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

M. B. Paterson*
Department of Mathematics
Royal Holloway
University of London
Egham, Surrey TW20 0EX, U.K.
m.b.paterson@rhul.ac.uk

D. R. Stinson†
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
dstinson@uwaterloo.ca

July 28, 2008

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.

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.

*Research supported by EPSRC grant EP/D053285/1
†Research supported by NSERC discovery grant 203114-06

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, 16, 17].

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 the scheme due to Jolly *et al* ([10]). 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

1. any two cluster heads (we call these *cluster head keys*), and

2. 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], though this seems to be unnecessary. The resilience of the Blom scheme, t, is set higher than the number of cluster nodes, m, so it would be simpler and more efficient just to predeploy pairwise keys between any two cluster heads. We will denote by $K_{i,j}^C$ the cluster head key that is held by C_i and C_j. This is a pairwise key, and its value cannot be determined, even if all the other $m-2$ cluster heads are compromised. (This modification does not affect the security of the scheme, nor does it affect our attack.)

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 ℓ identifiers of cluster heads, say $B_i \subseteq \{1, \ldots, m\}$[1]. K_i is computed as the sum of ℓ 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 $m \geq \ell$).

A different Blom scheme is associated with each cluster head. However, the bivariate polynomial associated with any cluster head always has 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. In any event, there is a different degree t univariate polynomial assigned to each cluster head. These polynomials, which will have coefficients defined over some finite field $\mathbb{F}_q$, 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 $\mathbb{F}_q$.

After deployment, a protocol is carried out so the nearest cluster head to S_i, say C_p, can learn the value of the key K_i. S_i sends the list B_i to C_p (it is possible, but not required that $p \in B_i$). For every $j \in B_i$, $j \neq p$, C_p obtains an encrypted share from C_j. That is, C_j computes $s_j = e_{K_{j,p}^C}(g_j(i))$ and sends s_j to C_p (observe that share s_j is encrypted with the cluster head key $K_{j,p}^C$). If $p \in B_i$, then C_p computes the share $g_p(i)$ by itself. Then 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.

[1] We assume for simplicity that the IDs of the m cluster heads are $1, \ldots, m$. This is irrelevant to the attack we will describe.

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". This is true before the IKDM process takes place; however, [5] does not really discuss the security of the IKDM process. Therefore, it is not clearly stated what kind of security guarantees are provided by their protocol. 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 [15, 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 ℓ 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 ℓ-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 $sn\ell/m^2$. If

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

then we can show that almost every point will occur more than t times[2]. This is proven by using a standard tail inequality for binomial distributions. This inequality can be found in [11, p. 502], for example.

Lemma 2.1. *Suppose $X_1, \ldots, X_N$ are independent random variables such that $\Pr[X_i = 1] = p$ 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 ℓ-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 ℓ 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 2.1. *The parameters suggested in [5] are $m = 100$, $n = 10000$ and $t = 128$. [5] does not discuss appropriate values for ℓ except to say that "To achieve sufficient security, large ℓ 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.

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

Note that phase I of the interpolation attack becomes easier as ℓ gets bigger. If $\ell = 10$, then we take $s = 16$; if $\ell = 20$, then we take $s = 8$, etc. That is, as ℓ 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 2.2. *We use the same parameters as in the previous example: $m = 100$, $n = 10000$ and $t = 128$. Then we can compute $\Pr[X \le 128]$ exactly using the following formula:*

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

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 $100(1.0 - \Pr[X \le 128])$.

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

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

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

Theorem 2.3. *Suppose the hierarchical sensor network has m cluster heads, n sensor nodes, and each sensor node is given ℓ 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 ℓ 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 2.3. *We use the same parameters as in the previous examples: $m = 100$, $n = 10000$ and $t = 128$.*

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

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 2.2.

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 compromise 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 ℓ 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 ℓ 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 3.1. *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.

4 Analysis and Discussion

The interpolation and reconstruction attacks can be mitigated by a careful choice of parameters. It is clear from Example 3.1 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.

In the communication model studied in [5], each sensor node communicates with only one cluster head. So it is unavoidable 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.

References

[1] R. Blom. An optimal class of symmetric key generation systems. *Lecture Notes in Computer Science* **209** (1985), 335–338 (EUROCRYPT '84 Proceedings).

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung. Perfectly secure key distribution for dynamic conferences. *Lecture Notes in Computer Science* **740** (1993), 471–486 (CRYPTO '92 Proceedings).

[3] S.A. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking* **15** (2007), 346–358.

[4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, May 2003, pp. 197–213.

[5] Y. Cheng and D. P. Agrawal. An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. *Ad Hoc Networks* **5** (2007), 35–48.

[6] A. K. Das and I. Sengupta. An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials. *3rd International Conference on Communication Systems Software and Middleware (COMSWARE 2008)*, pp. 9–16.

[7] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security* **8** (2005), 228–258.

[8] X. Du, Y. Xiao, M. Guizanic and H.-H. Chen. An effective key management scheme for heterogeneous sensor networks. *Ad Hoc Networks* **5** (2007), 24–34.

[9] L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM conference on Computer and Communications Security*, November 2002, pp. 41–47.

[10] G. Jolly, M. C. Kuscu, P. Kokate and M. Younis. A low-energy key management protocol for wireless sensor networks. *Eighth IEEE International Symposium on Computers and Communication, (ISCC 2003)*, pp. 335–340.

[11] D. E. Knuth. The Art of Computer Programming, Volume 1, Fundamental Algorithms, Third Edition. Addison-Wesley, 1997.

[12] J. Lee and D. R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Transactions on Information and System Security* **11-2** (2008), article No. 1, 35 pp.

[13] D. Liu, P. Ning and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security* **8** (2005), 41–77.

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

[15] D. R. Stinson. *Cryptography Theory and Practice, Third Edition*. Chapman & Hall/CRC, 2006.

[16] G. Taban and R. Safavi-Naini. Key establishment in heterogeneous self-organized networks. *Lecture Notes in Computer Science* **4572** (2007), 58–72 (ESAS 2007).

[17] P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu and T. La Porta. Efficient hybrid security mechanisms for heterogeneous sensor networks. *IEEE Transactions on Mobile Computing* **6** (2007), 663–677.