A Set-Covering Approach for Modeling Attacks on Key Predistribution in Wireless Sensor Networks

Patrick Tague
Network Security Lab (NSL)
Dept. of Electrical Engineering
University of Washington
Seattle, Washington, USA
Email: tague@ee.washington.edu

Jooyoung Lee
Dept. of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario, Canada
Email: j3lee@uwaterloo.ca

Radha Poovendran
Network Security Lab (NSL)
Dept. of Electrical Engineering
University of Washington
Seattle, Washington, USA
Email: radha@ee.washington.edu

***Abstract*— We study attacks by adversaries which aim to compromise links in a wireless sensor network through various techniques which are modeled using the set-covering problem. We discuss the effects of the attacks and present techniques which can be used to mitigate the effects of the attacks. Furthermore, we analyze the performance of various key predistribution schemes with and without the mitigation techniques.**

I. Introduction

The use of large-scale wireless sensor networks (WSNs) in hostile environments requires the development of secure decentralized protocols. The difficulties in developing such protocols lay in the sensor node limitations such as wireless radio range, battery energy, and computational capability. The restrictions on the computational capability of WSN nodes lead to the common assumption that secure protocols can rely only on symmetric key cryptography.

A promising solution for the establishment of symmetric keys in WSN applications is *key predistribution*, studied in various papers (e.g. [1]–[10]). Especially, we focus on the framework presented in [4] that consists of three phases: *key assignment*, *shared-key discovery*, and *path-key establishment*. In a key predistribution scheme (KPS), *seeds* are distributed to sensor nodes prior to network deployment. After deployment of the WSN, the shared-key discovery phase takes place, where two nodes in wireless communication range determine the existence of shared seeds. If the two nodes share seeds, a *link key* can be computed as a function of one or more of the shared seeds. The path-key establishment phase takes place if there is no common seed between a pair of nodes in wireless communication range. In this case, the nodes find a path of secure links between them and transfer a key in encrypted form via the path.

The concepts of key predistribution in [4] have also been combined with the key establishment schemes of [1], [2] based on threshold secret-sharing. In the framework of [10], each node is assigned a share from each of K polynomials randomly selected from a pool of P bivariate polynomials. Any pair of nodes which contain shares of a common polynomial can compute a unique link key. Once an adversary collects t shares of a polynomial, however, any link key computed using shares of the polynomial are compromised. Shared-key discovery and path-key establishment phases similar to those of [4] can be used with threshold schemes.

The predistributed seeds of a general KPS can take the form of cryptographic keys as in [4] or polynomial shares as in [10]. However, additional results have been proposed which change the way in which link keys are computed from the seeds. For example, in [3], [5], [7], [9], the shared-key discovery protocols compute a link keys using a cryptographic hash function with seeds as inputs. Regardless of how the seed is used, the shared-key discovery protocol must reveal information pertaining to which seeds are stored in each WSN node in order for neighboring nodes to determine if sufficient seeds are shared to establish a link key. For example, the authors of [4] propose the transmission by each node of the identifiers (IDs) of the seeds contained in the node, noting that this might reveal too much information to an adversary. The authors of [4] further propose the use of a private shared-key discovery protocol based on encryptions of random nonces in order to reduce the amount of information revealed to an adversary. We analyze and discuss these shared-key discovery protocols in Section III.

The primary contribution of this paper is presentation, analysis, and discussion of various attacks which can be performed by the adversary using the information revealed during the shared-key discovery protocol. We investigate the impact of various KPS attacks and discuss possible mitigation techniques.

For the purposes of this paper, we assume that the adversary is able to physically capture sensor nodes and access all information stored within the nodes. Furthermore, we assume the adversary is able to eavesdrop and record transmissions throughout the network and determine which WSN node is the sender and receiver of each transmission. We do not consider attacks on network protocols (including node replication, node fabrication, wormhole attacks, etc.) other than those directly involved with the KPS.

The paper is organized as follows. Section II discusses

This work is supported in part by the following grants: Collaborative Technology Alliance (CTA) from ARL, DAAD19-01-2-0011; ONR award, N00014-04-1-0479; ARO grant, W911NF-05-1-0491, and DOD IASP award.
This work was performed while J.L. was visiting NSL at the University of Washington.

several adversarial goals and corresponding attacks on the KPS. Section III discusses the information required in order for an adversary to mount the attacks discussed in Section II. Section IV presents possible techniques which can be used to mitigate the effects of the attacks. In Section V, we investigate the effect of the attacks and mitigation techniques on the KPS. In Section VI, we discuss the performance of KPSs in terms of message overhead and computational complexity. Finally, we summarize our results in Section VII.

II. Attacks on Key Predistribution

In this section, we investigate KPS attacks which can be mounted by adversaries with different goals. We consider adversaries with the following goals: recovery of all predistributed seeds, recovery of a sufficient set of predistributed seeds to compromise all links, and disconnection of the WSN such that there exist nodes which cannot exchange information.

A. Recovery of All Seeds

We first consider an adversary interested in recovering all predistributed seeds which exist in the WSN. We consider two possible attacks to achieve this goal: the *random capture attack* and the *seed-cover attack*.

Random capture attack: In [4] and several papers that followed (e.g. [5], [6], [9], [10]), the authors assume that the adversary captures each successive node at random, independent of previously captured nodes. This situation often corresponds to an adversary who physically captures a group of nodes before accessing the information within the sensors nodes.

Seed-cover attack: In [11], the authors investigate the effect of an adversary able to capture nodes in a sequence using the information recovered from previously capture nodes. We consider the problem in its generality.

The attack is based on the assumption that at each step the adversary can determine the number of uncompromised seeds contained in each uncaptured node and choose to capture the node with the maximum number of uncompromised seeds. The adversary is interested in minimizing the number of nodes which must be captured in order to recover all predistributed seeds. However, this is equivalent to solving the set-covering problem [12], known to be NP-hard. Hence, we provide an algorithm based on a greedy set-covering heuristic in Fig. 1, where $S(i)$ is the set of seeds contained in node i, N is the total network size, and $\mathcal{C}$ is the set of captured nodes. According to [12], this greedy algorithm is sub-optimal by at most a factor $\ln(K)+1$.

KPS schemes which rely on threshold secret-sharing [6], [10] require a threshold number t of shares of each secret to be compromised before links using the secret can be compromised. Thus, a slight modification must be made to reflect this difference. The algorithm for the seed-cover attack on threshold schemes is given in Fig. 2.

Seed-Cover Attack - [4] KPS:
Given: $S(1), \ldots, S(N)$
$X \leftarrow \bigcup_{i=1}^{N} S(i)$
$\mathcal{C} \leftarrow \emptyset$
while $|X| > 0$ **do**
  $\hat{n} \leftarrow \arg\max_{n \notin \mathcal{C}} |X \cap S(n)|$
  $X \leftarrow X \setminus S(\hat{n})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\hat{n}\}$
end while

Fig. 1. Greedy seed-cover attack algorithm.

Seed-Cover Attack - Threshold t KPS:
Given: $S(1), \ldots, S(N)$
$X \leftarrow \bigcup_{i=1}^{N} S(i)$
$c(x) \leftarrow 0, \forall x \in X$
$\mathcal{C} \leftarrow \emptyset$
while $\exists x \in X, c(x) < t$ **do**
  $\hat{x} \leftarrow \arg\max_{\{x \in X, c(x) < t\}} c(x)$
  find $n \notin C, \hat{x} \in S(n)$
  for all $x \in S(n)$ **do**
    $c(x) \leftarrow c(x) + 1$
  end for
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{n\}$
end while

Fig. 2. Greedy seed-cover attack algorithm for KPS based on threshold secret-sharing.

B. Recovery of Sufficient Keys

We next consider an adversary interested in recovering only the predistributed seeds which were used to compute link keys, thus compromising all of the links in the WSN. The adversary can choose to attack either the set of all possible links or only links that have been established in the WSN. The following attack is valid for either of these cases.

Link-cover attack: The attack is based on the assumption that the adversary can determine the set of seeds used to secure each link in the network and choose to capture the node which will allow the adversary to compromise the maximum number of additional links. We note that this attack is fundamentally different from the seed-cover attack because not all seeds need to be recovered to compromise all of the links in the network. Letting $S(i)$ denote the set of seeds stored in node i, the adversary can construct the collection of sets $\Phi = \{S(i) \cap S(j) : i \neq j\} \setminus \{\emptyset\}$, where each element represents the set of seeds shared by nodes i and j. The collection of subsets of $S(i)$ represent the possible sets of seeds used to secure links incident to node i. Again, we see that the optimal execution of the link-cover attack is equivalent to solving the set-covering problem [12] and hence is NP-hard. We provide an algorithm based on a greedy set-covering heuristic in Fig. 3 which is suboptimal by at most a factor $K\ln(2)+1$.

C. Disconnecting the Network

We next consider an adversary interested in capturing a sufficient number of sensor nodes to globally disconnect the WSN. In order to perform such an attack, sufficient information must be available for the adversary to construct the key graph representing all pairs of nodes able to compute

Link-Cover Attack:
Given: $S(1), \ldots, S(N)$
$\Phi \leftarrow \{S(i) \cap S(j) : i \neq j\} \setminus \{\emptyset\}$
$\mathcal{C} \leftarrow \emptyset$
while $|\Phi| > 0$ **do**
  $\hat{n} \leftarrow \arg\max_{n \notin \mathcal{C}} |\{X \in \Phi : X \subseteq S(n)\}|$
  $\Phi \leftarrow \{(S(i) \cap S(j)) \setminus S(\hat{n}) : i \neq j\} \setminus \{\emptyset\}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\hat{n}\}$
end while

Fig. 3. Greedy link-cover attack algorithm.

link keys. Once the key graph is available, the adversary can determine a separating set of nodes whose removal will disconnect the WSN. The choice of separating set can depend on the formation of the sensor network, flow of information, and the amount of effort the adversary is willing to expend.

If all communications are carried out using secure single-hop links, the adversary may not have to reconstruct the entire key graph. Since links can only exist between nodes within wireless communication range, the adversary only has to consider those links which exist in the key graph and the geometric random graph resulting from WSN deployment. Thus, the adversary may be able to disconnect the WSN by removing a set of nodes such that some nodes are physically unreachable from the remaining network. Such an attack is thus independent of the KPS. Hence, we are only concerned with adversaries interested in disconnecting the key graph, independent of the geometry of the WSN.

III. Performing Attacks

In order to perform the attacks discussed in Section II, the adversary must collect sufficient information by eavesdropping on network traffic and capturing nodes. If the shared-key discovery protocol consists of a plaintext exchange of seed IDs, the adversary can plan the sequence of attack events before physically disturbing the network.

To prevent the leakage of information under a key exchange in plaintext, the authors of [4] propose the use of a private shared-key discovery protocol. In this protocol, each node broadcasts

$$\alpha, E_{L_1}(\alpha), \ldots, E_{L_K}(\alpha),$$

where α is a random nonce and each L_i represents a seed. Any neighboring node able to decrypt a list item $E_{L_i}(\alpha)$ to recover α can determine that L_i is shared with the transmitting node.

Random-capture attack: The random-capture attack is equally effective under any shared-key discovery protocol, as the attacker does not take advantage of any of the information revealed by the shared-key discovery protocol. Hence, schemes which are not concerned with attacks other than random capture need not consider any shared-key discovery mechanism other than the exchange of seed IDs.

The random-capture attack serves as the baseline for comparison in Section V. Under a random-capture attack, the scheme of [4] results in a probability that any link between uncaptured nodes is compromised given that x nodes have been captured given by

$$\mathbf{fail}(x) = 1 - \left(1 - \frac{K}{P}\right)^K. \quad (1)$$

Seed-cover attack: The exchange of IDs reveals a significant amount of information to an adversary performing a seed-cover attack. However, we now show that the use of a private shared-key discovery protocol reveals just as much information. Under this attack, the first node capture is random because every node has an equal number of uncaptured keys. Once the first node is captured, the adversary can play the role of the captured node in the private neighbor discovery protocol and simply locate the node which shares the smallest number of keys.

Link-cover attack: The exchange of IDs allows for seamless performance of a link-cover attack. However, the use of a private shared-key discovery protocol does not allow the adversary to determine shared-key relationships between nodes until a sufficient number of the keys of each node are already captured. Hence, the adversary is not able to efficiently perform a link-cover attack when a private shared-key discovery protocol is used.

Disconnection attack: The information obtained during the exchange of IDs allows the adversary to completely reconstruct the key graph and perform a disconnection attack. Since the use of a private shared-key discovery protocol hides the key graph (except compromised links due to captured nodes), the adversary can only attempt to disconnect the network physically by disconnecting the geometric random graph representing physical node communication.

We conclude that a private shared-key discovery protocol is not sufficient to prevent the seed-cover attack. However, since the attacker cannot determine shared-key relationships between uncaptured nodes when the private shared-key discovery protocol is used, the adversary is unable to mount a link-cover or disconnection attack. We further discuss mitigation of the seed-cover attack in the next section.

IV. Attack Mitigation

In this section, we discuss techniques which can be used to mitigate the effect of the seed-cover attack presented in Section II. We discuss the use of *private ID exchange* and further investigate the impact of the seed-cover attack on a KPS.

A. Private ID Exchange

Similar to the private shared-key discovery technique presented in [4], we consider a generic idea of using two independent KPSs, say $\mathcal{KPS}_\alpha$ and $\mathcal{KPS}_\beta$, to exchange seed IDs privately. After exchanging $\mathcal{KPS}_\alpha$ IDs in plaintext, a pair of nodes sharing a seed in $\mathcal{KPS}_\alpha$ can compute a pairwise key which is then used to exchange IDs for $\mathcal{KPS}_\beta$. In this scheme, the link key is computed as a function of the shared $\mathcal{KPS}_\alpha$ and $\mathcal{KPS}_\beta$ seeds.

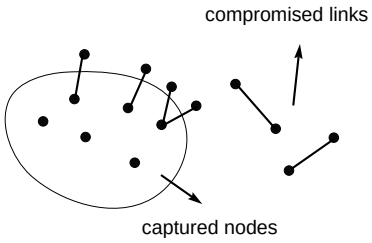

Fig. 4. Revealing of seed IDs under capture of x nodes.

However, we observe that even very small number of random node captures could reveal seed IDs for almost every node. If the adversary knows the $\mathcal{KPS}_\alpha$ seed which is used to encrypt and exchange $\mathcal{KPS}_\beta$ IDs between a pair of nodes, the adversary obtains seed IDs of $\mathcal{KPS}_\beta$ for the two nodes. Thus, the adversary is able to recover information from nodes other than those that are physically captures. By capturing x nodes, the adversary is able to recover the $\mathcal{KPS}_\beta$ IDs for the x captured nodes, any physical neighbors of the captured nodes which shared $\mathcal{KPS}_\alpha$ seeds, and nodes which are incident to links that are compromised due to the capture of the x nodes. For clarity, this scenario is illustrated in Fig. 4.

For simplicity, we only focus on the nodes incident with compromised links. Let N denote the total network size and d denote the average number of nodes in the neighborhood of a given node. In addition, let p_α denote the probability that any pair of nodes can establish a link key, referred to as the *local connectivity*, of $\mathcal{KPS}_\alpha$. Hence, the number of edges between nodes within wireless communication range that can establish a link key is approximately

$$e = \frac{1}{2} N d p_\alpha. \tag{2}$$

We look at this problem in detail. An adversary is able to obtain the seed IDs from captured nodes and their neighbors. Since the number of such neighbors is approximately dp_α, the probability that the set of IDs is revealed in this way is about $1-(1-\frac{dp_\alpha}{N})^x$, when an adversary captures x random nodes. As another possibility, an adversary knows the set of IDs if any link that the node used to exchange seed IDs is compromised. Based on the approximation that the link compromises are all independent, the overall probability $\mathbf{fail}_{id}(x)$ that a node reveals its seed IDs due to x random node captures is estimated as

$$\begin{aligned}\mathbf{fail}_{id}(x) &= 1-(1-\frac{dp_\alpha}{N})^x + (1-\frac{dp_\alpha}{N})^x \\ &\quad \times (1-(1-\mathbf{fail}(x))^{dp_\alpha}) \\ &= 1-(1-\frac{dp_\alpha}{N})^x (1-\mathbf{fail}(x))^{dp_\alpha}\end{aligned} \tag{3}$$

where $\mathbf{fail}(x)$ is computed over $\mathcal{KPS}_\alpha$.

We consider an example in which $N = 10,000$, $d = 40$, and $p_\alpha = 0.5$. If by capturing small number of nodes, an adversary is able to compromise 5% of the edges, or $5,000$ edges, the number of nodes incident with those edges might be a significant fraction of the total network size. We note that a node can be incident to multiple edges, so the number of nodes would still be less than N.

With the same parameters as above, suppose 50 node captures compromise 20% of the links between uncaptured nodes. Then we have

$$\mathbf{fail}_{id}(50) = 0.990, \tag{4}$$

which means the expected fraction of nodes with key identifiers revealed is about 99.0%.

Thus, the strategy that an attacker could take is to capture a small number of nodes at random, reveal the seed IDs for $\mathcal{KPS}_\beta$ for almost every node, and then mount a seed-cover attack. Hence, two or more layered KPSs do not provide additional resilience to a seed-cover attack, and an alternate method is required to mitigate the seed-cover attack.

B. Mitigating a Seed-Cover Attack

In light of the discussion on private shared-key discovery in Section III and the analysis on private ID exchange in Section IV-A, further investigation is needed in order to discover a technique for mitigating a seed-cover attack. We make the following claim, which is proved with a simple logical argument.

Claim 1: Any shared-key discovery protocol of a single KPS such that pairwise keys are determined by the set of common seeds and each node stores a constant number of seeds is susceptible to a seed-cover attack.

Proof: Assume that a shared-key discovery protocol is given, each node contains K out of P seeds, and the adversary has captured one node s. Assume further that the protocol is such that the adversary cannot use the information recovered from s to determine the number of seeds in a node n that are shared with s, i.e. a seed-cover attack cannot be mounted. However, this suggests that s and n were not able to execute a protocol to determine the existence of a shared key before s was captured, and the protocol is not an effective shared-key discovery protocol. ■

This claim suggests that it may be possible to mitigate a seed-cover attack if we allow K to vary between nodes. Hence, we investigate the effect of allowing nodes to hold varying numbers of seeds using the KPS of [4] for simplicity. We assume the private shared-key discovery protocol of [4] is being used, so only the seed-cover attack is of interest.

For each node, the key distribution center chooses K uniformly at random such that $K_1 \leq K \leq K_2$ and assigns a random selection of K keys from a key pool of size P. During the shared-key discovery phase, each node must mask the actual number of seeds it contains by transmitting

$$\alpha, \pi\left(E_{L_1}(\alpha), \ldots, E_{L_K}(\alpha), \nu_1, \ldots, \nu_{K_2-K}\right),$$

where each ν_i is a random nonce and π denotes a random permutation of the given elements. This prevents the adversary from knowing which of the quantities not corresponding to shared keys are unshared keys and which are useless information, thus reducing the effectiveness of the seed-cover attack. Such a scheme can be analyzed by noting that the average

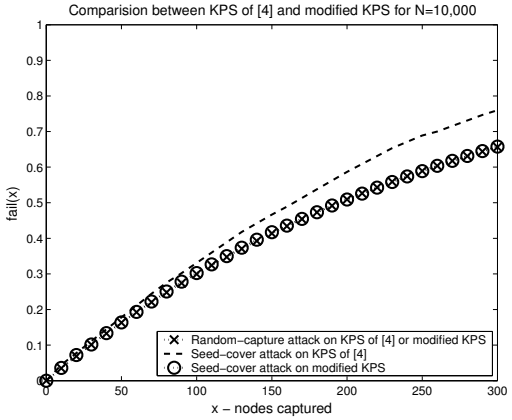

Fig. 5. Comparison between KPS of [4] with $P = 28,140$ and $K = 100$ and the modified KPS with $P = 28,140$ and K random between $K_1 = 80$ and $K_2 = 120$.

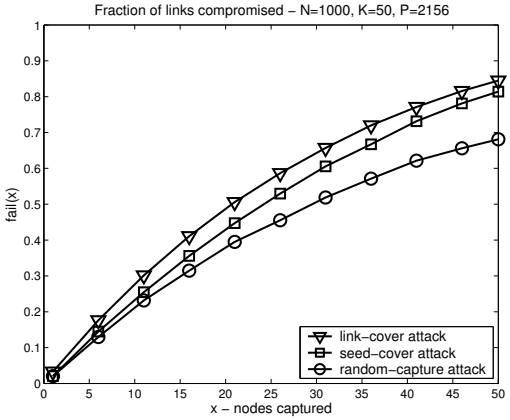

Fig. 6. Simulation of attacks on KPS of [4] for $N = 1,000$ and $K = 50$.

number of keys stored in each node is $K_{avg} = \frac{K_1+K_2}{2}$. Hence, local connectivity can be estimated as

$$p = \frac{1}{(K_2 - K_1 + 1)^2} \sum_{i=K_1}^{K_2} \sum_{j=K_1}^{K_2} \left(1 - \frac{\binom{P-i}{j}}{\binom{P}{j}}\right) \approx 1 - \frac{\binom{P-K_{avg}}{K_{avg}}}{\binom{P}{K_{avg}}}, \quad (5)$$

and the resilience to node capture can be estimated as

$$\mathbf{fail}(x) = 1 - \left(1 - \frac{1}{K_2 - K_1 + 1} \sum_{i=K_1}^{K_2} \frac{i}{P}\right)^x = 1 - \left(1 - \frac{K_{avg}}{P}\right)^x. \quad (6)$$

Example 1: We compare a KPS with $P = 28,140$ and $K = 100$ to a modified KPS with $P = 28,140$, $K_1 = 80$, and $K_2 = 120$. The local connectivity, average key storage, and the value of the function $\mathbf{fail}(x)$ is thus equivalent for both schemes. Since the modified KPS uses a randomized value of K, the resilience to a seed-cover attack is reduced to that of a random-capture attack, given by (1). The function $\mathbf{fail}(x)$ is plotted for each scheme under random-capture and seed-cover attacks in Fig. 5.

V. Simulations

We provide simulation results to demonstrate the effect of the attacks presented in Section II. Furthermore, we provide simulation results corresponding to the mitigation techniques proposed in Section IV.

In Fig. 6, we simulate $\mathbf{fail}(x)$ for a KPS as in [4] where $N = 1,000$ nodes are deployed with $K = 50$ keys each from a key pool of size $P = 2,156$. We plot the resulting $\mathbf{fail}(x)$ versus x for the random-capture, seed-cover, and link-cover attacks assuming that IDs are exchanged in plain text.

In addition to KPSs based on the scheme from [4], we are interested in schemes based on threshold secret-sharing, such as [6], [10], in which links are compromised as soon as the threshold t is reached for a given secret. In this case, we must use the modified version of the seed-cover attack given in Fig. 2. Furthermore, we note that the link-cover and seed-cover attacks are nearly identical for threshold schemes. In Fig. 7, we plot the simulated value of $\mathbf{fail}(x)$ for such a KPS where $N = 1,000$ nodes are deployed with shares of $K = 10$ of the $P = 90$ secrets using a threshold of $t = 5$. The plot demonstrates the difference between the random-capture and seed-cover attacks.

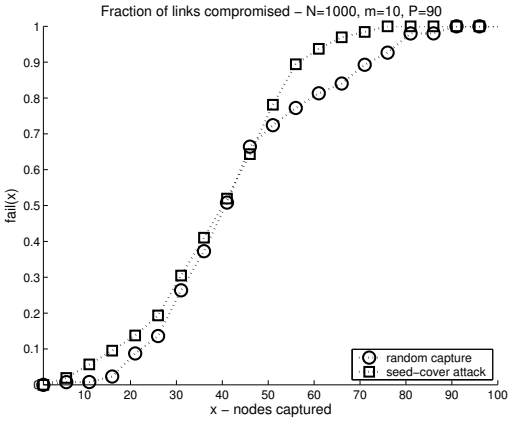

Fig. 7. Simulation of attacks on threshold KPS for $N = 1,000$, $K = 10$, and $t = 5$.

VI. Discussion

In a model presented in [4], the shared-key discovery phase occurs only when two sensor nodes are within wireless communication range. Then the total connectivity of the WSN is represented by the intersection of the geometric random graph representing the physical layer and the key graph.

If we relax the constraint and allow for shared-key discovery to execute between distant nodes by relaying messages via intermediate nodes, then any pair of nodes can establish a link key even though the intersection of the physical layer and the key graph is not connected, as long as the physical layer and key graph are each connected. If a node s wants to establish

a pairwise key with s', then it finds a path

$$s - s_1 - \ldots - s_m - s'$$

in the key graph and uses the intermediate links to exchange a random key k in encrypted form. In other words, s sends $E_{k_{s,s_1}}(k)$ to s_1, s_1 sends $E_{k_{s_1,s_2}}(k)$ to s_2, and so on until s' receives $E_{k_{s_m,s'}}(k)$ from s_m.

A KPS with a low local connectivity would show a stronger resilience to attacks compared to a KPS of a high local connectivity in the resulting network (with key storage fixed). In return, the establishment of link keys in a neighborhood would require a higher transmission overhead since paths between two nodes would be longer and the number of such paths would be even smaller.

Since a private shared-key discovery protocol (based on the encryptions of random nonces) also requires more computational complexity for encryption/decryption and message overheads (128-256 bits per key), the distribution center should carefully choose whether it takes a KPS with a low local connectivity based on plaintext ID exchange protocol or a KPS with a higher local connectivity based on a private shared-key discovery protocol.

VII. Summary

We have shown that various attacks on key predistribution schemes in WSN can be modeled using the set-covering problem. We present a collection of such attacks which require the adversary to either solve an NP-hard problem or choose a suboptimal solution. We have shown that no protocol requiring a contant number of predistributed seeds for a single KPS system in each node can be secure against the seed-cover attack. In order to mitigate the effects of a seed-cover attack, we proposed the randomization of the number of seeds K stored in each node.

References

[1] R. Blom, "An optimal class of symmetric key generation systems," in *Advances in Cryptology: Proceedings of EUROCRYPT '84, LNCS 209*, 1984.

[2] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Advances in Cryptology: Proceedings of CRYPTO '92, LNCS 740*, 1993.

[3] T. Leighton and S. Micali, "Secret-key agreement without public-key cryptography," in *Advances in Cryptology: Proceedings of CRYPTO '93, LNCS 773*, 1994.

[4] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002.

[5] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of IEEE Symposium on Security and Privacy*, 2003.

[6] W. Du, J. Deng, Y. Han, and P. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003.

[7] M. Ramkumar and N. Memon, "An efficient random key pre-distribution scheme," in *Proceedings of IEEE Conference on Global Communications*, 2004.

[8] S. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for distributed sensor networks," in *Proceedings of 9th European Symposium on Research Computer Security, LNCS 3193*, 2004.

[9] J. Lee and D. Stinson, "Deterministic key predistribution schemes for distributed sensor networks," in *Proceedings of Selected Areas in Cryptography, LNCS 3357*, 2004.

[10] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Transactions on Information and System Security*, vol. 8, no. 1, pp. 41–77, February 2005.

[11] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proceedings of the 2nd ACM Workshop on Security of Ad-hoc and Sensor Networks*, 2004.

[12] T. Cormen, C. E. Leiserson, and R. Rivest, *Introduction to Algorithms*. MIT Press, McGraw-Hill, 2000.

Appendix

The function $\mathbf{fail}(x)$ for a seed-cover attack on the KPS of [4] can be expressed recursively as

$$\mathbf{fail}(x) = \mathbf{fail}(x-1) + (1 - \mathbf{fail}(x-1))\, q(x) \qquad (7)$$

where $q(x)$ is the probability that a link which was uncompromised after $(x-1)$ nodes were captured is compromised when the x^{th} node is captured. Given that $\mathbf{fail}(0) = 0$, the closed-form solution for $\mathbf{fail}(x)$ is given by

$$\mathbf{fail}(x) = \sum_{i=1}^{x} q(i) \prod_{j=i+1}^{x} (1 - q(j)). \qquad (8)$$

The probability $q(x)$ is a function of the number of seeds M_{x-1} contained in the $(x-1)$ previously captured nodes and the maximum number of uncaptured keys $k_{max,x}$ contained in one of the remaining nodes, both of which are random variables. Given $M_{x-1} = m$ and $k_{max,x} = k$, the probability that a given uncaptured seed is captured in the x^{th} node is $\frac{k}{P-m}$. Hence, the probability $q(x)$ is thus given by

$$q(x) = \sum_{m=K}^{P} \sum_{k=0}^{K} \frac{k}{P-m} Pr[M_{x-1} = m, k_{max,x} = k], \qquad (9)$$

where $Pr[M_{x-1} = m, k_{max,x} = k]$ is the joint distribution of M_{x-1} and $k_{max,x}$. The probability $p_i(m)$ that a node contains exactly i uncaptured seeds when m of the P seeds are compromised can be computed as $p_i(m) = \frac{\binom{P-m}{i}\binom{m}{K-i}}{\binom{P}{K}}$. Hence, the probability $Pr[k_{max,x} = k | M_{x-1} = m]$ can be computed as

$$Pr[k_{max,x} = k | M_{x-1} = m] = c_1 \cdot p_k(m) \left(\sum_{i=0}^{k} p_i(m) \right)^{N-x-2} \qquad (10)$$

where c_1 is a normalizing constant to ensure the probability sums to 1 over all values of k. The probability $Pr[M_{x-1} = m]$ can be computed recursively as

$$Pr[M_{x-1} = m] = c_2 \sum_{k=0}^{K} Pr[M_{x-2} = m - k, k_{max,x-1} = k] \qquad (11)$$

where c_2 is a normalizing constant to ensure the probability sums to 1 over all values of m. The given equations can then be combined to yield an expression for $\mathbf{fail}(x)$ for a seed-cover attack on the KPS of [4].