© de Gruyter 2008
J. Math. Crypt. **2** (2008), 393–403 DOI 10.1515 / JMC.2008.018

# Two attacks on a sensor network key distribution scheme of Cheng and Agrawal

M. B. Paterson and D. R. Stinson

Communicated by Rainer Steinwandt

**Abstract.** A sensor network key distribution scheme for hierarchical sensor networks was recently proposed by Cheng and Agrawal. A feature of their scheme is that pairwise keys exist between any pair of high-level nodes (which are called cluster heads) and between any (low-level) sensor node and the nearest cluster head. We present two attacks on their scheme. The first attack can be applied for certain parameter sets. If it is applicable, then this attack can result in the compromise of most if not all of the sensor node keys after a small number of cluster heads are compromised. The second attack can always be applied, though it is weaker.

**Keywords.** Wireless sensor networks, key distribution.

**AMS classification.** 94A60.

## 1 Introduction

There has been considerable recent interest in sensor networks that have a hierarchical architecture. A commonly-studied model (see, for example, [10, 5]) is to assume the existence of a powerful base station, a number of $m$ high-level nodes (called *cluster heads*) and a larger number, $n$, of (low level) *sensor nodes*. Typical values for these parameters are $n = 10000$ and $m = 100$.

After deployment, any two cluster heads are assumed to be able to communicate directly. A sensor node is only required to communicate with the nearest cluster head. It is assumed that these communications can all be done directly (no intermediate nodes required). It is also assumed that $n/m$ sensor nodes are deployed in the vicinity of each cluster head. Additional details of this model can be found in [10, 5].

It is not assumed that cluster heads are tamperproof, and therefore there is the possibility that cluster heads might be compromised. The attack model is the standard "node capture" model. The adversary can observe all communications that take place between nodes in the network, and the adversary can capture a number of nodes and extract all the keys that are stored in them. We will mainly focus on a special attack where the adversary compromises $s$ out of the $m$ cluster heads.

There have been many proposals for key distribution protocols for sensor networks. See [3, 4, 7, 9, 12, 13] for several different approaches to this problem. The special case of hierarchical networks has also received considerable attention, and key distribution schemes for hierarchical networks have been presented in [5, 6, 8, 10, 14, 15, 17, 18].

The first author's research is supported by EPSRC grant EP/D053285/1.
The second author's research is supported by NSERC discovery grant 203114-06.

In this paper, we present an attack on the scheme proposed by Cheng and Agrawal [5]. This scheme can be viewed as a generalization of a well-known scheme due to Jolly, Kuscu, Kokate and Younis ([10]). A main motivation for the Jolly-Kuscu-Kokate-Younis scheme is the low energy consumption overhead. Cheng and Agrawal's modification is intended to provide enhanced security, even when cluster heads are not assumed to be tamperproof and they can be compromised by an adversary.

First, we summarize how the Cheng-Agrawal scheme works. Denote the cluster heads by $C_1, \ldots, C_m$ and the sensor nodes by $S_1, \ldots, S_n$. [5] assumes that there is a predeployed pairwise key between the base station and every other node (a *pairwise key* is a key that is held by exactly two nodes). The main objective of [5] is to describe how pairwise keys are created between

- any two cluster heads (we call these *cluster head keys*), and
- any sensor node and the closest cluster head (we will refer to these keys as *sensor node keys*).

The first objective is accomplished by using a Blom Scheme [1, 2]. We briefly summarize the construction of a *$t$-resilient Blom Key Predistribution Scheme* as it would be applied in the context of setting up pairwise keys for the $m$ cluster heads. In what follows, we assume for simplicity that the IDs of the $m$ cluster heads are $1, \ldots, m$, respectively. We will denote by $K^C_{i,j}$ the cluster head key that is held by $C_i$ and $C_j$. Here is the Blom Scheme:

**step 1** A prime number $p$ is chosen, and for $0 \leq i, j \leq t$, random elements $a_{i,j} \in \mathbb{Z}_p$ are chosen such that $a_{i,j} = a_{j,i}$ for all $i, j$. Then define the polynomial

$$f(x, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{i,j}\, x^i y^j \bmod p.$$

**step 2** For each cluster head $C_r$, define the polynomial

$$g_{C_r}(x) = f(x, r) \bmod p = \sum_{i=0}^{t} a_{r,i}\, x^i.$$

**step 3** For each cluster head $C_r$, the coefficient vector $(a_{r,0}, \ldots, a_{r,t}) \in (\mathbb{Z}_p)^{t+1}$ is stored in $C_r$.

**step 4** For any two cluster heads $C_r$ and $C_s$, the pairwise key is $K^C_{r,s} \in \mathbb{Z}_p$ is defined to be $K^C_{r,s} = f(r, s)$, where $C_r$ computes

$$K^C_{r,s} = g_{C_r}(s)$$

and $C_s$ computes

$$K^C_{r,s} = g_{C_s}(r).$$

From well-known properties of the Blom Scheme (see, for example [16, Ch. 10]), it follows that no set of $t$ cluster heads can determine any information about the pairwise key of two other cluster heads, but any set of $t + 1$ cluster heads can compute all the pairwise keys in the scheme. In order to provide complete resilience of the cluster node keys against any compromise of other cluster heads, it is suggested in [5] to set the resilience of the Blom Scheme (namely, $t$) to be higher than the number of cluster nodes (which is denoted by $m$). However, in this situation, it would be simpler and more efficient (from the point of view of storage, see step 3) just to predeploy independent pairwise keys between any two cluster heads, without using a Blom Scheme. However, this modification does not affect the security of the scheme, nor does it affect the attacks we will describe.

The second objective is realized using an "improved key distribution mechanism" (IKDM) described in [5, §3]. Each sensor node $S_i$ is given one key, say $K_i$, before deployment. $S_i$ is also given a list of $\ell$ identifiers of cluster heads, say $B_i \subseteq \{1, \ldots, m\}$. $K_i$ is computed as the sum of $\ell$ shares, each of which can be computed by one of the cluster heads identified in $B_i$ (recall that the number of cluster heads is $m$, so we assume that $m \geq \ell$).

Additional Blom Schemes are also used in the IKDM. In fact, a different Blom scheme is associated with each cluster head. However, the bivariate polynomials associated with any of these Blom Schemes always have the first variable set equal to the ID of that cluster head. So there is no point in using bivariate polynomials for the cluster heads; it suffices for a different degree $t$ univariate polynomial to be associated with each cluster head. These polynomials, which will have coefficients defined over the finite field $\mathbb{Z}_p$, will be termed *CH-polynomials*. The CH-polynomial assigned to $C_j$ will be denoted by $g_j(x)$.

Now, each share of a key $K_i$ is computed by evaluating a CH-polynomial at the point $i$. To be precise, $K_i$ is defined as follows:

$$K_i = \sum_{j \in B_i} g_j(i), \tag{1}$$

where the terms $g_j(i)$ are the *shares* of $K_i$. The shares and the keys are all elements of $\mathbb{Z}_p$.

After deployment, the following protocol is carried out so the nearest cluster head to $S_i$, say $C_p$, can learn the value of the key $K_i$.

**step 1** $S_i$ sends the list $B_i$ to $C_p$ (it is possible, but not required that $p \in B_i$).

**step 2** For every $j \in B_i$, $j \neq p$, $C_p$ obtains an encrypted share from $C_j$. That is, $C_j$ computes $s_j = e_{K^C_{j,p}}(g_j(i))$ and sends $s_j$ to $C_p$ (observe that share $s_j$ is encrypted with the cluster head key $K^C_{j,p}$). If $p \in B_i$, then $C_p$ computes the share $g_p(i)$ by itself.

**step 3** $C_p$ decrypts all the encrypted shares and computes the sum (1) to get $K_i$. Now $S_i$ and $C_p$ have a pairwise key.

The scheme in [10] is basically the case $\ell = 1$ of the Cheng-Agrawal scheme. In this situation, each sensor node key has only one share, namely the key itself.

The authors of [5] argue that because all the keys in their scheme are pairwise keys, the network is resilient to node compromise (even when allowing compromise of cluster heads). They say "Even if all the 100 cluster heads are compromised, none of the keys preloaded in the sensor nodes could be compromised in the network". However, this cannot be true, because the keys preloaded in the sensor nodes can be computed from the information that is stored in the cluster heads. In fact, Das and Sengupta [6] observe that the compromise of $s$ cluster heads, after the IKDM process has terminated, will result in the compromise of $100s$ of the sensor node keys.

### 1.1 Our contributions

We describe two attacks on the Cheng-Agrawal scheme in this paper. In Section 2, we present an attack that we call the "interpolation attack". In this attack, the compromise of a small number of cluster heads (after the IKDM process is completed) can result in the compromise of all or almost all of the sensor node keys in the network. The interpolation attack can possibly be thwarted by a careful choice of the parameters of the scheme. However, we describe another attack in Section 3; this attack is called the "reconstruction attack". The reconstruction attack can always be applied, though it is usually a weaker attack than the interpolation attack, in the sense that it will not result in the compromise of all the keys in the network.

## 2 Interpolation attack

Suppose an adversary records the communications that take place during the IKDM. Then the adversary compromises $s$ out of the $m$ cluster heads (we will assume that $s < m$, because the compromise of all $m$ cluster heads clearly reveals all the sensor node keys). This allows the adversary to decrypt all the messages that were sent to these $s$ cluster heads during the IKDM. After their decryption, the adversary has information pertaining to various CH-polynomials evaluated at various points. If any CH-polynomial has been evaluated at at least $t + 1$ points, then the polynomial can be reconstructed using Lagrange interpolation, e.g., as is done in Shamir secret sharing (see, for example [16, Ch. 13]). So the adversary can potentially recover many CH-polynomials by compromising a small number of cluster heads.

We now present an attack that we call the "interpolation attack". The attack has two phases, as follows:

**Phase I**

Capture $s$ cluster heads and recover the keys stored in them. Use these keys to decrypt all the encrypted shares sent to these $s$ cluster heads during the IKDM process. Then interpolate the obtained shares (using Lagrange interpolation) to recover CH-polynomials .

**Phase II**

Use the recovered CH-polynomials to compute sensor node keys.

## 2.1 Phase I of the attack

In this section, we discuss phase I of the interpolation attack. Recall that each sensor node $S_i$ contains a list $B_i$ consisting of $\ell$ of the $m$ cluster heads. This list is sent in the clear to a cluster head, so it is known to the adversary. We assume each list is a random $\ell$-subset (which we will call a *block*) of the $m$ *points* in the set $\{1, \ldots, m\}$ (i.e., cluster head IDs). By compromising $s$ cluster heads, the adversary gets $sn/m$ such blocks. The average number of occurrences of a point $x \in \{1, \ldots, m\}$ in the $sn/m$ blocks is $sn\ell/m^2$.

The idea of our attack is to compromise a sufficient number $s$ of cluster heads so that the average number of occurrences of a point in the $sn/m$ blocks is 25% higher than $t$. To be concrete, we suppose that we choose $s$ so that the following holds:

$$\frac{sn\ell}{m^2} = 1.25t, \tag{2}$$

Because we require $s < m$ and we also want (2) to be satisfied, it must be the case that $\ell > 1.25mt/n$.

There is nothing "magic" about the choice of the constant 1.25 in (2). If this constant is made larger, it would increase the success probability of our attack (but more cluster heads would be have to be compromised). Smaller values of this constant would decrease the success probability.

Having chosen $s$ so that (2) holds, we will argue that almost every point occurs more than $t$ times in the set of $sn/m$ blocks. This is proven by using a standard tail inequality for binomial distributions, which can be found in [11, p. 502 ], for example.

**Lemma 2.1.** *Suppose $X_1, \ldots, X_N$ are independent random variables such that*

$$\Pr[X_i = 1] = p \textit{ and } \Pr[X_i = 0] = 1 - p$$

*for all $i$. Define $X = X_1 + \cdots + X_N$. Then*

$$\Pr[X \leq N(p - \epsilon)] \leq e^{-\epsilon^2 N/(2p)}. \tag{3}$$

Note that $Np$ is the expected value of $X$, so this estimate gives an upper bound on the probability that $X$ is somewhat below its expectation.

We will apply the inequality (3), setting $N = sn/m$, $p = \ell/m$, and $\epsilon = .2\ell/m$. Simplifying and using (2), we get

$$\Pr[X \leq t] \leq e^{-.025t}.$$

Define a point to be *good* if it occurs at least $t + 1$ times in $s$ random $\ell$-subsets of $\{1, \ldots, m\}$. We have shown that, if $s = 1.25tm^2/(n\ell)$, then any given point is good with probability at least $1.0 - e^{-.025t}$. By linearity of expectation, it follows that the expected number of good points is at least $m(1.0 - e^{-.025t})$ under these assumptions. For each good point $j$, the adversary can reconstruct the polynomial $g_j(x)$. Therefore, we have the following theorem.

**Theorem 2.2.** *Suppose the hierarchical sensor network has $m$ cluster heads, $n$ sensor nodes, each sensor node is given $\ell$ random IDs of cluster heads, and sensor node keys are defined using CH-polynomials of degree $t$. If an adversary compromises $s = 1.25tm^2/(n\ell)$ cluster heads after the IKDM process, then the expected number of CH-polynomials that can be reconstructed using the interpolation attack is at least $m(1.0 - e^{-.025t})$.*

We present an example to illustrate the application of Theorem 2.2.

**Example 2.3.** The parameters suggested in [5] are $m = 100$, $n = 10000$ and $t = 128$. [5] does not discuss appropriate values for $\ell$ except to say that "To achieve sufficient security, large $\ell$ is desired" ([5, p. 42]). In order to apply Theorem 2.2, we choose $s = 160/\ell$. Then

$$\Pr[X \leq 128] \leq e^{-3.2} \approx .04076.$$

Therefore the interpolation attack recovers (on average) at least 96 of the 100 CH-polynomials by compromising $160/\ell$ cluster heads.

Note that phase I of the interpolation attack becomes easier as $\ell$ gets bigger. If $\ell = 10$, then we take $s = 16$; if $\ell = 20$, then we take $s = 8$, etc. That is, as $\ell$ is increased, the number of compromised cluster heads required by the attack decreases.

In practice, the interpolation attack will probably work better than the estimates derived above would indicate. This is because the inequality (2) overestimates the tail probability in the relevant binomial distribution. For specified values of the parameters, it is a simple matter to compute the tail probability exactly. This is illustrated in the next example.

**Example 2.4.** We use the same parameters as in the previous example: $m = 100$, $n = 10000$ and $t = 128$. Then we can compute $\Pr[X \leq 128]$ exactly using the following formula:

$$\Pr[X \leq 128] = \sum_{j=0}^{128} \binom{100s}{j} \left(\frac{\ell}{100}\right)^j \left(1 - \frac{\ell}{100}\right)^{100s-j}. \tag{4}$$

For example, when $\ell = 20$ and $s = 8$, the formula (4) yields .00218, as compared to the estimate (3) of .04076. When $\ell = 10$ and $s = 16$, the exact value is about .00349, as compared to the estimate of .04076. The expected number of reconstructable CH-polynomials in the interpolation attack is $100(1.0 - \Pr[X \leq 128])$.

## 2.2 Phase II

Now we turn to the second phase of the interpolation attack. Suppose the adversary has recovered $r$ of the $m$ CH-polynomials. Then the adversary can compute the key for a particular sensor node if the block corresponding to that node is a subset of the $r$ points corresponding to the recovered polynomials. This probability is easily seen to be

$$\frac{\binom{r}{\ell}}{\binom{m}{\ell}}. \tag{5}$$

**Table 1.** Expected number of sensor node keys that can be compromised

| number of recovered CH-polynomials ($r$) | $\ell = 10$ | $\ell = 20$ | $\ell = 40$ |
|---|---|---|---|
| expected value of $r$ | 99.65 | 99.78 | 99.94 |
| 95 | 5837 | 3193 | 725 |
| 96 | 6516 | 4033 | 1243 |
| 97 | 7265 | 5081 | 2116 |
| 98 | 8090 | 6383 | 3575 |
| 99 | 9000 | 8000 | 6000 |
| 100 | 10000 | 10000 | 10000 |

The following theorem is an immediate consequence of (5).

**Theorem 2.5.** *Suppose the hierarchical sensor network has $m$ cluster heads, $n$ sensor nodes, and each sensor node is given $\ell$ random IDs of cluster heads. Suppose that $r$ CH-polynomials are reconstructed during phase I of the interpolation attack. Then the expected number of sensor node keys that can be computed in phase II of the interpolation attack is*

$$\frac{n\binom{r}{\ell}}{\binom{m}{\ell}}.$$

If $r < m$, then it is clear that there will (probably) be some keys that are not compromised. In phase II of the interpolation attack, the number of uncompromised keys increases as $\ell$ increases. However, it is very likely that phase I will recover all $m$ of the CH-polynomials (i.e., $r = m$), in which case all $n$ sensor node keys can be compromised. We show some computations in the next example.

**Example 2.6.** We use the same parameters as in the previous examples: $m = 100$, $n = 10000$ and $t = 128$.

In Table 1, we determine the expected number of sensor node keys that can be compromised, for $\ell = 10$, 20 and 40, computed as a function of the number of CH-polynomials, denoted by $r$, that are reconstructed during the first phase of the attack. We also indicate the expected number of reconstructed CH-polynomials when $s = 160/\ell$ cluster heads are compromised during phase I. These values are computed using the formula (4), as in Example 2.4.

## 3 The reconstruction attack

We have already noted that the interpolation attack described in the previous section can be mounted only when $\ell > 1.25mt/n$. It is of interest to point out a weaker attack that can be carried out for any values of the parameters. We call this the "reconstruction attack". The interpolation attack only used the information received by the compromised cluster heads. In the reconstruction attack, we make use of the information transmitted by the compromised cluster heads.

As before, we assume that $s$ of the $m$ cluster heads are compromised after the IKDM process has completed. We mentioned in Section 1 that [6] observed that the adversary can immediately obtain the $sn/m$ sensor node keys that are stored in the $s$ compromised cluster heads. We say that these sensor keys have been *directly compromised.*

In this section, we point out that some additional sensor node keys can be (possibly) be compromised by reconstructing them from compromised shares. Let $\mathcal{J} = \{j_1, \ldots, j_s\}$ denote the set of IDs of the $s$ compromised cluster heads. Suppose $S_i$ is a sensor node whose nearest cluster head, say $C_p$, has not been compromised (hence $p \notin \mathcal{J}$). Suppose it happens that $B_i \subseteq \mathcal{J}$. Then all $\ell$ shares that were used to compute $K_i$ were encrypted with cluster head keys that have been compromised. Therefore the adversary can compute $K_i$. In this situation, we say that the sensor node key $K_i$ has been *reconstructed.*

Now, the probability that $B_i \subseteq \mathcal{J}$ is

$$\frac{\binom{s}{\ell}}{\binom{m}{\ell}}.$$

There are $n - sn/m = n(m-s)/m$ sensor nodes whose nearest cluster head has not been compromised. Therefore, the expected number of reconstructed sensor node keys is

$$\frac{n(m-s)\binom{s}{\ell}}{m\binom{m}{\ell}}.$$

The following theorem is now obvious.

**Theorem 3.1.** *Suppose the hierarchical sensor network has $m$ cluster heads, $n$ sensor nodes, and each sensor node is given $\ell$ random IDs of cluster heads. Suppose that $s$ cluster heads are compromised. Then the expected number of sensor node keys that can be compromised as a result of a reconstruction attack is*

$$\frac{n}{m}\left(s + \frac{(m-s)\binom{s}{\ell}}{\binom{m}{\ell}}\right). \tag{6}$$

When we set $\ell = 1$ and simplify (6), the total number of compromised sensor node keys is

$$\frac{sn}{m}\left(2 - \frac{s}{m}\right). \tag{7}$$

**Remark:** Because the scheme in [10] is essentially the case $\ell = 1$ of the Cheng-Agrawal scheme, it follows that this attack can also be applied to the scheme in [10].

**Example 3.2.** Suppose that $n = 10000$, $m = 100$ and $t = 160$. The interpolation attack is applicable only if $\ell > 2$, However, when $\ell = 1$ or 2, then we can use the reconstruction attack.

From (7), the expected number of compromised sensor node keys when $\ell = 1$ is $100s(2 - s/100)$. If $s = 10$, for example, then we expect to compromise 1900 sensor node keys. That is, compromising 10% of the cluster heads results in 19% of the sensor node keys being compromised.

When $\ell = 2$, the expected number of compromised sensor node keys can be computed from (6); it is $100s + (100 - s)s(s - 1)/99$. If we again take $s = 10$, then we expect to be able to compromise 1082 sensor node keys. So compromising 10% of the cluster heads results in 10.8% of the sensor node keys being compromised.

## 3.1 Analysis

The interpolation and reconstruction attacks can be mitigated by a careful choice of parameters. It is clear from Example 3.2 that the reconstruction attack is much less effective when $\ell \geq 2$ than it is when $\ell = 1$. So an appropriate strategy might be to choose $\ell = 2$ and $t = 1.6n/m$. This would prevent the interpolation attack from being applied.

To measure the effectiveness of the reconstruction attack when $\ell = 2$, we consider the ratio of the number of reconstructed sensor node keys to the number of directly compromised sensor node keys. This ratio is easily computed to be

$$\frac{(m - s)(s - 1)}{m(m - 1)}.$$

This ratio is maximized by setting $s = (m + 1)/2$, in which case the ratio is approximately $1/4$. For this value of $s$, about $n/2$ sensor node keys are directly compromised, and an additional $n/8$ sensor node keys (approximately) are reconstructed.

## 4 Conclusion

In the communication model studied in [5], each sensor node communicates with only one cluster head. This has the advantage that sensor nodes do not have to communicate with each other. However, an unavoidable consequence is that the compromise of $s$ cluster heads will result in the compromise of $sn/m$ sensor node keys. Therefore the best we can hope for is to ensure that no additional sensor node keys are compromised.

There is a straightforward way to ensure this if cluster heads are permitted to communicate with the base station during the key establishment phase. Each sensor node $S_i$ will send its ID to the nearest cluster head. Then the cluster head forwards the sensor node ID to the base station and the base station encrypts the key $K_i$ and sends it to the cluster head. Finally, the cluster head decrypts $K_i$.

This approach might not be acceptable in some application scenarios. For example, the base station might not be available during the key establishment phase for some reason. In such a situation, we would be required to use a protocol where cluster heads communicate with each other, such as the Cheng-Agrawal scheme. If this scheme is to be used, then it is important to choose parameters in such a way that the consequences of the possible attacks are minimized. Our suggestion is to divide each key into only two shares. This provides a good level of security under appropriate parameter choices and it also requires lower communication complexity than if keys are split into larger number of shares.

## References

[1] Rolf Blom, *An Optimal Class of Symmetric Key Generation Systems*. Proc. of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques, pp. 335–338. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[2] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung, *Perfectly-Secure Key Distribution for Dynamic Conferences*. CRYPTO (Ernest F. Brickell, ed.), Lecture Notes in Computer Science 740, pp. 471–486. Springer, 1992.

[3] Seyit A. Çamtepe and Bülent Yener, *Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks*, IEEE/ACM Transactions on Networking 15 (2007), pp. 346–358.

[4] Haowen Chan, Adrian Perrig, and Dawn Xiaodong Song, *Random Key Predistribution Schemes for Sensor Networks*. IEEE Symposium on Security and Privacy, pp. 197–213. IEEE Computer Society, 2003.

[5] Yi Cheng and Dharma P. Agrawal, *An Improved Key Distribution Mechanism for Large-Scale Hierarchical Wireless Sensor Networks*, Ad Hoc Networks 5 (2007), pp. 35–48.

[6] Ashok Kumar Das and Indranil Sengupta, *An Effective Group-Based Key Establishment Scheme for Large-Scale Wireless Sensor Networks Using Bivariate Polynomials*. COMSWARE, pp. 9–16. IEEE, 2008.

[7] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili, *A Pairwise Key Predistribution Scheme for Wireless Sensor Networks*, ACM Transactions on Information and System Security 8 (2005), pp. 228–258.

[8] Xiaojiang Du, Yang Xiao, Mohsen Guizani, and Hsiao-Hwa Chen, *An Effective Key Management Scheme for Heterogeneous Sensor Networks*, Ad Hoc Networks 5 (2007), pp. 24–34.

[9] Laurent Eschenauer and Virgil D. Gligor, *A Key-Management Scheme for Distributed Sensor Networks*. CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 41–47. ACM, New York, NY, USA, 2002.

[10] Gaurav Jolly, Mustafa C. Kusçu, Pallavi Kokate, and Mohamed F. Younis, *A Low-Energy Key Management Protocol for Wireless Sensor Networks*. ISCC, pp. 335–340. IEEE Computer Society, 2003.

[11] Donald E. Knuth, *The Art of Computer Programming, Volume 1 (3rd ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[12] Jooyoung Lee and Douglas R. Stinson, *On the Construction of Practical Key Predistribution Schemes for Distributed Sensor Networks Using Combinatorial Designs*, ACM Transactions on Information and System Security 11 (2008).

[13] Donggang Liu, Peng Ning, and Rongfang Li, *Establishing Pairwise Keys in Distributed Sensor Networks*, ACM Transactions on Information and System Security 8 (2005), pp. 41–77.

[14] B. Maala, Y. Challal, and A. Bouabdallah, *HERO: Hierarchical kEy management pRotocol for heterOgeneous wireless sensor networks*. Wireless Sensor and Actor Networks II, IFIP International Federation for Information Processing 264, pp. 125–136. Springer, 2008.

[15] Leonardo B. Oliveira, Hao C. Wong, M. Bern, Ricardo Dahab, and A. A. F. Loureiro, *SecLEACH – A Random Key Distribution Solution for Securing Clustered Sensor Networks*. NCA '06: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications, pp. 145–154. IEEE Computer Society, Washington, DC, USA, 2006.

[16] Douglas R. Stinson, *Cryptography: Theory and Practice, Third Edition*. Chapman & Hall/CRC, 2006.

[17] Gelareh Taban and Reihaneh Safavi-Naini, *Key Establishment in Heterogeneous Self-organized Networks*. ESAS (Frank Stajano, Catherine Meadows, Srdjan Capkun, and Tyler Moore, eds.), Lecture Notes in Computer Science 4572, pp. 58–72. Springer, 2007.

[18] Patrick Traynor, Raju Kumar, Heesook Choi, Guohong Cao, Sencun Zhu, and Thomas F. La Porta, *Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks*, IEEE Transactions on Mobile Computing 6 (2007), pp. 663–677.

Received 28 July, 2008; revised 27 November, 2008

**Author information**

M. B. Paterson, Information Security Group, Department of Mathematics, Royal Holloway, University of London, Egham, Surrey TW20 0EX, U.K..
Email: `m.b.paterson@rhul.ac.uk`

D. R. Stinson, David R. Cheriton School of Computer Science, University of Waterloo, ON, N2L3G1, Canada.
Email: `dstinson@uwaterloo.ca`