT scheme, the IRT scheme, and another related scheme termed JERTe. The JERTe scheme is similar to the JERT scheme, except that $q_j = 1/m$ for all paths. The performance of the JERTe scheme is presented to show the effectiveness of the technique of sending different amounts of symbols to paths of different lengths. Based on Fig. 12, we conclude that the JERT scheme outperforms the IRT and JERTe schemes with much lower transmission cost. We can further conclude that the selection of q_j has major effects on transmission overhead in the JERT scheme.

We present the secret-disclosure probability of the JERT, IRT, and JERTe schemes in Fig. 13. In order to distinguish the performance of different schemes, we artificially increased γ to 128 and 192. Such values of γ result in higher secret-disclosure probabilities. Based on Fig. 13, it can be concluded that the JERT scheme outperforms the IRT and JERTe schemes with lower secret-disclosure probability.

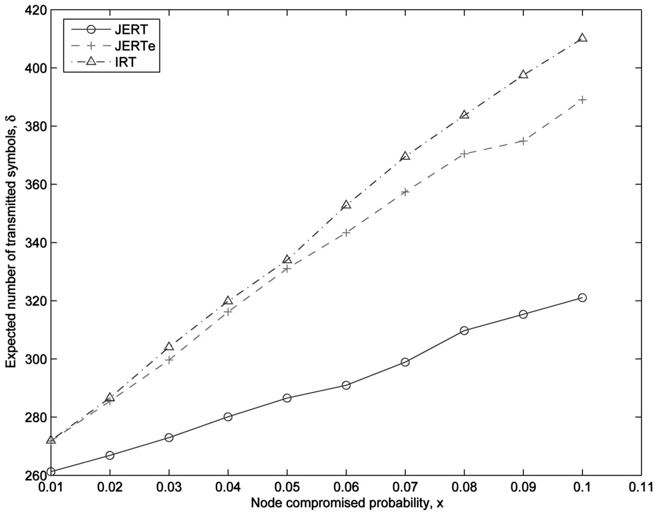


Fig. 12. Comparing the expected number of transmitted symbols in the JERT scheme and related schemes, $(n, k) = (1{,}023{,}255)$.

7 Conclusions

Key predistribution techniques for security provision of WSNs have attracted significant interests recently. One class of such key predistribution schemes loads a relatively small number of keys randomly chosen from a large key pool prior to deployment. After being deployed, each sensor tries finding common keys with its direct neighbors to establish link keys. Such link keys will then be used to protect the wireless communication between themselves. Due to the randomness of the key selection process, some neighboring sensors do not share any common key. In order to establish a link key among such neighbors, a multihop secure path may be used to deliver the secret. Such a delivery technique, however, renders several security problems such as secret information being disclosed and secret information being modified or dropped when some sensors are compromised by the adversary.

In this paper, we have proposed and investigated the JERT scheme for the secret link key establishment process of key predistribution techniques. The JERT scheme uses the powerful MDS codes to encode the secret link key information and send the code words through multiple multihop paths. It provides a nice "Just-Enough" property: the redundant symbols are transmitted only when they are needed to enable the destination to decode the secret information. This feature reduces the transmission cost and provides extra protection of the secret information against disclosure. Furthermore, the JERT scheme has the salient feature of its flexibility of trading transmission for lower information disclosure.

Acknowledgments

J. Deng's work was supported in part by the Louisiana Board of Regents under RCS Grant LEQSF (2005-08)-RD-A-43. Y.S. Han's work was supported in part by the National Science Council of Taiwan under Grant NSC 95-2221-E-305-003. The authors would like to express their special thanks to the associate editor Professor V. Gligor and the three anonymous reviewers for their valuable comments. An early version of this paper was presented at the International Conference on Mobile Ad-hoc and Sensor Networks (MSN '05), Wuhan, China, December 2005.

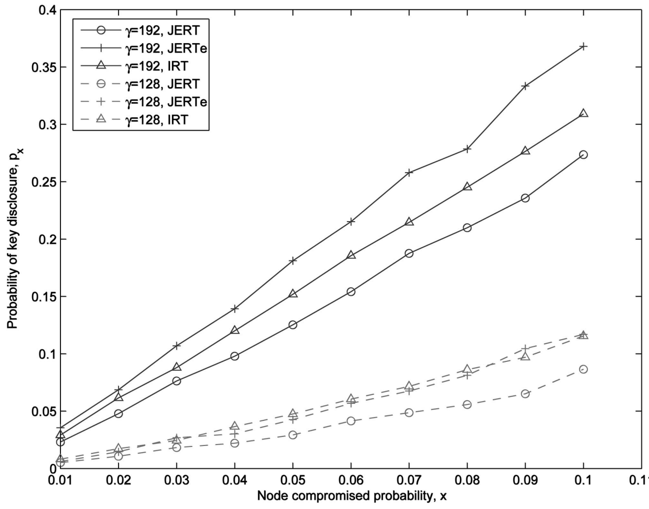


Fig. 13. Comparing the secret-disclosure probability of the JERT scheme and related schemes, with $(n, k) = (1{,}023{,}255)$ and different γ

References

[1] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02),* pp. 41-47, Nov. 2002.
[2] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy (S&P '03),* pp. 197-213, May 2003.
[3] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* pp. 42-51, Oct. 2003.
[4] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* pp. 52-61, Oct. 2003.
[5] W. Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, and A. Khalili, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *ACM Trans. Information and System Security,* vol. 8, no. 2, pp. 228-258, May 2005.
[6] D. Liu, P. Ning, and R. Li, "Establishing Pairwise Keys in Distributed Sensor Networks," *ACM Trans. Information and System Security,* vol. 8, no. 1, pp. 41-77, Feb. 2005.
[7] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Comm.,* vol. 1, no. 4, pp. 660-670, Oct. 2002.
[8] S.A. Camtepe and B. Yener, "Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks," *Proc. Ninth European Symp. Research Computer Security (ESORICS '04),* pp. 293-308, 2004.
[9] J. Lee and D.R. Stinson, "A Combinatorial Approach to Key Predistribution for Distributed Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '05),* Mar. 2005.
[10] P. Papadimitratos and Z.J. Haas, "Secure Message Transmission in Mobile Ad Hoc Networks," *Elsevier Ad Hoc Networks,* vol. 1, no. 1, pp. 193-209, July 2003.
[11] A. Tsirigos and Z.J. Haas, "Multipath Routing in the Presence of Frequent Topological Changes," *IEEE Comm. Magazine,* pp. 132-138, Nov. 2001.
[12] M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," *J. ACM,* vol. 36, no. 2, pp. 335-348, Apr. 1989.
[13] D. Huang and D. Medhi, "A Byzantine Resilient Multi-Path Key Establishment Scheme and Its Robustness Analysis for Sensor Networks," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '05),* p. 240, Apr. 2005.

[14] A. Shamir, "How to Share a Secret," *Comm. ACM,* vol. 22, no. 11, pp. 612-613, Nov. 1979.

[15] R.J. McEliece and D.V. Sarwate, "On Sharing Secrets and Reed-Solomon Codes," *Comm. ACM,* vol. 24, no. 9, pp. 583-584, Sept. 1981.

[16] M.B. Pursley and S.D. Sandberg, "Incremental-Redundancy Transmission for Meteor-Burst Communications," *IEEE Trans. Comm.,* vol. 39, no. 5, pp. 689-702, May 1991.

[17] S.B. Wicker and M.J. Bartz, "Type-II Hybrid-ARQ Protocols Using Punctured MDS Codes," *IEEE Trans. Comm.,* vol. 42, nos. 2-4, pp. 1431-1440, Feb.-Apr. 1994.

[18] W. Du, J. Deng, Y.S. Han, S. Chen, and P.K. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. IEEE INFOCOM '04,* pp. 586-597, Mar. 2004.

[19] S.B. Wicker and V.K. Bhargava, *Reed-Solomon Codes and Their Applications.* IEEE Press, 1994.

[20] I.S. Reed and X. Chen, *Error-Control Coding for Data Networks.* Kluwer Academic, 1999.

[21] S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications,* second ed. Prentice Hall, 2004.

[22] G.C. Clark, Jr. and J.B. Cain, *Error-Correction Coding for Digital Comm.* Plenum Press, 1981.

[23] W. Lou and Y. Fang, "A Multipath Routing Approach for Secure Data Delivery," *Proc. IEEE Military Comm. Conf. (MILCOM '01),* pp. 1467-1473, Oct. 2001.

[24] P. Papadimitratos, Z.J. Haas, and E.G. Sirer, "Path Set Selection in Mobile Ad Hoc Networks," *Proc. ACM MobiHoc '02,* pp. 1-11, June 2002.

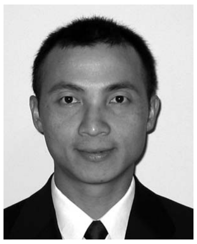
[25] P. Papadimitratos and Z.J. Haas, "Secure Data Transmission in Mobile Ad Hoc Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 2, pp. 343-356, Feb. 2006.

[26] T.K. Truong, W.L. Eastman, I.S. Reed, and I.S. Hsu, "Simplified Procedure for Correcting Both Errors and Erasures of Reed-Solomon Code Using Euclidean Algorithm," *IEE Proc.,* vol. 135, no. 6, pp. 318-324, Nov. 1988.

[27] S.-L. Shieh, S.-G. Lee, and W.-H. Sheen, "A Low-Latency Decoder for Punctured/Shortened Reed-Solomon Codes," *Proc. 16th IEEE Int'l Symp. Personal, Indoor and Mobile Radio Comm. (PIMRC '05),* pp. 2547-2551, Sept. 2005.

[28] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors," *Comm. ACM,* vol. 43, no. 5, pp. 51-58, May 2000.

[29] J. Deng and Y.S. Han, "Using MDS Codes for the Key Establishment of Wireless Sensor Networks," *Proc. Int'l Conf. Mobile Ad Hoc and Sensor Networks (MSN '05),* pp. 732-744, Dec. 2005.



Jing Deng received the BE and ME degrees in electronic engineering from Tsinghua University, Beijing, in 1994 and 1997, respectively, and the PhD degree in electrical and computer engineering from Cornell University, Ithaca, New York, in 2002. From 2002 to 2004, he was with the CASE Center and the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, New York, as a research assistant professor, supported by the Syracuse University Prototypical Research in Information Assurance (SUPRIA) Program. He was a teaching assistant from 1998 to 1999 and a research assistant from 1999 to 2002 in the School of Electrical and Computer Engineering, Cornell University. He is currently an assistant professor in the Department of Computer Science, University of New Orleans. He is an associate editor for the *International Journal of Mobile Communications, Networks, and Computing* (JMCNC). He was a cochair of the IEEE International Workshop on Ad Hoc and Ubiquitous Computing (AHUC 2006) and the sponsorship chair of the First International Conference on Multimedia Services Access Networks (MSAN 2005). He has served on the technical program committees of several IEEE conferences, including the Second IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2005), MASS 2006, 2006 IEEE Global Telecommunications Conference (GLOBECOM), and 2007 IEEE Wireless Communications and Networking Conference (WCNC). His research interests include mobile ad hoc networks, wireless sensor networks, wireless network security, energy efficient wireless networks, and information assurance. He is a member of the IEEE, the IEEE Computer Society, the IEEE Communications Society, and the ACM.

Yunghsiang S. Han received the BS and MS degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and the PhD degree from Syracuse University, Syracuse, New York, in 1993. From 1993 to 1997, he was an associate professor in the Department of Electronic Engineering, Hua Fan College of Humanities and Technology, Taipei. From 1997 to 2004, he was with the Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan, where he was promoted as a full professor in 1998. From June to October 2001, he was a visiting scholar in the Department of Electrical Engineering, University of Hawaii, Manoa. From September 2002 to January 2004, he was a SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and the CASE Center, Syracuse University, New York. He is currently with the Graduate Institute of Communication Engineering, National Taipei University, Taipei. His research interests include wireless networks, security, and error-control coding. He is the recipient of the 1994 Syracuse University Doctoral Prize. He is a senior member of the IEEE.