2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

Key Management for Wireless Sensor Networks Using Trusted Neighbors

Xue Yuan, Helmut Jürgensen
Department of Computer Science
The University of Western Ontario
London, Ontario
Canada, N6A 5B7
hjj@csd.uwo.ca

Reza Azarderakhsh, Arash Reyhani-Masoleh
Department of Electrical and Computer Engineering
The University of Western Ontario
London, Ontario
Canada, N6A 5B7
razarder@uwo.ca, areyhani@eng.uwo.ca

Abstract

Establishing and maintaining secure communication paths in a wireless sensor network poses mathematical and technical challenges. We suggest to use random nonces to set up secure communication links after the network deployment phase. This idea exploits some of the inherent features of sensor networks, e.g.: nodes are densely deployed; the wireless communication areas of neighbouring nodes have a large overlap. We outline the analytical properties of such a scheme and show simulation results which corroborate them for practical purposes.

1. Security Issues in Sensor Networks

Wireless sensor networks typically consist of many nodes which are extremely limited in their computational capabilities, amount of available memory, and which have a considerable power consumption. The lifetime of such nodes depends on their ability to conserve power [1]. Sensor nodes are often operated in remote areas, unattended, inaccessible and without maintenance; therefore, such a network can be easily assaulted [10], both intentionally und accidentally. Thus a high level of data security may be required as nodes could be captured and communication could be compromised. Various cryptographic schemes could be used to afford security against various types of attacks. As the structure and topology of such a network need not be known prior to deployment and may even change later due to effects of its environment, establishing secure communication paths is an important issue. Currently, key pre-distribution schemes are considered as a practicable solution [10]. Eschenauer and Gligor [6] proposed a randomized key predistribution scheme, which relies on probabilistic key sharing among nodes; as the mathematical foundations this approach uses the theory of random graphs [5]. This scheme, referred to as the *(basic) EG scheme* in the sequel, relies on pre-loading a set of symmetric keys into the sensor nodes before they are deployed. If two nodes share a common key after deployment, they communicate securely with each other. Several other schemes using variations of this idea have been proposed in the literature. For a comprehensive survey see [11, 12]. As shown there, these schemes have the common disadvantage of not scaling well to large area coverage [12]. To address this issue we introduce a new scheme which uses nonces to establish secure communication. This idea was first proposed and developed in [11] and also announced in [12]. The present paper explains this idea in some detail and presents simulation results which support the theoretical work; a more in-depth presentation of the analytical results is being prepared [13]. We use random nonces to secure links set up after the key establishment phase; in this way one guarantees that an adversary arriving after the deployment of the network is unable to compromise links without actually capturing nodes. Analytical results of [11] and additional simulation results presented here show that this scheme achieves a far better connectivity and resilience than the EG scheme and that it outperforms the latter when integrated with deployment knowledge.

2. Key Management and Connectivity

The key management problem in distributed sensor networks has been studied with different objectives and metrics. Using few master keys would make the network extremely vulnerable; using pairwise keys for all sensor nodes requires unaffordable resources. Eschenauer and Gligor proposed a distributed key establishment mechanism which relies on probabilistic key sharing among the nodes and uses a shared-key discovery protocol for key establishment [6]; a large pool of symmetric keys, the *key pool,* is generated by a server. Each node is pre-loaded with a randomly selected set of keys from the key pool prior to deployment. After deployment, each node broadcasts its key information to its one-hop neighbours. Neighbouring nodes will

978-0-7695-3492-3/08 $25.00 © 2008 IEEE
DOI 10.1109/EUC.2008.125

have a key in common with some probability. If they happen not to have a key in common, they may use other nodes as intermediates. Establishing secure connections in this way is called *path-key establishment.* A variant to this idea was proposed in [3] as follows: In a *q-composite key pre-distribution scheme* two sensors set up a joint key only when they share at least q common keys where $q \geq 2$. This scheme was proposed to achieve an improved resilience against node capture attacks. In [4] a key management scheme is proposed which uses deployment knowledge; it is shown there that the performance—connectivity, memory, resilience—of networks can be substantially improved when deployment knowledge is used. Another key pre-distribution scheme, based on [2], was proposed in [7]. Instead of a large key pool a large pool of polynomials is used; in the key discovery phase each sensor node finds a node with which it shares a polynomial to establish a common key. An improved scheme involving knowledge prior to deployment and location-based information is described in [8]. A critical evaluation of these and other schemes is provided in [12].

3. Nonces to Establish Connections

We describe the trusted-neighbour key pre-distribution protocol, which relies on a so-called *real-world attacker model* as defined in [11]. The link key is replaced by a random value after the key discovery phase as follows: When the key discovery phase is complete each node has identified keys shared with its neighbours. If nodes A and B have a key in common, their communication can be secured using such a key; they replace such a key by a random nonce; the network is perfectly secure if the adversary arrives after this. In the proposed *trusted-neighbour key pre-distribution* scheme the memory space and the number of preloaded keys required for the network to be connected are reduced. Changing the link key to a random value implies that a node needs to have more memory for key management. Each node has to store at least d_{required} random values in addition to the preloaded keys; the value of d_{required} is a lower bound on the expected degree of the nodes to guarantee that the resulting network is connected with a high enough probability [11]. Let p_{required} be this probability, and let p_{actual} be the actual probability of establishing a secure link between a pair of neighbours. One needs $p_{\text{actual}} \geq p_{\text{required}}$ to hold true.

We define neighbours as trusted if their wireless communication coverage areas overlap and they share a secure link. Here we distinguish the probability $p_{\text{n-n}}$, that of a pair of neighbours being able to establish a secure link through preloaded key material, from the local connectivity p_{actual}, the probability a node to set up a secure link with its neighbours directly or indirectly with the help of trusted neighbours. The values of $p_{\text{n-n}}$ and p_{actual} are the same when only direct links are considered.

If more than one neighbour can participate in generating a secret key k_{AB} for the link between A and B, k_{AB} it is computed as $k_{AB} = v_1 \oplus v_2 \oplus \ldots \oplus v_q$, where $\oplus$ denotes the operation of bit-wise exclusive or, each v_i is a random value routed through the ith common neighbour in a path, and where q is the minimum number of trusted neighbours required to set up the secure link.

3.1. Calculating Local Connectivity

As the probability of any pair of two neighbouring nodes establishing a secure link is $p_{\text{n-n}}$, the probability of a common neighbour sharing a secure link with two neighbouring nodes is $p_{\text{n-n}}^2$. Let $p(i)$ be the probability that exactly i common neighbours share a secure link with neighbouring nodes. When there are n_c common neighbours, the number of different choices of common neighbours sharing a secure link with both nodes is $\binom{i}{n_c}$. Hence we have:

$$p(i) = \binom{i}{n_c} p_{\text{n-n}}^{2i} (1 - p_{\text{n-n}}^2)^{n_c - i}.$$

The probability p_{indirect} of establishing a secure link through trusted neighbours is the probability of sufficiently many neighbours sharing a secure link with the two nodes, hence

$$p_{\text{indirect}} = 1 - \sum_{i=0}^{q-1} p(i).$$

The local connectivity p_{actual} is the probability for a node to establish secure links with its neighbours directly or indirectly.

$$p_{\text{actual}} = p_{\text{n-n}} + (1 - p_{\text{n-n}}) \cdot p_{\text{indirect}}.$$

When $q = 1$, at most one trusted neighbour is required to set up a secure link with both nodes. To obtain a connected key-sharing graph the local connectivity p_{actual} should be at least p_{required}. According to [3] there are, on expectation, $0.5865n'$ nodes within the common wireless coverage area of two neighbouring nodes. Since $p_{\text{required}} = \frac{d_{\text{required}}}{n'}$, where n' is the expected number of neighbours, one needs

$$p_{\text{n-n}} + (1 - p_{\text{n-n}})(1 - (1 - p_{\text{n-n}}^2)^{0.5865n'}) \geq \frac{d_{\text{required}}}{n'}$$

to hold.

3.2. Connection Establishment

In the key discovery phase, each node first discovers, for each of its neighbours, all common keys preloaded before

network deployment. We use a more secure method similar to Merkle's puzzle system [9] as proposed in [11]. Each node generates k puzzles and broadcasts its list of puzzles to its neighbours. Every node which responds with a correct answer to a puzzle is identified as possessing the associated key. Let i be the identifier of the key K_i used to encrypt the puzzle. The ith puzzle is created as $E_{K_i}(a, i)$, where E_{K_i} is the encryption function using key K_i and a is a constant term which remains the same for all puzzles. To the beginning of the broadcast message the node attaches an identifier of its own and a in plain text to provide enough redundancy for a neighbouring node to identify that it has correctly decoded a puzzle and the common key it shares with the sender of the message. Since i is a local identifier to a node's key ring, it does not reveal any key-sharing patterns to an adversary. The identifier i will be included in the response message to the sender in plain text so as to identify the common key used. For any two neighbouring nodes failing to identify a common preloaded key, we need to find their trusted neighbours. This can be accomplished by multi-casting a list of neighbours with which a node did not discover a common key to each neighbour with which a common key is discovered. Each node compares the lists received with its own list of these neighbours with which it has established secure links. If there is a match, a random nonce should be sent to both nodes which will be used to secure the link between them. A simple implementation of this protocol requires two rounds of broadcasting for each node: first of the list of puzzles; second to broadcast the list of neighbours with which a secure link has not been establisghed in the initial key set-up.

However, a single round of broadcasting suffices [11]: A node can compare the list it received with the list of neighbours with which it does not have a key in common itself. Suppose, for example, that node A has five neighbours B, C, D, E, F with which it has keys in common. Node A generates $\binom{5}{3}$ random nonces for each pair of B, C, D, E and F. To node B, node A sends the list consisting of C, D, E and F, together with the random nonces associated with each of the pairs BC, BD, BE and BF. Similar messages will be sent to C, D, E and F. If nodes B and C are neighbours without a common key, they would use the same random nonce included in the messages from node A to them, respectively, as their link keys. Each node then waits for a certain period of time long enough to receive a message containing puzzles from the neighbours. If a puzzle is solved correctly, a node not only sends the common key identifier discovered but also information regarding other nodes with which a common key is discovered in the reply message. If node B solves the ith puzzle sent by node A, then node B replies to A in the following format

$$(i, E_{K_{ij}}(\mathrm{nonce}_B, \mathrm{list}(\mathrm{nid}_1, \mathrm{nonce}_{A,\mathrm{nid}_1}, \ldots))).$$

The key K_{ij} is a common key in the key rings of nodes A and B, respectively, of which i and j are the identifiers. The random value nonce_B generated by node B in the reply to A and the corresponding random value nonce_A generated by node A in the reply to B are used to create the *link key* as $\mathrm{nonce}_A \oplus \mathrm{nonce}_B$. The list of the identifiers $\mathrm{nid}_1, \mathrm{nid}_2, \ldots$ of the other neighbours with which node B has discovered a common key, together with their corresponding nonces $\mathrm{nonce}_{A,\mathrm{nid}_1}, \mathrm{nonce}_{A,\mathrm{nid}_2}, \ldots$ is attached to the message. Node A compares its neighbours with which it failed to establish a secure link in the initial key set-up with the list of nodes in the above message. If a neighbouring node C fails to set up a secure link with A and is listed in B's reply to A, node A will send a message to node C to confirm the link key created by node B as $(m, E_{\mathrm{nonce}_{AC}}(m))$, where m is random value. Node C should be able to use nonce_{AC} it received from node B to decrypt the message m. If m is successfully recovered, node C confirms that node A does indeed have the nonce_{AC} which will then be used to secure the link between nodes A and C. Each pair of nodes confirms the trusted neighbours they have identified within the set time. The random nonces routed from the commonly identified trusted neighbours will then be used to generate the link keys.

3.3. Security Analysis

In this section, we first analyze the relationship of the probability of two nodes sharing at least a common key $p_{\text{n-n}}$ and the network density n' to obtain a connected key-sharing graph when $q = 1$. Next we calculate the expected communication overhead of trusted-neighbour key pre-distribution protocol (for details see [11, 13]). As stated above, one needs

$$n' p_{\text{n-n}} + n'(1 - p_{\text{n-n}})(1 - (1 - p_{\text{n-n}}^2)^{0.5865n'}) \geq d_{\text{required}}$$

to obtain. Using $1 - p_{\text{n-n}} < 1$ one gets

$$\begin{aligned}(1 - p_{\text{n-n}})(1 - p_{\text{n-n}}^2)^{0.5865n'} &\leq (1 - p_{\text{n-n}}^2)^{0.5865n'} \\ &\leq \frac{n' - d_{\text{required}}}{n'}.\end{aligned}$$

As $\frac{d_{\text{required}}}{n'} \ll 1$, we use x to estimate $\ln(1 + x)$ and $1 - x$ to estimate e^{-x}. Thus the probability of two nodes sharing at least a common key when $q = 1$ satisfies

$$p_{\text{n-n}} \geq \sqrt{1 - \exp\left(\frac{\ln \frac{n' - d_{\text{required}}}{n'}}{0.5865n'}\right)} \approx \frac{d_{\text{required}}}{3n'}.$$

So, the required probability for establishing a secure connection is decreased to one third in the proposed scheme when compared to the EG scheme.

3.4. A Connectivity Example

The following simple example is intended to illustrate the effect of the nonce-based scheme when used with the EG scheme; the calculations are based on [11]. Assume a sensor network with $N = 10,000$ nodes deployed in an area 1000×1000 which each node has a wireless transmission range of $r = 40$. The expected number of neighbours of a node will be $n' = 48$. For achieving a global connectivity of $p_{\text{global}} = 0.999$ as determined by a random graph connectivity of $d_{\text{required}} = 16$ and the required probability it suffices to have $p_{\text{required}} = \frac{16}{48} = 0.33$. Given $K = 100,000$, a key ring size of $k = 200$ is needed to satisfy the required probability. In this case 16 neighbours are connected and 32 neighbours are not connected. Now $p_{\text{n-n}} \geq \frac{d_{\text{required}}}{3n'} = \frac{16}{3\times 48} = 0.11$ and the required probability of two nodes to share at least a common key is reduced to about one third compared to the EG scheme. This reduction decreases the number of keys needed to be preloaded to each node. We calculate the key ring size as $k = 106$. When each key has been updated to secure the link with each neighbour using a random nonce, each node is required to store $k + d_{\text{required}} = 106 + 16 = 122$ keys which is far less than 200 keys required by the EG scheme [11].

3.5. Resilience Against Node Capture

Resilience to node capture roughly means to which extent the damage of a node being captured can be controlled. If the secure link is set up through a trusted neighbour indirectly, an adversary can compromise the link if eavesdropping on at least one hop from the trusted neighbour to one node is possible. If the adversary has the probability $f_{\text{d}}(x)$ to compromise a direct link when x nodes are captured, the probability to compromise an indirect link set up through one trusted neighbour is $f_{\text{ind}}(x) = 2f_{\text{d}}(x) - f_{\text{d}}(x)^2$. If q trusted neighbours are involved, the probability to compromise an indirect link is $(2f_{\text{d}}(x) - f_{\text{d}}(x)^2)^q$. Let $f(x)'$ be the expected probability to compromise an additional link set up directly or indirectly when x nodes are captured. When $q = 1$, we have

$$f(x)' = \frac{n'p_{\text{n-n}}}{16} f_{\text{d}}(x) + \frac{(16 - n'p_{\text{n-n}})}{16} f_{ind}(x)\cdot$$

The value of $\frac{n'p_{\text{n-n}}}{16}$ can be estimated as $\frac{1}{3}$ and we have:

$$f(x)' = \frac{1}{3} f_{\text{d}}(x) + \frac{2}{3}(2f_{\text{d}}(x) - f_{\text{d}}(x)^2) \approx 1.6 f_{\text{d}}(x).$$

The expected fraction of total keys or links compromised in the EG scheme can be estimated as

$$f(x) = 1 - \left(1 - \frac{k}{K}\right)^x \approx \frac{k}{K}x$$

where K is the size of the key pool and k is the size of the key ring of a node with $k \ll K$. If the probability required for two nodes to share at least one key remains the same, the decrease of the ratio $\frac{k}{K}$ implies the decrease of k, the number of keys required to be preloaded. As seen in the example above with $n' = 48$, $k = 200$ and $d_{\text{required}} = 16$ as a given parametres, we calculate the security parametre as $\frac{k}{K} = 0.002$ which determines the scheme's resilience to node capture. The probability of sharing at least a common key between two nodes is $p_{\text{n-n}} = 0.11$. The ratio $\frac{k}{K}$ should be $\frac{0.001}{1.6} = 0.00125$ to obtain the same security performance $f(x)'$ as that of the EG scheme. Comparing the value of $p_{\text{n-n}} = 0.11$ with the corresponding value of 0.33 in te basic EG scheme, the security ratio $\frac{k}{K}$ reduces to $\frac{k'}{K}$ for k' as follows: For the pool size of $K = 100,000$, $k' = \frac{1}{2}k$ will satisfy the given required probability. Therefore, the resilience is significantyly improved.

3.6. Simulation Results

We performed simulations using the following parametres: $N = 10,000$, $K = 100,000$, deployment region of 1000×1000, wireless transmission range of a node $r = 40$ average number of neighbours $d = 48$.

Local Connectivity We compare our simulation results with the basic EG scheme [6] and the scheme using deployment knowledge [4]. We also integrated our trusted-neighbour key distribution scheme into the deployment knowledge scheme to evaluate the probability and performance. Figure 1 illustrates the local connectivity of these schemes. Note that the local connectivity is the probability of sharing a key between two nodes against various key ring sizes k. The simulation results match the analytical results presented above and of [11] according to which the trusted-neighbour key pre-distribution scheme improves the connectivity when the number of connected nodes increases. For $K = 100,000$ the connectivity rises sharply when $p_{\text{n-n}}$ is between 0.1 and 0.3. To achieve $p_{\text{n-n}} = 0.33$, our scheme needs 106 keys while the basic EG scheme needs 200 keys pre-loaded. Our scheme when integrated with the deployment knowledge outperforms the basic scheme and also more advanced ones. With the trusted-neighbour key distribution the scheme achieves the best connectivity. However, the scheme *per se* does not act like deployment knowledge and thus, requires a large key ring size to achieve a prescribed level of connectivity.

Resilience In Figure 2 the resilience is plotted agianst the number of nodes comprimised comparing the same schemes as above. The trusted-neighbour scheme reduces the key ring size required to provide the same security level as that of the basic scheme with the same network density

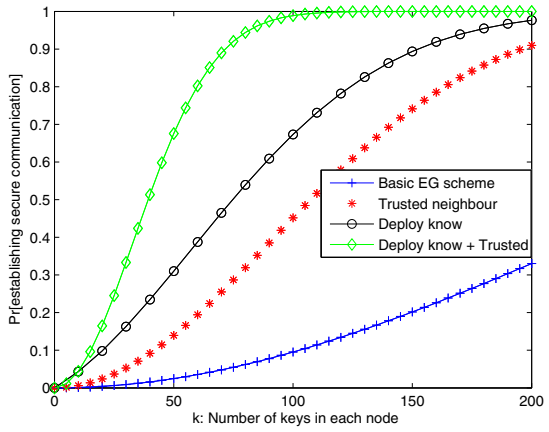

Figure 1. Local Connectivity

at least to a half, and it outperforms it significantly when it is combined with deployment knowlege [4].

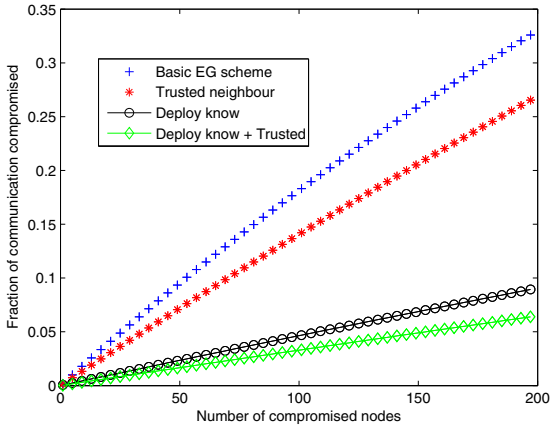

Figure 2. Network Resilience

3.7. Communication Overhead

We measure the communication requirements in terms of the lengths of the messages to communicate keys. A random value used to generate a link key is treated as one unit of message length. First, each node broadcasts k puzzles $E_{K_i}(a, i)$ to each neighbour together with its own node identifier and a random constant term a in plain text. This requires a total length of $k + 1$ on average. Next, the node which solves at least one puzzle replies in the format $(i, E_{K_{ij}}(\text{nonce}_B, \text{list}(\text{nid}_1, \text{nonce}_{A,\text{nid}_l}, \ldots)))$ with a total length of $1 + (n'p_{\text{n-n}} - 1) = n'p_{\text{n-n}}$ to $n'p_{\text{n-n}}$ nodes on expectation. Finally, a node confirms with each neighbour with which it is going to set up a secure link through a trusted neighbour in the format of $(m, E_{\text{nonce}_{AC}}(m))$ with a total length of 1 to $16 - n'p_{\text{n-n}}$ number of nodes on expectation. The communication overhead for each node on expectation to establish $d_{required} = 16$ secure links directly or indirectly with its neighbours during the key establishment phase is

$$\begin{aligned} &n'(k+1) + n'p_{\text{n-n}}n'p_{\text{n-n}} + (d_{required} - n'p_{\text{n-n}}) \\ &\quad = n'(k+1) + n'p_{\text{n-n}}(n'p_{\text{n-n}} - 1) + 16 \cdot \end{aligned}$$

In the basic EG scheme, the expected communication overhead for each node to establish d_{required} secure links with its neighbours during the key establishment phase is

$$n'(k+1) + d_{\text{required}} = n'(k+1) + 16.$$

As an example, compare two sensor networks providing the same security and the same network density $n' = 48$: $k = 200$ and $p_{\text{n-n}} = 0.33$ for the basic scheme; $k = 94$, $p_{\text{n-n}} = 0.11$ for the trusted neighbour key pre-distribution scheme when $q = 1$. In the basic scheme, the expected communication overhead is $48(200 + 1) + 16 = 9664$. In the trusted neighbour key pre-distribution, the expected communication overhead is $48 \cdot (94+1) + 48 \cdot 0.11 \cdot (48 \cdot 0.11 - 1) + 16 = 4597$. Therefore, the expected communication overhead using the trusted neighbour scheme when $q = 1$ is about half of the expected communication overhead using the basic scheme when the network density and the security are the same during the key establishment phase. In the trusted neighbour scheme $n'p_{\text{n-n}}$ can be estimated as $\frac{16}{3}$. Figure 1, shows that the key ring size using the trusted neighbour scheme is about half of the key ring size using the basic scheme when the network density and the security are the same. If the network density is n_1 and the key ring size is k_1 in the trusted-neighbour scheme, the expected communication overhead using the trusted-neighbour scheme over the expected communication overhead using the basic EG scheme is one half of that.

4. Conclusion

Under the assumption of a real-world attacker model, we propose to use random nonces to secure links after the key establishment phase. Using random nonces to secure network communication guarantees that the network is trivially secure if the adversary arrives when the key establishment phase is complete.

Exploiting the feature of dense sensor node scattering, we propose *trusted-neighbour reinforcement.* Our scheme reduces the memory requirement by half compared to the scheme of Eschenauer and Gligor and performs even better when it is integrated with deployment knowledge. Analyti-

cal and simulation results demonstrate significant improvements in local connectivity and resilience against node capture over the basic EG scheme.

Acknowledgements
This research was supported by the Natural Sciences and Engineering Council of Canada through a research grant to H. Jürgensen.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci: Wireless sensor networks: A survey. *Computer Networks* **38** (2002), 393–422.

[2] C. Blundo, A. de Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung: Perfectly-secure key distribution for dynamic conferences. In E. F. Brickell (editor): *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Lecture Notes in Computer Science* **740**, 471–486, Springer-Verlag, Berlin, 1993.

[3] H. Chan, A. Perrig, D. X. Song: Random key predistribution schemes for sensor networks. In *2003 IEEE Symposium on Security and Privacy (S&P 2003), 11-14 May 2003, Berkeley, CA, USA*. 197–213, IEEE Computer Society, Washington, 2003.

[4] W. Du, J. Deng, Y. S. Han, P. K. Varshney: A key predistribution scheme for sensor networks using deployment knowledge. *IEEE Transactions on Dependable Secure Computing* **3**(1) (2006), 62–77.

[5] P. Erdös, A. Rényi: On the evolution of random graphs. *Bulletin of the Institute of International Statistics* **38** (1961), 343–347.

[6] L. Eschenauer, V. D. Gligor: A key-management scheme for distributed sensor networks. In V. Atluri (editor): *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washingtion, DC, USA, November 18-22, 2002*. 41–47, ACM Press, New York, 2002.

[7] D. Liu, P. Ning: Establishing pairwise keys in distributed sensor networks. In S. Jajodia, V. Atluri, T. Jaeger (editors): *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washingtion, DC, USA, October 27-30, 2003*. 52–61, ACM Press, New York, 2003.

[8] D. Liu, P. Ning: Improving key predistribution with deployment knowledge in static sensor networks. *ACM Transactions on Sensor Networks* **1** (2005), 204–239.

[9] R. C. Merkle: A digital signature based on a conventional encryption function. In C. Pomerance (editor): *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings. Lecture Notes in Computer Science* **293**, 369–378, Springer-Verlag, Berlin, 1988.

[10] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, M. Galloway: A survey of key management schemes in wireless sensor networks. *Computer Communications* **30** (2007), 2314–2341.

[11] Y. Xue: *Key Management Schemes for Distributed Sensor Networks*. PhD thesis, The University of Western Ontario, 2008.

[12] Y. Xue, H. Jürgensen: Maximum supported network coverage area and cost evaluation of key predistribution schemes. In F. Makedon, L. Bailie (editors): *Proceedings of the 1st ACM International Conference on Pervasive Technologies Related to Assistive Environments, PETRA 2008, Athens, Greece, July 16–18, 2008. ACM International Conference Proceedings Series* **282**, Association for Computing Machinery, New York, 2008. [Article No. 42], 8 Pages.

[13] Y. Xue, H. Jürgensen. Secure paths in wireless networks with real-world attackers, 2008. Manuscript, in preparation.