SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:431–445
Published online 31 March 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.991

RESEARCH ARTICLE

Key splitting: making random key distribution schemes resistant against node capture

Mohammad Ehdaie[1]*, Nikos Alexiou[3], Mahmoud Ahmadian Attari[1], Mohammad Reza Aref[2] and Panos Papadimitratos[3]

[1] CCL, Department of Electrical and Computer Engineering, K.N.Toosi University of Technology, Tehran, Iran
[2] ISSL, Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
[3] School of Electrical Engineering, KTH, Stockholm, Sweden

ABSTRACT

A large number of random key pre-distribution (RKD) schemes have been proposed in the literature to secure wireless sensor network applications, relying on symmetric key cryptography. However, sensor nodes are exposed to physical compromise by adversaries, who target the symmetric keys stored at each node. With the stolen keys in their possession, the adversaries are then able to compromise communication links between benign nodes. Here, the big challenge arises: how to increase resilience of RKD schemes for wireless sensor networks to node capture, while maintaining the flexibility and low-cost features of RKD? We propose the idea of key splitting to address this problem, without the need of any special-purpose hardware. Our key splitting scheme neither increases per-node storage nor introduces additional computation and communication overheads. Nevertheless, it can achieve better connectivity. More importantly, it significantly increases resilience to node compromise, when the adversary does not have overwhelming computational power. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
wireless sensor networks; random key distribution; node capture attack; key splitting

***Correspondence**
Mohammad Ehdaie, CCL, Department of Electrical and Computer Engineering, K.N.Toosi University of Technology, Tehran, Iran.
E-mail: mohammad@ehdaie.com

1. INTRODUCTION

Wireless sensor networks (WSNs) support a wide range of applications, including monitoring for industrial processes, environmental pollution, and agriculture as well as home automation, health care, traffic control, and forest fire detection. However, WSN nodes have restricted computing and storage capabilities, and operate on a stringent energy budget. Given these limitations, traditional security mechanisms are not applicable, rendering WSNs vulnerable to numerous attacks.

A large gamut of security schemes for WSNs has been proposed in the literature. Asymmetric key cryptography, while feasible [1], comes at a high cost (computation, communication overhead, and thus power) [2]. Public key cryptography is limited to infrequent operations (e.g., [3]), or it is precluded altogether. Given the resource constraints of the sensor nodes, symmetric key cryptography is widely accepted to secure sensor networks. Each key shared by a pair of nodes is used to secure their communication. Keys can be shared not only by neighboring (i.e., in direct communication) but also physically remote nodes. In both cases, they can be used to establish security associations and secure links, as well as authenticate nodes, and have been used in different ways to secure the WSN protocols.

However, sensor nodes are low-cost small-footprint platforms. This means it is impossible to offer tamper-resistant features [4] and build security schemes upon this requirement. Moreover, sensors operate unattended [5] and exposed to adversaries. As a result, nodes may be physically compromised: an adversary can extract the symmetric keys stored at each sensor and then use the captured keys to attack the WSN protocols. The compromised symmetric keys allow the adversaries to masquerade as legitimate nodes and inject arbitrary messages in the network that are still authenticated.

The impact of capturing a WSN node depends on the way keys are managed and used by sensor nodes. Each sensor stores multiple symmetric keys. One option is to make a centralized assignment that prescribes exactly the keys each node can use for specific peers (other nodes) [6,7]. In that case, if each key is used only for one link, that is, one pair of nodes, a captured node can at most misbehave to its *a priori* associated peers.

On the other hand, a looser yet more flexible and scalable approach can be used. Nodes are first given a randomly chosen set of keys (called the *key ring*) from a large pool. Then they contact their neighboring peers to discover shared symmetric keys (if any), and finally, the shared keys are used to generate a secret key and securely communicate with the related peer [8,9]. Random key pre-distribution (RKD) schemes support larger WSN networks compared with the aforementioned pairwise schemes [6,7], an important characteristic, along with simplicity and flexibility, for many sensor applications. The flip side of such schemes is their higher vulnerability to node capture. A captured node allows attacks against its associated peers. More important, for longer node capturing history, more keys are revealed to the adversary, who is now able to compromise additional links (between non-compromised nodes) throughout the network. Intuitively, the more numerous the captured nodes are the more likely the compromise of other secure links in the network.

Given the advantages of RKD, *is it possible to enhance its resilience to node capture?* Can we do this without altering its salient features, reducing its efficiency, or without special purpose tamper-resistant hardware? This work addresses the previous questions. We propose a simple yet very effective solution we term *key splitting* [10]. Our solution is applicable and can be extended for any RKD scheme.

In a nutshell, rather than using key rings, we use rings of key parts, which can be put together to form entire keys. Rather than looking for matching keys, any two peers search for a sufficient number of key parts; if they have such key parts in common, they establish a secure link using a transformation of them as the shared key. As our analysis shows, the improvement in resilience is significant. The probability of link compromise we achieve can be up to half of that for existing widely used schemes. At the same time, we maintain the same storage requirements: simply put, rather than keeping a ring of m keys, we keep a ring of $m \times z$ key parts each one being $1/z$-th of original key in size. Moreover, the connectivity, that is, the likelihood that two nodes can be securely associated, can be enhanced.

In the rest of the paper, we first briefly define our system, the adversary models, and the problem at hand (Section 2). We discuss related work in Section 3, and then we present our key splitting scheme in Section 4. A detailed analysis of its resilience is performed in Section 5. In Section 6, we conclude the paper.

2. SYSTEM MODEL AND PROBLEM STATEMENT

2.1. System model

Each WSN node is assigned a set of m keys termed the *key ring*. Keys are randomly chosen from a *key pool*, of size $|P|$, and they are stored at each sensor before its deployment. At any point in time, two sensors can run a key discovery protocol (KDP) to discover their common keys, in order to then establish a secure communication link. Consider two sensors that wish to communicate securely and run the KDP. In the basic RKD approach [8], only one common key is required to establish a secure connection. In a widely referenced variant of the basic scheme, the so-called q-composite RKD approach, the nodes have to share (and discover) at least q common keys in order to establish a secure connection [9]. For the rest of the paper, we consider both the basic and the q-composite RKD schemes and compare our scheme with them.

2.2. Adversary model

We assume an adversary, *Adv*, that can capture a number of wireless sensors and thus construct a set of compromised key rings. A secure link between a pair of sensors can be compromised if the gamut of captured keys by *Adv* includes all those used to establish the link. If *Adv* wants to compromise the link between two nodes using the basic scheme, the shared key between them should be included in the set of captured keys. If the nodes use the q-composite scheme, then all the $x \geq q$ keys shared between them need to be stolen to compromise the link.

Consider an adversary who is able to capture nodes at special rate. We define the average required time for capturing a node as a time unit. Besides capturing nodes, we assume a brute-force power (BFP) for *Adv*. Let us introduce a new parameter to address *Adv*'s capability in running a brute-force attack. We say that the *BFP* of *Adv* is equal to *bfp* if it is possible for *Adv* to search (and run the required operations in) all the 2^{bfp} space in a time unit. For example, if we say that $bfp = 32$, it means that *Adv* can search all possible 32-bit numbers in a time unit. Equivalently, *Adv* can search all possible 31-bit numbers in a half of a time unit or search all possible 40-bit numbers in 2^8 time units. We discuss more about this ability in Section 5.

2.3. Problem statement

An adversary, *Adv*, compromises a number of nodes and obtains their key rings. Links between benign nodes are vulnerable if their corresponding shared keys are included in the captured set of keys. As a result of the node compromise, a fraction α of the WSN links between benign and non-compromised nodes is compromised (fully compromised links), that is, *Adv* possesses the corresponding shared key(s). Alternatively, as we will see in our scheme, *Adv* knows for another fraction, β, of the WSN links, a part of the key(s) shared and used by the non-compromised nodes (partially compromised links). In the latter case, *Adv* can attempt to compute the missing part(s) and compromise the link.

Overall, the challenge is how to enhance RKD resilience against node capture, in particular how to reduce link compromise. Given the aforementioned defined adversary attacking over a period $[0, t]$, we assume that there is

DOI: 10.1002/sec

no reparative action by the WSN. Thus, $s = 1 \times t = t$ nodes are captured, and their keys are compromised during that period (time unit = required time for capturing one node). In the same time, *Adv* tries to guess non-compromised part(s) of keys with trial and error. On the basis of *Adv*'s BFP, all or a fraction of partially compromised links in the given time interval can be broken. We want to *minimize* the total fraction of compromised links at time t.

On the basis of the security properties of the network, *Adv* may be interested in compromising as many links as possible in a given time interval or even aim to break a desired fraction of links in a reasonable time. In this case, our goal is to maximize the required time for breaking this number of links. In Section 5, we use both of these models to compare different schemes with each other.

Adversarial benefit parameter: We now introduce a new parameter called the adversarial benefit parameter *AdvBen*. We will use this parameter as a measure of cumulative benefit for the adversary during the attack.

Our main concern for the analysis is the fraction of network links compromised by *Adv*. Consider a constant rate of node capture. As the attack unfolds in time, the adversary captures more nodes and is thus capable of compromising additional network links. Given a single point in time $t > 0$, and using our analysis from Section 5, we are able to measure the fraction of compromised links at time t using the $fail_{time}$ function: $fail_{time}(t)|\mathcal{R} \rightarrow [0, 1]$.

$$fail_{time}(t) = \frac{\text{Num. of comp. links at time } t}{\text{Number of all network links}} \quad (1)$$

However, the resilience of any RKD scheme against node compromise cannot be fully captured only at a given point time, through the probability that a given link is compromised. Instead, as the attacker progresses with the attack (compromising an increasing number of nodes and working toward compromising links), what matters is the overall "progress" of the attacker or inversely the "resistance" of the system.

Therefore, we wish to measure the benefit for the adversary to hold a fraction of compromised network links over time until t. To achieve that, we introduce our *AdvBen* parameter. Using *AdvBen*, we are able to directly compare the resilience of RKD schemes against our *Adv*, for t time units of network exposure to the node capturing attack. We compute *AdvBen(t)* as the integral of the fail function, over the time interval $[0, t]$.

Definition 2.1. *For a key distribution scheme and for a given adversarial power settings, the adversarial benefit is defined as the integral of the fail function:*

$$AdvBen(t) = \int_0^t fail_{time}(\tau)d\tau \quad (2)$$

Notation: For easy reference, the notation we use throughout the paper is as follows:

- P: key pool.
- $|P|$: key pool size.
- m: size of the key ring.
- q: the minimum number of shared keys for two nodes to have a secure link.
- L: the length of a key in bits.
- k: a key or a key slice.
- p_c: probability that a pair of nodes share at least q keys (to have a secure link).
- $p(i)$: probability of two nodes sharing exactly i keys.
- z: number of slices of an original key.
- γ: the fraction of broken links the adversary wants to achieve.

3. RELATED WORK

An overview of the large palette of security problems in WSNs is given in [11–13]. Some of the most important security attacks are tampering of sensors and node capture [4], denial of service [14], attacks to secure routing and secure neighbor discovery [15,16], and Sybil attacks [6,17]. Security challenges of sensor networks are unique in nature, because of their limitations in storage, computation, and communication capabilities. Traditional security approaches, such as public key cryptography, are therefore unsuitable for frequent usage [2]. Symmetric key approaches, relying on RKD schemes have been proposed instead, to overcome these limitations [7–9,18].

Each sensor is equipped with a set of symmetric keys called the key ring, randomly chosen from a key pool. RKD schemes define the way that two sensors can establish a common symmetric key, using the overlaps in their key rings. To discover common keys, a KDP has to be run. The main RKD schemes proposed are the basic [8], the q-composite [9], and the random pairwise schemes [7,9,18]. The first two are briefly discussed in Section 2. The random pairwise scheme associates a sensor identity with one key from the sensor's key ring. The corresponding sensors can establish a secure channel using the associated key.

The starting point for many attacks is an adversary who compromises a sensor and obtains its key ring. A node replication attack is described in [19] and a collusion attack against random pairwise schemes in [20]. Tamper-resistant devices can be used to protect against node capture. However, the increased cost of tamper-resistant devices is a limiting parameter for extended usage of such devices [11].

Communication links between sensors may be compromised by adversaries who manage to capture the keys used to set up the secure connection [7,9,18]. Having captured a number of nodes, the adversary can derive the secret keys used for communication between two sensors. RKD schemes play an important role in the resilience against the attack. In the basic scheme, when $m = 200$ keys are stored

per sensor and each sensor can establish a secure channel with $\frac{1}{3}$ of its neighbors ($p_c = 0.33$), an adversary has to capture about 50 nodes to compromise 10% of the links [9]. For the same setup, the 2-composite scheme has better resilience against the attack, and more than 70 nodes have to be captured to compromise the same percentage of links. The random pairwise scheme achieves increased resilience, since at most, m links can be compromised from one captured node. Advanced stealthy strategies from colluding adversaries are studied in [21], where adversarial nodes combine their knowledge of captured keys to maximize their link compromise.

Despite the obvious advantages of the random pairwise scheme, it introduces restrictions in the maximum supportable network size. The total size it can support can be computed as $n = m/p_c$ and is directly constrained by the sensors' limited memory [7,9]. This limitation may render the random pairwise schemes unsuitable for a number of applications. Large and flexible network deployments, as well as frequent addition of new nodes, are better handled by the q-composite or the basic scheme. However, the security vulnerabilities of these schemes bring out new challenges on how to make these schemes more resilient against attacks. In the next section, we merge good properties of the basic scheme and the q-composite scheme, that is, good connectivity and good resilience.

4. THE KEY SPLITTING SCHEME

The key splitting scheme is a method to increase the resilience against node capture attacks. We split each of the keys in the pool into z equal *parts*, creating a new key pool of size $z \times |P|$. We now explain the main intuition behind key splitting by presenting an example of the basic RKD scheme and key splitting. In the next section, we study the effectiveness of the key splitting scheme in detail.

Basic scheme: Consider two nodes A and B with key rings of size m. The two nodes use the basic RKD scheme to establish their common secret. Each sensor key has a length of L bits, and therefore, each sensor needs $m \times L$ bits of storage for its key ring. Figure 1 shows an instance of a KDP run between A and B for the basic scheme. Methods on how to run the KDP are included in the related work, and we do not enter into a detailed discussion here.

After completing the KDP, the two nodes discover that they share k_7. Then they can use k_7 to establish a secure channel. Now, consider the adversary *Adv*; all is needed is a bit of "luck" to capture a node, somewhere in the network, with k_7 stored in its memory. Then it is only a matter of time and a few trials to compromise the link of A and B.

The question we aim to address in this work is the following: "How could the probability of compromising the link between A and B be decreased, without introducing any computational or storage overhead?"

Key splitting: To answer the previous question, we suggest a straightforward and effective method: to *split the keys*. Consider again A and B but in a slightly different setup; each key in P is now split into z equal parts. Each of the new key parts, or let us say *key slices*, will now have a length of L/z bits. Splitting the keys has no storage overhead for the nodes. With the same memory usage at each sensor, $z \times m$ key slices can now be stored, creating a key ring of $z \times m$ key slices. For example, let $z = 2$ and keys with original length of $L = 128$ bits. We split the keys into two parts and now generate (and store) twice as many keys of size 64 bits.

A and B now run the KDP to find their common key slices. To preserve the same security level with a symmetric key of size L, we require that A and B should now discover z key slices of size L/z, instead of just one. Figure 2 illustrates this key splitting scenario. Having discovered k_3 and k_9, A and B can establish a symmetric key of size L, just as in the previous example.

But in the latter key splitting scenario, they have a great advantage over *Adv*, who tries to compromise their link. Instead of having to get one key only, as in the basic scheme, the adversary has to get z (here, $z = 2$) key slices now. Having compromised the same number of nodes in both cases, *Adv* is less likely to hold both secret pieces in the second case than just one. Building on this observation, we analytically prove the effectiveness of the scheme in Section 5 and verify that network connectivity is not deteriorated. This is actually the main advantage of our scheme over the q-composite scheme. In the q-composite scheme, the resilience is increased with the cost of memory or connectivity, that is, it needs more keys to be stored in the key ring or the connectivity is deteriorated; while in the key splitting scheme, the resilience is improved without such cost.

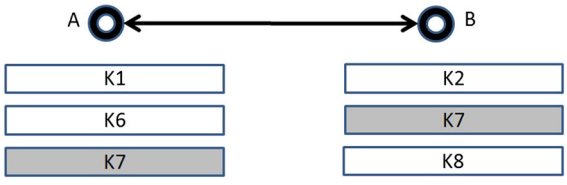

Figure 1. Random key pre-distribution scheme for A and B that discover a shared key of size L.

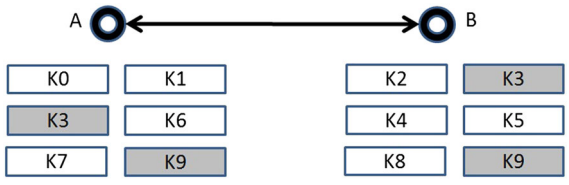

Figure 2. Key splitting scheme for nodes A and B that discover two shared key parts of size $L/2$.

DOI: 10.1002/sec

Definition 4.1. *z-Splitting is an RKD scheme, with key slices of size L/z, a key pool of size $|P| \times z$, and key rings of size $m \times z$, where every pair of nodes that share at least z key slices can establish a secure channel using those slices.*

The steps of the z-splitting scheme are as follows:

- *Key pool generation:* Generate a key pool of $z \times |P|$ key slices, each one of length L/z bits.
- *Key assignment:* $m \times z$ key slices are randomly assigned to each sensor.
- *Link establishment:* Sensors run the KDP in pairs and discover j key slices in common. If $j \geq z$, they use a hash function, h, to derive a secret key as $h(k_1||k_2||...||k_j)$ where || denotes the concatenation and $k_1, k_2, ..., k_j$ are the shared key slices.

4.1. q-composite key splitting

In 2003, Chan *et al.* [9] proposed an improvement over the basic RKD scheme [8]. They suggested increasing the threshold for required number of shared keys to establish a secure channel. This way, they increased the resilience of the scheme against node capture for small scale attacks. We can apply their method on key splitting to improve its resilience, too.

Definition 4.2. *(z,q)-Splitting, with $q \geq z$, is an RKD scheme, with key slices of size L/z, a key pool of size $|P| \times z$, key rings of size $m \times z$, where every pair of nodes that share at least q key slices can establish a secure channel using those slices.*

The steps to establish a link are the same as with z-Splitting, except that in (z,q)-Splitting, the required number of shared key slices is q instead of z.

5. SCHEME ANALYSIS

We analyze the resilience to node capture comparing the key splitting scheme to the basic and q-composite RKD schemes. For easier presentation, we first study the case of a single node/key ring captured and then we extend our analysis for the case of multiple compromised key rings. We review performance issues for key splitting, notably connectivity and memory storage, and propose the best network configuration parameters (q and z) for different security requirements. Finally, we compare key splitting to the rest of RKD schemes using our *AdvBen* parameter and calculate the maximum supportable network sizes.

In brief, we find that our key splitting scheme improves resilience against node capture, as long as the adversary does not have overwhelming computational power. We calculate the probability that the adversary possesses all of the keys or key slices for the q-composite and key splitting schemes, as well as for the basic scheme. Finally, we consider the case of a subset of the key parts for a link being compromised.

5.1. Security analysis

Resilience against node capture is the core of this study. The fraction of compromised secure communication links between pairs of benign nodes is our main metric of interest. This is expressed by the measure $Fail(s)|\mathcal{N} \rightarrow [0, 1]$, where s is the number of compromised nodes or equivalently, key rings. This is the fraction of compromised secure links of a WSN, given that the adversary has *captured s* key rings

$$Fail(s) = \frac{\text{Num. of comp. links given s comp. nodes}}{\text{Number of all network links}} \quad (3)$$

Note here that in contrast to $Fail(s)$, our $fail_{time}(t)$ function from Section 2 calculates the fraction of compromised nodes at a given point in *time*, given that the adversary captures nodes at a given *rate*. For the rest of the paper, we assume that the adversary knows the number of shared keys. This is essentially a worst case for the scheme, or a best case for the adversary, because it removes uncertainty for the adversary.

5.1.1. Simple case.

Consider $Fail(s)$ with $s = 1$. Let $V_1 - V_2$ be a link in the network and V be the captured node; we compute the probability of the event A ={the link is compromised}. Let B be the event $\{V = V_1 \text{ or } V = V_2\}$, that is, the captured node is one of the end nodes of the considered link. $Fail(1)$ is equal to $Pr\{A\}$.

5.1.1.1. Single node capture: basic scheme. In the basic scheme, it is claimed [8] that the probability of breaking a link when one node is captured is $\frac{m}{|P|}$, that is, $Fail(1) = \frac{m}{|P|}$, where m is the memory size and $|P|$ denotes the size of the key pool. We now show that the probability of compromising the link between V_1 and V_2 is $\frac{2}{n} + \frac{m}{|P|}$. Using the Bayes rule, we obtain

$$\begin{aligned} Fail(1) &= Pr\{A\} \\ &= Pr\{A|B\}Pr\{B\} + Pr\{A|B'\}Pr\{B'\} \end{aligned} \quad (4)$$

There are n nodes in the network and each link has two end nodes, so $Pr\{B\} = \frac{2}{n}$. The probability of not capturing any of the two link's edges is thus $Pr\{B'\} = 1 - \frac{2}{n}$. All the communication links of a compromised node succumb to the adversary, and therefore, the probability of compromising these is $Pr\{A|B\} = 1$. Finally, because m keys are compromised per captured node, and there are $|P|$ keys in total, we obtain

$$Pr\{A|B'\} = \frac{m}{|P|} \quad (5)$$

From Equation 4 and assuming $|P|, n \to \infty$, we obtain

$$Fail(1) = 1 \times \frac{2}{n} + \frac{m}{|P|} \times (1 - \frac{2}{n}) \simeq \frac{2}{n} + \frac{m}{|P|} \quad (6)$$

5.1.1.2. Single node capture: key splitting scheme. For simplicity, consider the case that each pair of nodes uses only two half-part keys, even if they have more in common. From the *Fail* function, as per Equation (4), we also have $Pr\{B\} = \frac{2}{n}$ and $Pr\{B'\} = 1 - \frac{2}{n}$, and $Pr\{A|B\} = 1$. Now, for the $Pr\{A|B'\}$, note that when a single node is captured, $2m$ key slices are compromised. The considered link uses two halves of keys from the key pool, and it will be compromised if both of them are known to the adversary, that is, $(2m/2|P|)^2 = (m/|P|)^2$. Note that if only one-half of a key is revealed to the adversary, we do not count such a link as compromised. We discuss specifically the ramifications of the adversary having partial knowledge of the key splits further in the succeeding text. Thus, we obtain

$$Pr\{A|B'\} = \left(\frac{m}{|P|}\right)^2 \quad (7)$$

By substitution, we have

$$Fail(1) = 1 \times \frac{2}{n} + \left(\frac{m}{|P|}\right)^2 \times \left(1 - \frac{2}{n}\right) \simeq \frac{2}{n} + \left(\frac{m}{|P|}\right)^2 \quad (8)$$

From Equations (6) and (8), we see that the *Fail* metric is much lower for the key splitting scheme compared with the basic RKD scheme. Results can be even better in favor of key splitting, in the case that more than two key slices are used to establish the connection. We show that in the next part.

5.1.2. General case.

For each scheme, we calculate the probability of compromising a link when s nodes are captured, but none of the compromised nodes is incident on the considered link.

5.1.2.1. Multiple node capture: basic scheme. Recall the probability from Equation (5); thus, the probability to not reveal a key is $(1 - \frac{m}{|P|})$. When s nodes are captured, the probability that a key is not revealed is $(1 - \frac{m}{|P|})^s$. As a result, the probability that a key, and consequently a link that uses the key, is comprised is

$$1 - \left(1 - \frac{m}{|P|}\right)^s \quad (9)$$

5.1.2.2. Multiple node capture: q-composite scheme [9]. For the q-composite scheme, assume two nodes share i keys (with $i \geq q$). The probability that two nodes have i keys in common is

$$p(i) = \frac{\binom{m}{i} \times \binom{|P| - m}{m - i}}{\binom{|P|}{m}} \quad (10)$$

Also, the probability that two nodes can establish a secure link is $p_c = p(q) + p(q+1) + \cdots + p(m)$.

Then, with s compromised nodes, the probability of that link being compromised is $(1 - (1 - \frac{m}{|P|})^s)^i$. Hence, the probability that any link between two benign (non-compromised) nodes is compromised is

$$\sum_{i=q}^{m} \left(1 - \left(1 - \frac{m}{|P|}\right)^s\right)^i \times \frac{p(i)}{p_c} \quad (11)$$

5.1.2.3. Multiple node capture: (z,q)-Splitting scheme. A link would be compromised if all of i key slices shared by the two incident nodes are known to the adversary. Let $p'_z(i)$ be the probability that the two nodes have i key slices in common

$$p'_z(i) = \frac{\binom{z \times m}{i} \times \binom{z \times |P| - z \times m}{z \times m - i}}{\binom{z \times |P|}{z \times m}} \quad (12)$$

and p'_c denotes the probability that two nodes can establish a secure link in key splitting scheme, that is, $p'_c = p'_z(q) + p'_z(q+1) + \cdots + p'_z(z \times m)$.

Then, the probability for a link to be compromised is

$$\sum_{i=q}^{z \times m} \left(1 - \left(1 - \frac{m}{|P|}\right)^s\right)^i \times \frac{p'_z(i)}{p'_c} \quad (13)$$

In Figure 3, we plot the probability of compromising a link when s nodes are captured for different schemes:

- *Basic RKD scheme* (Equation (9)),
- *q-composite scheme* with $q = 1$ (Equation (11) with $q = 1$)
- *q-composite scheme* with $q = 2$ (Equation (11) with $q = 2$)

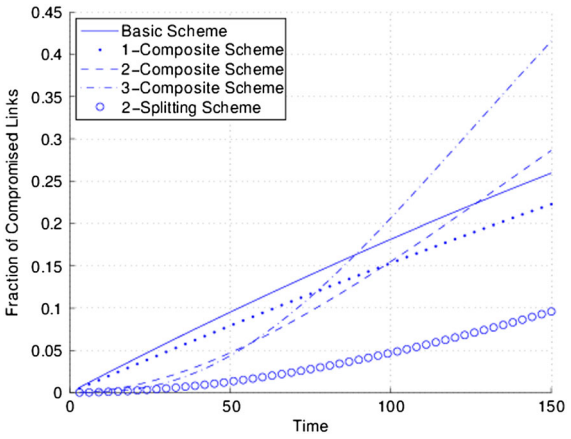

Figure 3. Resilience against node capture; comparison. ($p_c = 0.33$, $m = 200$).

- *q-composite scheme* with $q = 3$ (Equation (11) with $q = 3$)
- *Key splitting scheme* with $z = q = 2$ (Equation (13)).

Our key splitting scheme achieves a much lower probability of link compromise, increasing resilience against node capture attacks. For example, if 50 nodes in the network are captured, 9.5%, 7.9%, 4.7%, and 4.4% of other links would be compromised with basic scheme, 1-composite scheme, 2-composite scheme, and 3-composite scheme, respectively; while in the key splitting, only 1.3% of links are compromised. Even in the case of 150 captured nodes, less than 10% of the links are compromised for the 2-splitting scheme, while the rest of the schemes perform much worse. The difference between 2-composite and 2-splitting is that the former tries to increase the resilience with the cost of memory/connectivity; while the latter tries to have such improvement without such cost. The simulations show that we have approximately

$$fail_{2-composite}(s) = fail_{2-splitting}(2s) \tag{14}$$

In the key splitting scheme, an adversary may get only some of the key slices used to establish a link, that is, one of the two 64 bits key slices in the 2-splitting scheme. This is a case of information leakage. Given that the key parts are inputs to a one-way hash function h to produce the complete key, the adversary cannot generate the complete key, unless a brute-force attack is done for all the possible missing key parts. We calculate this probability that out of i shared key parts x of them are revealed:

$$\sum_{i=q}^{z\times m} \binom{i}{x} \left(\left(1-\frac{m}{|P|}\right)^{s}\right)^{i-x} \left(1-\left(1-\frac{m}{|P|}\right)^{s}\right)^{x} \times \frac{p'_z(i)}{p'_c} \tag{15}$$

We plot this probability in Figures 4 and 5 for the 2-splitting and 4-splitting schemes. We observe that the fraction of fully compromised links is low, especially for the 4-splitting scheme. However, the adversary has an increased probability of capturing a small number of slices, especially as more nodes are captured over time. This gets a good chance to the adversary to run a brute-force attack and obtain the missing parts. In the section that follows, we prove why even if the adversary can obtain parts of the complete keys, our splitting scheme still performs much better compared with the alternative schemes in the literature.

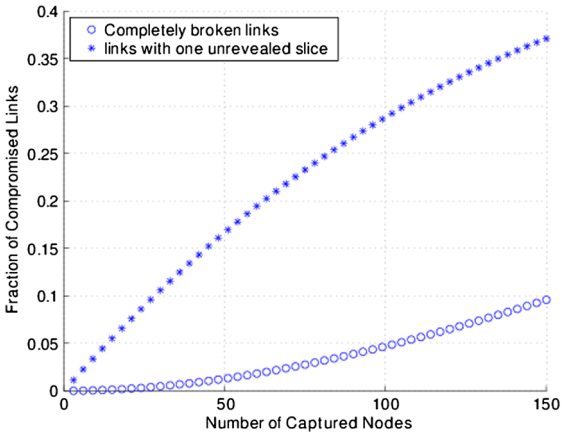

Figure 4. Fraction of fully compromised links and fraction of links with information leakage; 2-splitting. ($p_c = 0.33$, $m = 200$).

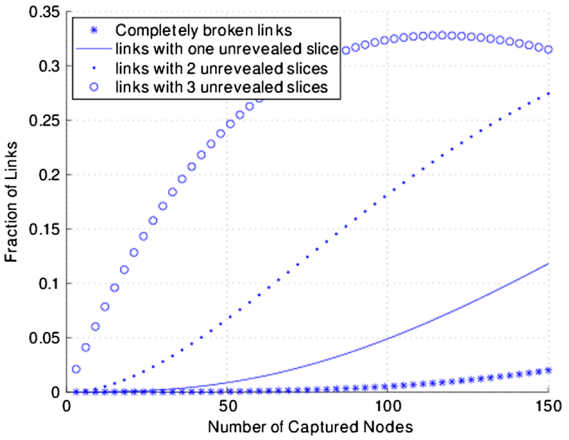

Figure 5. Fraction of fully compromised links and fraction of links with information leakage; 4-splitting. ($p_c = 0.33$, $m = 200$).

5.1.3. Computational compromise of links.

Considering relative strengths for the adversary, notably, the rates of capturing nodes and the brute-force checks, key splitting comes with a trade-off. That is, while splitting the keys greatly improves resilience against node capture, an adversary holding a subset of the key slices forming the secret key could brute-force the unknown key part. Considering the case that two key slices are used, the brute-force check would be done on strings of 64 bits long. Depending on how fast the adversary can run the brute-force checks and the available adversarial hardware, additional links can be compromised. We add this latter factor to key splitting in Figure 6, using an equal capture rate for all the schemes.

We assume a network with 100 000 links and a capture rate of one node per time unit for the adversary. Next, we consider really the worst case for our key splitting scheme by assuming an adversary that can run 10 brute-force checks in a time unit, that is, run 10×2^{64} computations per time. We plot the fraction of links that the adversary is able to compromise during the time. We observe that the curve for key splitting rises slightly only. Adversaries with overwhelming computation powers could perform better, but key splitting certainly raises the bar. Next, we test key splitting for different brute-force attack powers.

5.1.4. Adversarial brute-force power effect.

We assume an adversary that can capture one node per time unit. We want to study the effect of running a brute-

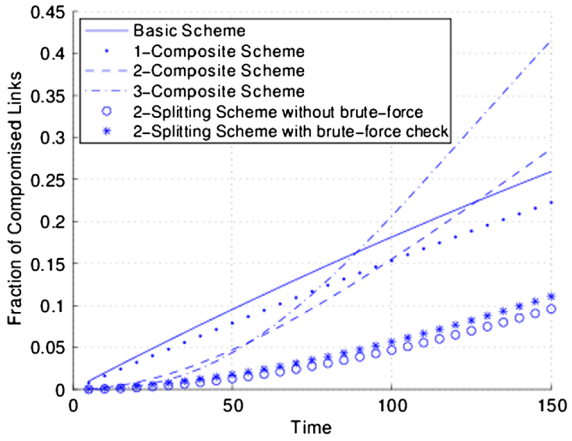

Figure 6. Resilience against node capture and exploitation of information leakage; comparison. ($p_c = 0.33$, $m = 200$).

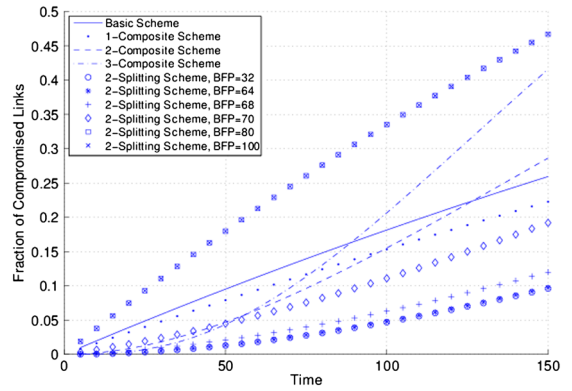

Figure 7. Resilience against node capture and effect of adversarial power; comparison. ($p_c = 0.33, m = 200$).

force check over the key splitting scheme. First, we run our simulations for different values of BFP parameters and then compare the results with the basic and q-composite schemes. Recall from Section 2, that BFP shows the number of brute-force operations that the adversary is capable of running in a time unit. For example, $BFP = 48$ means that the adversary can search (and try) all possible 48-bit binary strings in a time unit or equivalently all 32-bit binary strings 2^{16} times in a time unit.

We present our results for different BFP powers in Figure 7. We observe that for BFP values up to 64, the curve for the 2-splitting scheme does not change considerably and performs better than all the remaining q-composite and the basic schemes. Even for $BFP = 70$, which means 2.7×10^{11} times searching all possible 32-bit binary strings in a time unit, key splitting shows better resilience when more than 50 nodes are captured. It is only against extremely capable adversaries with overwhelming power that the key splitting scheme has worse performance, that is, $4,5 \times 10^{15}$ searches of 32-bit strings in a time unit. Clearly, this computational power is not practical at all. A simple calculation shows that if adversary has capability to compute 10^9 hash functions in a second (with using several super computers and very fast implementation), it takes more than 580 years to search a 2^{64} key space one time.

A considerable point is that for very high BFP values, that is, $BFP \geq 80$, the curve for 2-splitting does not change at all. The reason is that for these values, the adversary is able to compromise all *partially broken* links, and there are no more links to compromise with brute-force checking.

5.1.5. Advanced brute-force check.

Consider a partially broken link and an adversary who runs a brute-force check to get the unrevealed key slice. When the adversary finds the unrevealed key slice, he or she will compromise that link. In addition, the adversary has this chance to use this new key slice for compromising other links in the network. We study this scenario and show that the percentage of compromised links by this method is negligible, when the adversary does not have overwhelming power.

In a single time unit, the adversary can capture one node and obtains its key ring, that is, $2 \times m$ key slices in a 2-splitting scheme. Using a brute-force check, he or she is able to search a 2^{BFP} key space. Thus, the adversary finds 2^{BFP-64} key slices in a time unit. When this number is much smaller than the number of key slices in a key ring ($2 \times m$), the effect of this kind of attack is negligible. Figure 8 shows the fraction of compromised links for different situations: no brute-force checking, normal brute-force checking (when the revealed key slice is used only for the considered link), and advanced brute-force checking (when the revealed key slice is used for the considered link and other possible links in the network). We assume an adversary with very high (reasonably impractical) computational power ($BFP = 68$) and observe that the curve for key splitting rises slightly. For more reasonable *BFP* values, the change in the curve is negligible.

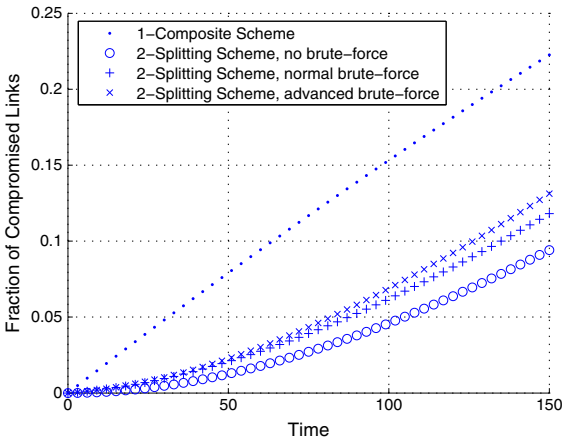

Figure 8. Effect of advanced brute-force check. ($p_c = 0.33$, $m = 200$).

DOI: 10.1002/sec

5.1.6. z parameter effect.

We now compare the resilience of different z-splitting schemes against each other and the 1-composite scheme, as it is one of the best performing RKD schemes in our analysis.

In Figure 9, we observe the effect of the z parameter on resilience against node capture for a period of 150 time units. 16-splitting performs better than 8-splitting, and 8-splitting performs better than 4-splitting and 2-splitting schemes. Therefore, splitting the key in many parts improves resilience against node capture. However, WSNs may be exposed to node capture attacks for longer time intervals.

In order to study such a scenario, we extended the exposure period for the network and tested our scheme over 300 time units. Figure 10 demonstrates our results. As time elapses, 8-splitting and 16-splitting curves rise rapidly. However, 2-splitting, which was the worst performing scheme in the previous scenario, now shows the best resilience.

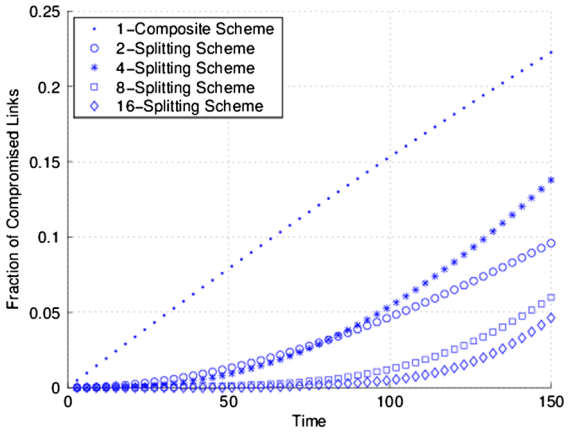

Figure 9. Resilience against node capture and effect of z parameter (number of slices); comparison. ($p_c = 0.33, m = 200$).

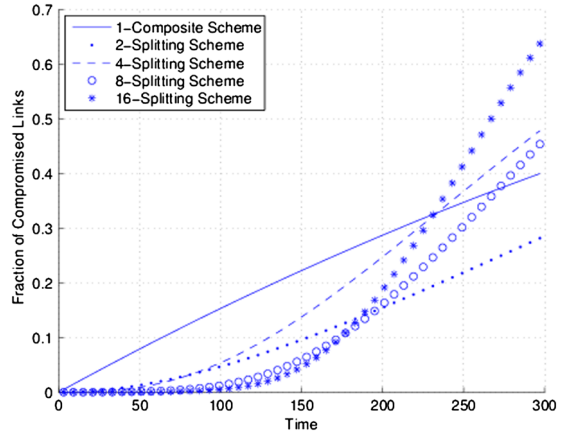

Figure 10. Resilience against node capture and effect of z parameter (number of slices); an extended view comparison. ($p_c = 0.33, m = 200$).

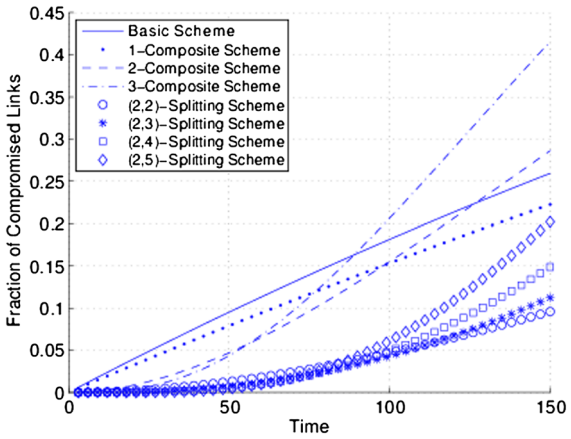

Figure 11. Resilience against node capture and effect of q parameter; comparison. ($p_c = 0.33, m = 200$).

Before choosing the best z configuration for the network, it is very important to consider the properties of WSN and its security requirements. For better resilience over a short time interval, that is, a tactical network with a shorter life time, 8-splitting or 16-splitting increases the bar for the attacker. However, for longer time intervals, that is, WSN unattended for long periods, 2-splitting performs much better.

5.1.7. q parameter effect.

In Figure 11, we compare the resilience of (z, q)-splitting, using two key slices for each key ($z = 2$) and different values of q, ranging from 2 to 5. We can see that for lower numbers of captured nodes (less time for the adversary to capture), increasing q leads to increased node capture resilience. However, as time elapses, $q = z$ shows the best resilience. Our results are consistent with the analysis in [9], where q-composite schemes with $q > 1$ show better resilience for low number of captured nodes.

5.1.8. A general comparison.

In this part, we aim to answer the following question: *what network configuration achieves maximum resilience against node capture attacks?* In other words, we are searching for the optimal z and q values to configure the sensor network for maximum resilience against our adversary *Adv*. For our analysis we use two variants of m and p_c setups. For $Setup_A$, we use $m = 200, p_c = 0.33$ and for $Setup_B$, $m = 100, p_c = 0.5$. Both are commonly used in practice and in the related work.

Assume that the goal of *Adv* is to compromise a given percentage of network links. We use γ to denote the fraction of broken links the adversary wants to achieve. To compare the key splitting variants, we calculate the time needed by *Adv* to reach γ. We test our key splitting scheme for $z = 1, 2, 4, 8$ key slices and $q = z..12$ for the threshold parameter. Note that $z = 1$, that is, splitting into one slice only, corresponds to the q-composite scheme. For all our

simulations, we used keys of 128 bits length and two types of adversaries with $BFP = 49$ and $BFP = 70$.

Table I presents the best performing network configurations for z and q. We observe that some of the tested configurations perform better according to the γ parameter. For example, if the adversary wants to compromise 1% of the network links, the best configuration is $z = 4$ and $q = 7$ for $BFP = 49$, for either of the two setups of m and p_c. The (2,2)-splitting scheme achieves better resilience in many of the tested scenarios. For computationally strong adversaries ($BFP = 70$), the (1,1)-splitting (i.e., equivalent to 1-composite scheme) performs well when we set $\gamma = 30\%$. We can argue here that z and q parameters should be chosen according to the strength and the objective of the adversary. In a real world example, assume a tactical sensor network that cannot tolerate compromising of links greater than 1%. This can be translated to 1000 broken links in a network with 10^5 links. In order to achieve maximum resilience, the network in $Setup_A$ could be configured with $z = 4$, and $q = 7$ or $q = 9$, according to the expected adversarial computational power.

5.1.9. Adversarial benefit analysis.

We now wish to extend our analysis further and compare the key splitting schemes with the rest of the RKD schemes, using the adversarial benefit parameter (Equation (2)), as defined in Section 2. In other words, we compute the overall benefit for the adversary to hold a fraction of links over time.

Figure 12 shows the performance of the different schemes according to *AdvBen*, for a network operation time of 300 time units. 2-splitting is the most resilient scheme compared with the rest. 8-splitting performs slightly worse, while all the key splitting schemes are harder to *break* compared with q-composite and the basic schemes.

5.1.10. Simulation results.

In this part, we show that the simulation results match with the formulations in previous sections. Consider a simple network setting with 100 nodes and a memory size of 50 keys for each node. To reach connectivity level of $p_c = 0.5$, we use a key pool of size $P = 3577$ for 1-composite scheme and a key pool of size $P = 2946$ (5892 key slices) for 2-splitting scheme. In Figure 13, we plot the curves for

Table I. Best (z,q) network configuration; $Setup_A(m = 200, p_c = 0.33)$, $Setup_B(m = 100, p_c = 0.5)$.

	$Setup_A$				$Setup_B$			
	$BFP = 49$		$BFP = 70$		$BFP = 49$		$BFP = 70$	
γ (%)	z	q	z	q	z	q	z	q
1	4	7	4	9	4	7	4	7
10	2	2	2	2	2	2	4	6
20	2	2	2	2	2	2	2	2
30	2	2	1	1	2	2	2	2

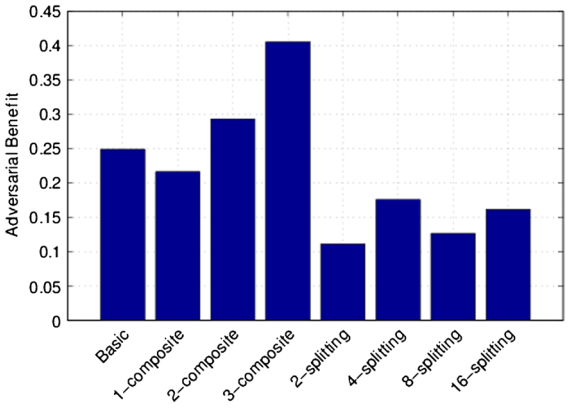

Figure 12. Normalized adversarial benefit from node capture in different schemes; comparison. ($p_c = 0.33, m = 200$, 100 samples in time interval $[0, 300]$).

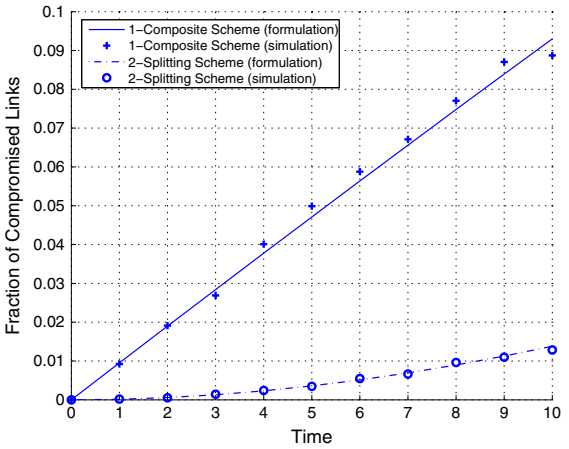

Figure 13. Fraction of compromised links; simulation results. ($p_c = 0.5, m = 50$).

fraction of compromised links according to Equations (11) and (13) as well as the simulation results. This figure shows correctness of our calculations in previous sections and demonstrates the resilience of key splitting scheme against node capture in comparison with q-composite scheme.

5.2. Performance analysis

We consider next the connectivity achieved by our scheme in comparison with the basic and q-composite RKD schemes. Consider a key pool of size $|P|$ and sensor nodes with memory $m \times L$ bits. The probability that two nodes have at least a common key in the basic scheme is

$$p_{c,basic} = \sum_{i=1}^{m} p(i) \tag{16}$$

where $p(i)$ denotes the probability that two nodes share exactly i keys (Equation 10). For the q-composite scheme, the conditions are tighter. Two nodes can communicate

securely if they have at least q common keys. Thus, the connectivity decreases to

$$p_{c,q-comp} = \sum_{i=q}^{m} p(i) \tag{17}$$

where $p(i)$ is the same as .

For key splitting, note that the key pool has $z \times |P|$ key slices and each WSN node stores $z \times m$ such key parts. Two nodes can establish a connection if they share at least z key slices. If we consider the general case, (z, q)-splitting, they should share at least q key slices to start a secure connection:

$$p_{c,split} = \sum_{i=q}^{z \times m} p'_z(i) \tag{18}$$

where $p'_z(i)$ is given in Equation (12).

We plot p_c in Figure 14, for different values of m in a constant pool size (in bits) and compare the schemes. For the q-composite scheme, we consider the case $q = 2$ (For $q = 1$, its connectivity is the same as that of the basic scheme). For our scheme, we consider (2,2)-splitting as well as (4,4)-splitting.

We observe that the key splitting scheme achieves much better connectivity than the q-composite scheme. In comparison with the basic scheme, sometimes we obtain a higher connectivity, while sometimes the connectivity is lower. Because there is a trade-off between connectivity and memory usage of this scheme, one may be interested in reducing the memory usage by keeping the connectivity constant. Besides, reducing the memory usage leads to an improvement in resilience against node capture, because the resilience against node capture is related to the memory usage in a reverse manner.

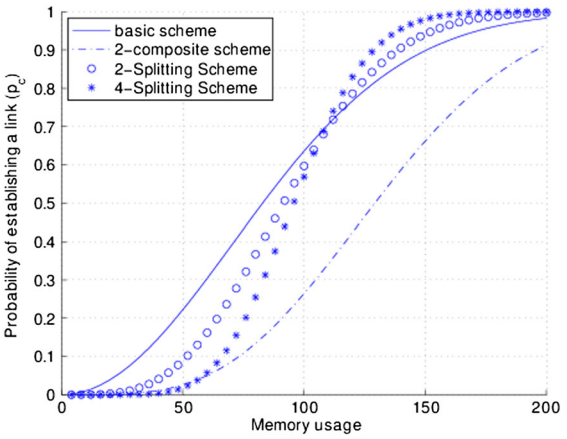

Figure 14. Connectivity versus memory usage for different random key pre-distribution schemes and key splitting; comparison. (P = 10 000 keys, or equivalently 20 000 half keys or 40 000 quarter keys).

This figure shows that, for example, if we consider the memory usage m = 140 (140 complete keys, 280 half-part keys, 560 quarter-part keys), then the value of p_c will be 0.863, 0.9056, and 0.9549 for the basic scheme, 2-splitting scheme, and 4-splitting scheme, respectively. This means a 5% and 11% improvement in p_c in two versions of our scheme in comparison with the basic scheme. To address the importance of such improvement, consider as an example that we have a network with n = 10 000 nodes and we want to increase the probability that the network be connected from 0.99999 to 0.999999. Such increase needs an 11% improvement over p_c.

On the other hand, if we are interested in a connectivity that corresponds to p_c = 0.9, the amount of memory usage will be 151, 139, and 129, respectively, that is, an 8% and 15% reduction in memory usage, respectively. Such reduction could be so important in memory constrained devices as sensor nodes, where a 4 KB RAM memory is very typical. Also, remember that decreasing the memory usage yields in increasing the resilience against node capture, because an adversary obtains a lower number of keys by capturing a constant number of nodes.

In this example, for $p_c > 0.7$ (approximately), we see an improvement over connectivity or memory usage. Usually, this is a high probability that two nodes have common keys; but, it could be employed in some situations, for example, when the density of nodes in an environment is very low. In this case, every node has a limited number of neighbors. So, it is very important for a node to establish a secure link with most of its neighbors. While for smaller values of p_c, we have to pay some costs (in terms of memory usage or connectivity), usually the improved security is worth the cost.

5.3. Comparison and numerical illustration

Table II shows a comparison between the following schemes:

- the basic scheme
- the 2-composite scheme
- the 2-splitting scheme
- the (2,4)-splitting scheme

We use a constant pool size $|P|$ = 10 000 and compare the memory usage as well as the resilience of the schemes with different p_c values. Table II demonstrates that key splitting schemes have better resilience in all situations. Regarding memory usage, 2-splitting uses a bit more memory compared with the basic scheme and (2,4)-splitting uses a bit more memory compared with the 2-composite scheme. However, this is not so when p_c is high. In all situations, the basic scheme and the 2-splitting scheme have lower memory usage than the 2-composite and the (2,4)-splitting schemes.

Table II. A comparison between resilience and memory usage of different schemes.

	$p_c = 0.33$			$p_c = 0.5$			$p_c = 0.8$		
Scheme	m	Fail(50)	Fail(100)	m	Fail(50)	Fail(100)	m	Fail(50)	Fail(100)
Basic	64	0.27	0.46	84	0.34	0.56	127	0.47	0.72
2-composite	109	0.14	0.38	130	0.16	0.44	173	0.18	0.52
2-splitting	77	0.08	0.23	92	0.09	0.27	123	0.10	0.32
(2,4)-splitting	121	0.03	0.19	136	0.03	0.23	166	0.04	0.29

5.4. Comparison with other random key pre-distribution schemes

In the previous subsections, we compared the resilience of key splitting scheme with the resilience of two important RKD schemes, that is, the basic scheme and the q-composite scheme. Now, we would like to extend our analysis and compare the key splitting with some more RKD schemes.

There are several works in the literature that engage hash functions to increase the resilience of RKD schemes. They usually develop one or several chains of keys or key materials and take advantage of the one-way property of hash functions.

One interesting improvement is the so-called key chain improvement [22]: consider a key pool as in the basic scheme. Extend the pool by creating some hash chains with the primary keys as roots of the chains. For each node, select m keys randomly from the extended key pool, such that no more than one key is selected from each chain.

If the primary key pool size and memory size are the same as in the basic scheme, connectivity will not be affected. But, this idea can decrease the chances of an adversary to compromise secure links in the network: consider a link that is secured with a key, say K. Even if the adversary captures some nodes and obtains a key in the chain corresponding to K, she succeeds if and only if the adversary's key is closer to the root than K. In the best case, the probability that the adversary fails to compromise the link given that she owns a key in the same chain as of K would be 33% [22]. It is less than the improvement by key splitting scheme.

As an another example, adaptive random pre-distribution [23] uses some hash chains to improve the connectivity. However, our analysis shows that the resilience is approximately the same as the basic scheme.

Zhao *et al.* proposed a *hashed RKD* utilizing hash chains and auxiliary nodes[24]. They showed that for very small number of captured nodes, their scheme is resistant against node capture attack. However, using their formula for fail function, we observe that it is not resistant if the number of captured nodes increases. We set the parameter $L = 300$ and $t = 5$ (some constants in their scheme) as they suggested and set $q = 10$ to reach the connectivity level near 0.33 as other schemes in the comparison.

Some schemes are location-aware and utilize deployment knowledge of nodes to improve the performance of the scheme. One of the most important ones is the scheme by Du *et al.* [25], which we mark it as *location-aware scheme* in the comparison figure. We plot its fail function according to their simulations in their paper. We observe that this scheme is comparable with the key splitting scheme, that is, for some values of s (number of captured nodes), key splitting scheme is more resistant than location-aware scheme and for other values of s, location-aware scheme is more resistant than key splitting scheme. This shows the excellence of our scheme: resilience of key splitting, without use of deployment knowledge, is approximately equal to the resilience of one scheme that uses deployment knowledge of nodes.

Levi *et al.* proposed another RKD scheme [26], entitled *ABCD* scheme. It uses reusable key pools. However, key splitting performs better than this scheme, too.

Figure 15 shows a comparison between resilience of the aforementioned RKD schemes. For the basic scheme, we set $m = 200$ and $p_c = 0.33$. For key splitting scheme, we use the same setting and also set $BFP = 64$. For chain improvement scheme, we use the same setting as of the basic scheme and also $\alpha = \frac{2}{3}$, that is, the best case for this scheme. For adaptive random pre-distribution scheme, the same setting as of the basic scheme is used. The setting for hashed RKD scheme is mentioned earlier. Also, we used the same setting for the location-aware scheme. In ABCD scheme, the authors claim that the memory usage is decreased to $m = 95$; however, the connectivity is $p_c = 0.33$ to be able to make a comparison

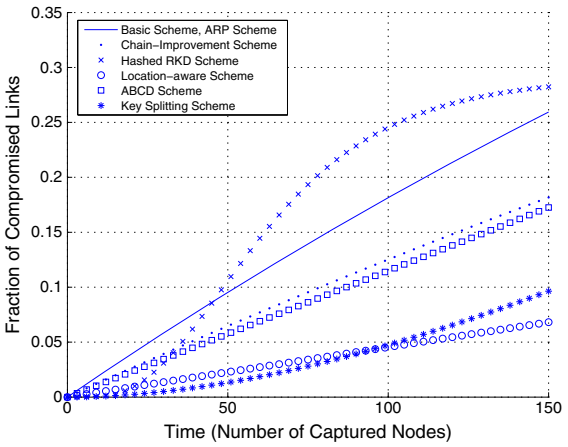

Figure 15. Resilience of some important key distribution schemes; comparison.

DOI: 10.1002/sec

with the others. This figure demonstrates that key splitting scheme and location-aware scheme are more resistant than other schemes.

5.5. Maximum supportable network size

In this section, we analyze the maximum supportable network size for the 2-splitting scheme and compare it to the rest of the RKD schemes. We calculate the maximum supportable size according to the fraction of compromised nodes that the network can tolerate and still be considered secure.

Given a standard capture rate for all the schemes, as well as a brute-force attack against key splitting, we observed that node capture resilience is drastically improved. Recalling Figure 6, an adversary would need 53 time units to compromise 10% of the network links when the basic scheme is used. For the 2-composite scheme, 77 time units are needed, while our splitting scheme introduces a real improvement, forcing the adversary to consume 140 time units to achieve 10% of broken links, even when a brute-force attack is done.

We now wish to measure the maximum size of a WSN that can be supported by the RKD schemes, given that there is a limit in the fraction of compromised links, after which the network is no longer secure. Following the initial deployment, each sensor can establish a secure connection with an expected number of nodes throughout the network. A node's degree can be computed as $d = n \times p_c$, where n is the network size and p_c is the given probability for any two nodes to establish a secure connection. It follows that d represents an average number of links an adversary could compromise after capturing a node.

For a long enough history of captured nodes, an adversary is expected to hold a large gamut of captured keys. This should allow the attacker to be confident enough and start compromising sensor links, throughout the network, but this time at numbers greater than d. Let f_m represent the percentage of compromised communications, after which the node capture attack in no longer bearable for the network. In other words, f_m represents a limit for the fraction of compromised links, to consider to the network secure. A secure network can thus be defined as ($f \leq f_m$), where f is the percentage of communications compromised by the adversary. As in [9], we define the global payoff requirement according to d:

Global payoff requirement: The adversarial gain following a node capture, measured in communication links compromised, should be on average of the node's degree. The total fraction of compromised communications should be kept below the f_m limit ($f \leq f_m$).

The payoff requirement defines the acceptable level of compromised links from a single node capture. We perform our analysis using the payoff requirement on the basis of time units needed by the adversary, to compromise a given percentage f_m of communications. f_m is a given parameter, and for our analysis, we set it equal to 0.1.

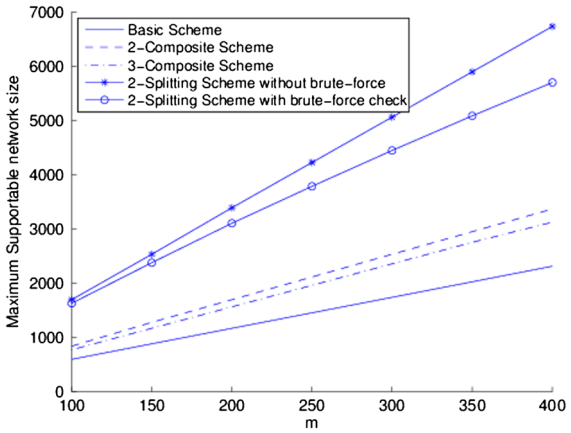

Figure 16. Maximum supportable network size for different network configurations ($p_c = 0.33, f_m = 0.1$).

Let $f(t)$ be the fraction of secure links compromised when t time units have passed. It follows that t_m is the time needed by the adversary to achieve $f_m = f(t_m)$. There are $\frac{nd}{2}$ links in total in the network. According to the payoff requirement, the number of network links compromised in the network should be less than the degree of the node, thus $\left(\frac{nd}{2} - x_m d\right) f_m \leq x_m d$. By simplifying, we obtain

$$n \leq 2x_m \left(1 + \frac{1}{f_m}\right) \tag{19}$$

For $p_c = 0.33$, $f_m = 0.1$, and $m = 200$, the maximum supportable network size for the 2-splitting scheme without a brute-force attack is 3388 nodes. For the 2-composite and the basic RKD schemes, the maximum supportable network sizes are 1694 nodes and 1166 nodes equivalently. Even if a brute-force attack is launched against the network, the 2-split scheme performs better and supports a network size of 3102 nodes. Figure 16 shows the maximum supportable network sizes for different network configurations.

6. CONCLUSION

We considered the problem of increasing the resilience of RKD schemes to node capture. We proposed a new scheme, key splitting, which can significantly increase the resilience compared with existing schemes. As long as the adversary does not have overwhelming computational power, to brute-force links for which it has partial knowledge of the secret material, our scheme greatly increases resilience to node capture attacks. Essentially, key splitting raises the bar against a powerful and sophisticated adversary.

We analyzed key splitting, using variants of slitting the key, and proved it is applicable to every RKD scheme. We achieve a significantly lower fraction of compromised links compared with other schemes, especially when

the number of compromised nodes increases. Key splitting achieves better network connectivity and is thus able to support much larger secure network deployments, compared with the state of the art. Finally, we have proven that all these benefits come at no extra communication and computational overhead for the sensors.

ACKNOWLEDGEMENTS

This work is partially supported by the Iranian Research Institute for ICT (grant T/500/19241).

REFERENCES

1. Liu A, Ning P. TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks, in *Proceedings of the 7th IPSN*, Washington, DC, USA, 2008.
2. Piotrowski K, Langendoerfer P, Peter S. How public key cryptography influences wireless sensor node lifetime, in *Proceedings of the 4th ACM SASN*, New York, 2006.
3. Papadimitratos P, Luo J, Hubaux J-P. A randomized countermeasure against parasitic adversaries in wireless sensor networks. *IEEE Journal on Selected Areas in Communications* 2010; **28**: 1036–1045.
4. Hartung C, Balasalle J, Han R. Node compromise in sensor networks: the need for secure systems. *Technical Report CU-CS-990-05*, University of Colorado at Boulder, 2005.
5. Bohli J, Papadimitratos P, Verardi D, Westhoff D. Resilient data aggregation for unattended WSNs, in *IEEE LCN SenseApp*, Bonn, Germany, October 2011; 994–1002.
6. Newsome J, Shi E, Song D, Perrig A. The sybil attack in sensor networks: analysis & defenses, in *Proceedings of the 3rd IEEE International Symposium on Information Processing in Sensor Networks*, New York, 2004; 259–268.
7. Du W, Deng J, Han YS, Varshney PK. A pairwise key pre-distribution scheme for wireless sensor networks, in *Proceedings of the 10th ACM conference on Computer and communications security*, New York, 2003; 42–51.
8. Eschenauer L, Gligor VD. A key-management scheme for distributed sensor networks, in *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS)*, New York, 2002; 41–47.
9. Chan H, Perrig A, Song D. Random key predistribution schemes for sensor networks, in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2003; 197–213.
10. Ehdaie M, Alexiou N, Ahmadian M, Aref M, Papadimitratos P. Key splitting for random key distribution schemes, in *Proceedings of the 7th workshop on Secure Network Protocols (NPSec)*, Austin, Texas, USA, 2012.
11. Perrig A, Stankovic J, Wagner D. Security in wireless sensor networks. *Communications of the ACM* 2004; **47**: 53–57.
12. Giruka VC, Singhal M, Royalty J, Varanasi S. Security in wireless sensor networks. *Wireless Communications and Mobile Computing* 2008; **8**(1): 1–24.
13. Shi E, Perrig A. Designing secure sensor networks. *Wireless Communications, IEEE* 2004; **11**: 38–43.
14. Wood AD, Stankovic JA, D A, A J. Denial of service in sensor networks, in *Upper Saddle River*, Prentice Hall, Inc, 2002.
15. Poturalski M, Papadimitratos P, Hubaux J-P. Towards provable secure neighbor discovery in wireless networks, in *ACM Workshop on Formal Methods in Security Engineering*, Alexandria, VA, USA, October 2008; 31–42.
16. Poturalski M, Papadimitratos P, Hubaux J-P. Secure neighbor discovery in wireless networks: formal investigation of possibility, in *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Tokyo, Japan, March 2008; 189–200.
17. Douceur JR. The sybil attack, in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London, UK, 2002; 251–260.
18. Liu D, Ning P. Establishing pairwise keys in distributed sensor networks, in *Proceedings of the 10th ACM conference on Computer and Communications Security (CCS)*, New York, 2003; 52–61.
19. Parno B, Perrig A, Gligor V. Distributed detection of node replication attacks in sensor networks, in *Security and Privacy, 2005 IEEE Symposium on*, Oakland, California, US, may 2005; 49–63.
20. Moore T. A collusion attack on pairwise key predistribution schemes for distributed sensor networks, in *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, Pisa, Italy, March 2006; 5–255.
21. Papadimitratos P, Deng J. Stealthy pre-attacks against random key pre-distribution security, in *Proceedings of the IEEE International Conference on Communications - Communication and Information Systems Security Symposium (ICC'12 CISS)*, Ottawa, Canada, 2012; 251–260.
22. Kur J, Matyas V, Svenda P. Two improvements of random key predistribution for wireless sensor networks, in *Proceedings of the International Conference on Security and Privacy in Communication Networks*, Padua, Italy, 2012.

23. Huang S-I, Shieh S, Wu S. Adaptive random key distribution schemes for wireless sensor networks, *Computer Security in the 21st Century*, Springer, 2005; 91–105.
24. Zhao H, Hu J, Qin J, Varadharajan V, Wan H. Hashed random key pre-distribution scheme for large heterogeneous sensor networks, in *Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, 2012; 706–713.
25. Du W, Deng J, Han YS, Chen S, Varshney P. A key management scheme for wireless sensor networks using deployment knowledge, in *Proceedings of the IEEE INFOCOM 2004, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, Washington, DC, 2004; 586–597.
26. Levi A, Tasci S, Lee Y, Lee Y, Bayramoglu E, Ergun M. Simple, extensible and flexible random key predistribution schemes for wireless sensor networks using reusable key pools. *Intelligent Manufacturing Springer* 2010; **21**(5): 635–645.