A Random Perturbation-Based Scheme for Pairwise Key Establishment in Sensor Networks[*]

Wensheng Zhang and Minh Tran
Dept. of Computer Science
Iowa State University
Ames, IA 50014, USA
{wzhang,ttminh}@cs.iastate.edu

Sencun Zhu and Guohong Cao
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802, USA
{szhu,gcao}@cse.psu.edu

ABSTRACT

A prerequisite for secure communications between two sensor nodes is that these nodes exclusively share a pairwise key. Although numerous pairwise key establishment (PKE) schemes have been proposed in recent years, most of them have no guarantee for direct key establishment, no resilience to a large number of node compromises, no resilience to dynamic network topology, or high overhead. To address these limitations, we propose a novel *random perturbation-based (RPB) scheme* in this paper. The scheme guarantees that any two nodes can directly establish a pairwise key without exposing any secret to other nodes. Even after a large number of nodes have been compromised, the pairwise keys shared by non-compromised nodes remain highly secure. Moreover, the scheme adapts to changes in network topology and incurs low computation and communication overhead. To the best of our knowledge, the RPB scheme is the only one that provides all these salient features without relying on public key cryptography. Through prototype-based evaluation, we show that the RPB scheme is highly efficient and practical for current generation of sensor nodes. In particular, to support a sensor network with up to 2^{16} nodes, establishing a pairwise key of 80 bits between any two 8-bit, 7.37-MHz MICA2 motes only requires about 0.13 second of CPU time, 0.33 KB RAM space, and 15 KB ROM space per node.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General - Security and protection; C.2.1 [Computer-Communication Networks]: Network Architecture and Design - Wireless communication; K.6.5 [Management of Computing and Information systems]: Communication Networks - Security and protection

[*]This work was partially supported by the National Science Foundation (CNS-0627354, CNS-0524156, CNS-0627382, CNS-0643906, and CNS-0519460) and Army Research Office (W911NF-05-1-0270).

General Terms

Security, Algorithm, Design

Keywords

Pairwise Key Establishment, Random Perturbation, Polynomial, Sensor Network Security

1. INTRODUCTION

In a typical wireless sensor network, node-to-node communication is the most common communication model [1,2]. For example, a node may exchange routing control information with a neighbor, send its sensor readings (or decisions) to a direct neighbor node towards a base station or towards a cluster head (or an aggregation point) that is probably multiple hops away. A node may also communicate with a mobile sink to provide sensor data or other services [3]. Because messages are transmitted in the open air, inter-node communication is subject to simple eavesdropping. Especially, if two communicating nodes are not within each other's transmission range, their messages may have to go through multiple hops, which further increases the risk of being eavesdropped or being modified. To secure their communication, it is necessary for the communicating nodes to share a secret key for encryption and authentication in the first place.

In general, pairwise key establishment (PKE) in sensor networks is challenging because of the potentially large network scale and the constrained system resources. Moreover, sensor networks are often deployed in unattended and adversarial environments. Due to these challenges, a PKE scheme must meet the following requirements:

- *Resilience to Large Number of Node Compromises* — A PKE scheme should be resilient to a large number of node compromises because sensor nodes are low-cost; hence they cannot afford tamper-resistance hardware. Recent advances in physical attack show that even memory chips with built-in tamper-resistance mechanisms are subject to various memory read-out attacks [9–12]. Thus, an adversary may capture many sensor nodes and analyze them to obtain their secret keys.
- *Guaranteed Key Establishment* — A PKE scheme should guarantee that any two nodes can establish a pairwise key whenever needed.
- *Direct Key Establishment* — A PKE scheme should allow two nodes that can communicate (directly or indirectly) with each other to establish a pairwise key

Table 1: Comparison of Major PKE Schemes With Respect to Several Desired Properties [ProbKShare: Probabilistic Key Predistribution-based PKE Schemes; ProbPShare: Probabilistic Polynomial Share Predistribution-based PKE Schemes]

Scheme / Property	RPB (ours)	SNEP ([1])	LEAP ([2])	ProbKShare ([4,5] etc.)	ProbPShare ([6,7] etc.)	Blundo ([8])
Resilience to Large Number of Node Compromises	√	√	√		√	
Guaranteed Key Establishment	√					√
Direct Key Establishment	√					√
Resilience to Dynamic Network Topology	√					√
Efficiency	√		√	√	√	

without exposing secrets to or obtaining secrets from any third parties (e.g., a central on-line server or other helper nodes). The involvement of third parties is highly undesirable because third parties may have been compromised, they may not be available, and more messages have to be exchanged among involved nodes.

- *Resilience to Dynamic Topology* — A PKE scheme should work even if one or both nodes are mobile. In some applications, a mobile sink (a mobile sensor [13] or a mobile soldier) may perform some tasks in a sensor network, which require secure communication between the mobile sink and sensor nodes.

- *Efficiency* — A PKE scheme should be efficient in computation, communication, and storage.

In summary, a practical PKE scheme for sensor networks should be efficient, resilient to attacks, and guaranteeing direct key establishment irrespective of network topology and node mobility. Although many PKE schemes [1, 2, 4–7, 14–18] have been proposed for sensor networks, most of them make tradeoffs among different requirements. Table 1 shows that none of the major PKE schemes provide all the required properties except our RPB scheme.

Recently, a number of public key-based approaches have been proposed for PKE in sensor networks. With only software implementation, the public key-based PKE approaches incur a delay ranging from several seconds to tens of seconds [19–21]. The performance can be significantly improved with special hardware support [22], but the introduction of special hardware raises the manufacturing cost, contradicting the goal of *low-cost* in deploying sensor networks. Although the performance of the public key-based approach might also be improved through the techniques such as instruction-level optimization for particular architectures, our RPB scheme is still more efficient and easier to implement since it involves only simple operations, has low cost, and can be implemented independently of the architecture.

Contributions This paper presents a random perturbation based (RPB) scheme for pairwise key establishment in sensor networks. Compared to the previous PKE schemes our RPB scheme makes the following contributions:

- First, any two nodes that can communicate with each other can always establish a pairwise key whenever needed, regardless of the network size and topology, node density, and node mobility. Moreover, nodes do not expose secrets to or obtain secrets from other nodes for PKE; hence, established keys are not exposed to others. These properties make our RPB scheme applicable to large-scale distributed sensor networks as well as mobile ad hoc networks.

- Second, the cost for an adversary to break the pairwise keys shared by non-compromised node pairs is prohibitively high, even after the adversary has compromised a large number of nodes.

- Third, both the idea of adding random noise into a key establishment process and our key construction technique are very novel. It is well known that a threshold secret sharing based system [8, 23, 24] provides the binary security property. That is, the system is unconditionally secure if the number of colluding users are no more than a threshold value, whereas the system is completely broken if the threshold value is exceeded. By introducing random perturbation, our RPB scheme blurs the threshold value in Blundo scheme [8] and provides strong security under colluding attacks by a large number ($\gg$ the threshold) of nodes. Although our scheme does not provide unconditional security as Blundo scheme does, our design has ensured that its computational security is strong enough to defeat any known attacks.

- Fourth, through analysis and prototype implementation in real sensors, we demonstrate that the RPB scheme is highly favorable for the current generation of sensor nodes because it is computationally efficient, only requires a small storage space, and has little communication overhead.

Organization The rest of the paper is organized as follows. Section 2 introduces the system model and briefly describes the key distribution scheme proposed by Blundo et al. [8], which is the basis of our RPB scheme. Section 3 and 4 provide the basic idea and the detailed description of the RPB scheme. Section 5 analyzes the security properties. Prototype implementation and evaluation are reported in Section 6. Finally, Section 7 concludes the paper.

2. PRELIMINARIES

2.1 System Model

We consider a wireless sensor network that is composed of low-power, low-cost sensor nodes, e.g., the Berkeley MICA motes [25]. These nodes have limited power supply, storage space, and computational capability. In particular, each MICA2 mote [25] has an 8-bit 7.37-MHz processor, 4 KB primary memory (SRAM), and 128KB program memory

(ROM). Due to the constrained resources, computationally expensive and energy-intensive operations are not favorable for such systems. In addition, each sensor node is not tamper-resistant. Once a sensor node is captured, the adversary can read its memory to get all information stored there. The sensor network is administrated by an *offline authority*, which is responsible for node initialization and deployment. Before deploying a node, the authority assigns the node a unique identity (ID) from a set of legitimate IDs.

2.2 A Polynomial-Based Key Predistribution Scheme

In the context of sensor networks, we briefly review the polynomial-based key predistribution scheme proposed by Blundo *et al.* [8], which is the basis of our RPB scheme. To predistribute pairwise keys, the authority randomly picks a t-degree symmetric, bivariate polynomial

$$f(x,y) = \sum_{0 \le i,j \le t} A_{i,j} x^i y^j$$

over a finite field F_q, where q is a large prime number. The authority assigns a unique id (e.g., u) to each node before deploying it into the network. The authority also computes and preloads a univariate *polynomial share* of $f(x,y)$ for the node. In particular, for the node of id u, the preloaded share is $f(u,y) = \sum_{j=0}^{t} B_{u,j} y^j$, where $B_{u,j} = \sum_{i=0}^{t} A_{i,j} u^i$. For any two nodes u and v, node u can compute the key shared with node v, i.e., $f(u,v)$, by evaluating $f(u,y)$ at $y = v$. Node v can compute $f(v,u)$ in the similar way. Since $f(x,y)$ is symmetric, $f(u,v) = f(v,u)$. So, node u and v can agree on the same key for communication. The above process is also illustrated in Fig. 1.

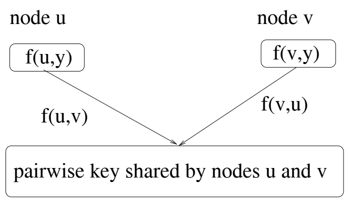


Figure 1: A polynomial-based scheme for generating pairwise keys

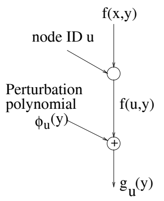


Figure 2: Generating the perturbed polynomial $g_u(y)$

The security proof in [8] ensures that this scheme is unconditionally secure and t-collusion resistant; i.e., a coalition of no more than t compromised nodes cannot know anything about the key shared by any two non-compromised nodes. However, if $(t+1)$ or more nodes are compromised, the adversary can find out the pairwise key shared by any two non-compromised nodes. Suppose nodes $u_0, u_1, \cdots, u_t$ are compromised. The adversary can construct $(t+1)$ systems of linear equations, and each system includes $(t+1)$ linear equations, where the i^{th} system of linear equations is as follows:

$$\sum_{j=0}^{t} A_{j,i} u_k^j = B_{u_k,i},\ k = 0, \cdots, t. \tag{1}$$

By solving these linear equations, the adversary can find out all the coefficients of $f(x,y)$, i.e., $A_{i,j}$ $(0 \le i,j \le t)$.

In the above scheme, the security level can be improved by increasing t. However, this is not scalable since the computational complexity and the storage overhead increase rapidly as t increases. To address the problem, Liu and Ning [7] proposed schemes that combine the above scheme with the key pool idea [4,5]. But, these schemes cannot guarantee direct key establishment. Also, these schemes allow two nodes failing in directly establishing a pairwise key to find other nodes to help set up a key, which may result in extra (sometimes high) communication overhead and another severe security breach, i.e., exposing secret keys to other nodes which could be compromised.

3. BASIC IDEA OF THE RPB SCHEME

To securely establish pairwise keys and meanwhile prevent a large number of compromised colluding nodes from breaking the pairwise key shared by any two innocent nodes, we propose a random perturbation-based (RPB) scheme. This scheme relies on polynomials to generate pairwise keys, and the polynomials are defined over a finite field denoted as F_q, where q is a prime number. Before presenting the basic idea of the RPB scheme, we first list some notations and introduce a new concept called *perturbation polynomials*.

3.1 Notations

Following is a list of notations used in presenting the basic idea of RPB:

- q, l: q is a prime number ($q > 2$), and l is the minimal integer such that $2^l > q$. Thus, every element in field F_q can be represented by l bits.
- S: a set of legitimate IDs for sensor nodes. In this paper, we let $S \subset \{0, \cdots, q-1\}$.
- r: a positive integer such that $2^r < q$.
- Φ: a set of perturbation polynomials (to be defined in Section 3.2).
- $f(x,y)$: a symmetric polynomial, in which the degree of x and y are both t (t is a system parameter).
- $g_u(y)$ ($u \in S$): a t-degree univariate polynomial that is preloaded to node (with id u) before it is deployed.

3.2 Perturbation Polynomials

In RPB, we introduce the concept of *perturbation polynomial*, which is defined as follows:

> Given a finite field F_q, a positive integer r ($2^r < q$), and a set of node IDs S ($S \subset \{0, \cdots, q-1\}$), a polynomial set Φ is a set of *perturbation polynomials regarding r and S* if any polynomial $\phi(.) \in \Phi$ has the following ***limited infection*** property:
>
> $$\forall u \in S, \phi(u) \in \{0, \cdots, 2^r - 1\}.$$

The above definition ensures that the value of a perturbation polynomial will not be grater than $2^r - 1$; i.e., it has at most r bits. This property is exploited in our design of the RPB scheme. Note that, adding a r-bit number to a l-bit number (l is the minimal integer such that $q < 2^l$), the least significant r bits of the l-bit number are directly affected, while whether its most significant $l - r$ bits is changed or not depends on if a carry being generated from the least significant r bits in the addition process. For example, adding

$(101000)_2$ by $(0101)_2$ changes its least significant $r = 4$ bits but does not change the most significant $l - r = 2$ bits; however, adding it by $(1010)_2$ changes both its least significant 4 bits but also the most significant 2 bits.

3.3 Basic Idea of The RPB Scheme

In the basic polynomial-based scheme [8], where any two nodes (with IDs u and v) are given shares ($f(u, y)$ and $f(v, y)$) of a symmetric polynomial $f(x, y)$, they can always find a match ($f(u, v)$) to be used as the shared key of size l bits. Different from this, the RPB scheme does not give each node the original share but the perturbed share, which is the sum of the original share and a perturbation polynomial with the limited infection property. The motivation for adding the perturbation with limited infection can be summarized as follows:

- First, adding perturbation polynomials makes it harder to break the symmetric polynomials. This is because the adversary cannot obtain the original shares of polynomial $f(x, y)$, and thus, as to be discussed in Section 5, it has prohibitively high complexity to break $f(x, y)$ even if it has compromised a large number of sensor nodes.
- Second, two nodes can still establish a key, though the addition of perturbation polynomials changes the values of the original match key ($f(u, v)$) at both sides. The principle behind this can be explained as follows:

  The addition of the perturbation polynomials directly affects the r least significant bits of the l-bit original match key (this is because of the way we construct ϕ to have the limited infection property) and may also affect the most significant $l - r$ bits of the original match key due to the *carry* generated in the addition process. Because the perturbation polynomials added to nodes u and v are different, this addition changes the original match key at both nodes into new values that do not match anymore. However, we can throw away the least significant r bits of the results after the addition, and thus, we only have to deal with the most significant $l - r$ bits of the results. In some cases these $l - r$ bits stay the same at both nodes u and v, so we still have a match to be used for our shared key; in other cases some of these $l - r$ bits are changed but, as to be shown later, they must belong to only two predictable cases, so we can still find a match to be used as the shared key.

To further explain the above basic idea, we now introduce the three major steps of the RPB scheme: *system initialization*, *predistribution of perturbed polynomials* and *key establishment*.

3.3.1 System Initialization

The authority arbitrarily constructs a bivariate polynomial $f(x, y)$, where the degrees of x and y are both t (t is a system parameter), and for any x and y, $f(x, y) = f(y, x)$. Then, the authority picks system parameter r, constructs a node ID set $S \subset \{0, \cdots, q-1\}$, and constructs a set of perturbation polynomials Φ regarding S and r. The procedure for constructing S and Φ are detailed in Section 4.4.

3.3.2 Predistribution of Perturbed Polynomials

Before a node (with id $u \in S$) is deployed, due to the reasons presented in Section 3.3, the authority does not preload the original polynomial share of $f(x, y)$, i.e., $f(u, y)$, to the node. Instead, as shown in Fig. 2, the authority randomly picks a polynomial $\phi_u(y)$ from Φ and preloads $g_u(y) = f(u, y) + \phi_u(y)$ to node u. Note that node u is only given the coefficients of $g_u(y)$, so it cannot find out the coefficients of either $f(u, y)$ or $\phi_u(y)$ from $g_u(y)$.

3.3.3 Pairwise Key Establishment

We now show how any two nodes (say u and v) can establish a pairwise key. When node u wants to communicate securely with node v, it initiates the key establishment process:

Step 1: Node u evaluates $g_u(y)$ at $y = v$, and represents the evaluation result in l binary bits.

Step 2: It uses the most significant $l - r$ bits of $g_u(y)$, denoted as $K_{u,v}$, as the key.

Step 3: Node u sends $h(K_{u,v})$ to node v, where $h(.)$ is a secure hash function such that any node overhearing $h(K_{u,v})$ cannot derive $K_{u,v}$. In theory, $h(.)$ can be any secure hash function. Since RC5 [26] has been implemented in sensor nodes (e.g., in TinySEC [27]) and have been adopted in the implementation of other sensor network key management schemes such as [7], we use it as the hash function in our prototype implementation.

After receiving $h(K_{u,v})$, node v goes through the following steps to construct three keys denoted as $K_{v,u}$, $K^-_{v,u}$ and $K^+_{v,u}$:

Step 1: Node v evaluates $g_v(u)$, $g_v(u) + 2^r$ and $g_v(u) - 2^r$.

Step 2: Each evaluation result is represented in l binary bits, and its most significant $l - r$ bits is computed and assigned to $K_{v,u}$, $K^+_{v,u}$ or $K^-_{v,u}$, respectively.

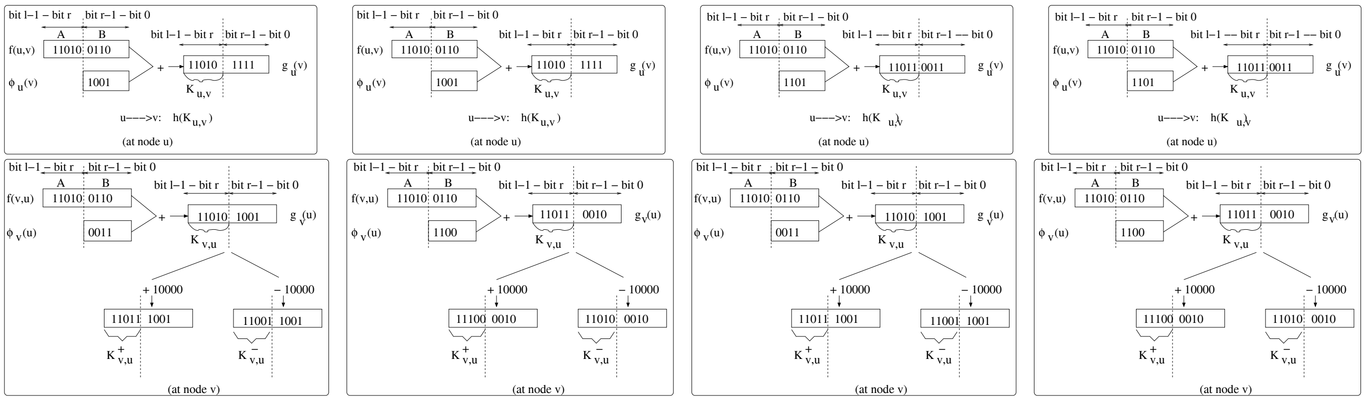
As stated in Theorem 1, one of $K_{v,u}$, $K^-_{v,u}$ and $K^+_{v,u}$ that are computed by node v must be the same as $K_{u,v}$ that is sent by node u.

THEOREM 1. *For any two nodes u and v, where $\{u, v\} \subset S$, it holds that $K_{u,v} = K_{v,u}$, $K_{u,v} = K^+_{v,u}$, or $K_{u,v} = K^-_{v,u}$.*

PROOF. (See Appendix A)
□

Examples: Some examples are shown in Fig. 3 to illustrate the property stated in Theorem 1. In these examples, the prime number q is 457 (i.e., $(111000111)_2$), l is 9, and r is 4. That is, $f(u, v)$, $f(v, u)$, $g_u(v)$ and $g_v(u)$ can be represented by $l = 9$ binary bits; $\phi_u(v)$ and $\phi_v(u)$ can be represented by $r = 4$ binary bits. For the examples in Fig. 3, $K_{u,v}$, $K_{v,u}$, $K^-_{v,u}$ and $K^+_{v,u}$ are the most significant $l - r$ bits of $g_u(v)$, $g_v(u)$, $g_v(u) - 2^r$ and $g_v(u) + 2^r$. We can see four cases regarding the matching between these keys:

- Case i: As shown in Fig. 3(a), if neither $f(u, v) + \phi_u(v)$ nor $f(v, u) + \phi_v(u)$ generates a carry from bit $r - 1$ to bit r, we have $K_{u,v} = K_{v,u}$.



(a) case i: $K_{u,v} = K_{v,u}$ (b) case ii: $K_{u,v} = K^{-}_{v,u}$ (c) case iii: $K_{u,v} = K^{+}_{v,u}$ (d) case iv: $K_{u,v} = K_{v,u}$

Figure 3: Examples of Using RPB to Generate Pairwise Keys [All the arithmetic operations are over finite field F_q and $q = (111000111)_2$.]

- Case ii: As shown in Fig. 3(b), if only $f(v, u) + \phi_v(u)$ generates a carry from bit $r-1$ to bit r, we have $K_{u,v} = K^{-}_{v,u}$.
- Case iii: As shown in Fig. 3(c), if only $f(u, v) + \phi_u(v)$ generates a carry from bit $r-1$ to bit r, we have $K_{u,v} = K^{+}_{v,u}$.
- Case iv: As shown in Fig. 3(d), if both $f(u, v) + \phi_u(v)$ and $f(v, u) + \phi_v(u)$ generates a carry from bit $r - 1$ to bit r, we have $K_{u,v} = K_{v,u}$.

Based on Theorem 1, node v can find out $K_{u,v}$ by computing $h(K_{v,u})$, $h(K^{+}_{v,u})$ and $h(K^{-}_{v,u})$, and comparing them with $h(K_{u,v})$ sent by node u as follows:

- $h(K_{u,v}) = h(K_{v,u}) \Rightarrow K_{u,v} = K_{v,u}$.
- $h(K_{u,v}) = h(K^{+}_{v,u}) \Rightarrow K_{u,v} = K^{+}_{v,u}$.
- $h(K_{u,v}) = h(K^{-}_{v,u}) \Rightarrow K_{u,v} = K^{-}_{v,u}$.

3.4 Remaining Issues

Comparing Fig. 1 to Fig. 3, we can see that using the RPB scheme reduces the size of the generated pairwise key because some bits of the polynomial evaluation results are cut off. To deal with this problem, we use multiple polynomials to generate multiple key segments, and combine them together to form a pairwise key with the desired size. This will be described in Section 4.

Another challenge in implementing the RPB scheme is to construct a set of perturbation polynomials that has the limited infection property. In Section 4, we will also investigate this issue and propose an algorithm to solve it.

4. DETAILED DESCRIPTION OF THE RPB SCHEME

In this section, we present more details of the RPB scheme that can support large-scale networks and can compute keys with a large size. In addition to the notations q, l, r, S and Φ introduced in the above, we further introduce the following notations:

- N: the desired size of a sensor network; i.e., the largest number of sensor nodes that a sensor network is expected to contain.
- L: the desired size (in the unit of bits) of each pairwise key.
- $f_i(x, y)$ $(i = 0, \cdots, m-1)$: a set of symmetric bivariate polynomials constructed by the offline authority, where the degrees of x and y are both t.
- $g_{u,i}(y)$ $(i = 0, \cdots, m-1,\ u \in S)$: a set of univariate t-degree polynomials preloaded to node with id u before it is deployed.

4.1 System Initialization

Based on the system requirements, including the desired sensor network size (N) and the desired pairwise key size (L), the following steps are performed to bootstrap the system:

(1) The authority arbitrarily constructs $m = \lceil L/(l - r) \rceil$ polynomials $f_i(x, y)$ $(i = 0, \cdots, m - 1)$ over F_q. Note that for any two nodes u and v, only $l-r$ bits of $f_i(u, v)$ can be used for pairwise key (as shown in Fig. 3). Therefore, $\lceil L/(l-r) \rceil$ of such polynomials are required for generating a pairwise key of L bits.

(2) The authority constructs a set (S) of legitimate ids and a set (Φ) of perturbation polynomials such that, Φ has the limited infection property regarding S and r. How to perform this step is an important but complicated issue. To help readers get a big picture of the RPB scheme, we defer a detailed description of the construction process to Section 4.4.

4.2 Predistribution of Perturbed Polynomials

Before a node is deployed, the offline authority assigns to it a unique id u from the id set S and m univariate polynomials $g_{u,i}(y)$ $(i = 0, \cdots, m - 1)$, where

$$g_{u,i}(y) = f_i(u, y) + \phi_{u,i}(y),\ i = 0, 1, \cdots, m - 1, \qquad (2)$$

and each $\phi_{u,i}(y)$ is a perturbation polynomial randomly picked from the perturbation polynomial set Φ. Note that node u cannot find out $f_i(u, y)$ or $\phi_{u,i}(y)$ from $g_{u,i}(y)$. Also, all $\phi_{u,i}(y)$ $(i = 0, \cdots, m - 1,\ u \in S)$ are picked from Φ independently.

4.3 Pairwise Key Generation

The key establishment process between any two nodes u and v is similar to the one described in Section 3, except that u and v should establish the key based on the shares of multiple (m) polynomials. We describe this process in the following steps:

(1) For each $g_{u,i}(y)$ $(i = 0, \cdots, m-1)$, node u computes a key segment $s_{u,i}$, which is the most significant $(l - r)$ bits of $g_{u,i}(v)$. A concatenation of these key segments, denoted as $K_{u,v} = (s_{u,0} \mid s_{u,1} \mid \cdots \mid s_{u,m-1})$, is used as the pairwise key shared with node v.

(2) Having computed $K_{u,v}$, similar to the process described in Section 3, node u should send a hash value of $K_{u,v}$ to node v. This can be constructed as the exclusive-OR of the hash values of $s_{u,i}$ $(i = 0, \cdots, m-1)$; i.e., $H(K_{u,v}) = h(s_{u,0} \mid R_0) \bigoplus h(s_{u,1} \mid R_1) \bigoplus \cdots \bigoplus h(s_{u,m-1} \mid R_{m-1})$. Then, $H(K_{u,v})$ is sent to v. In the computation of $H(K_{u,v})$, $R_0, \cdots, R_{m-1}$ are large random numbers shared by all sensor nodes. They are used such that, for any two different i and j in $\{0, \cdots, m-1\}$, even if $s_{u,i} = s_{u,j}$, $h(s_{u,i} \mid R_i) \neq h(s_{u,j} \mid R_j)$ since $R_i \neq R_j$. Note that using $\bigoplus$ to construct the $H(K_{u,v})$ is a special case of the XOR-MAC scheme [28], which has been proved to be secure.

(3) On receiving $H(K_{u,v})$ from node u, node v computes three segments (denoted as $s_{v,i}$, $s^+_{v,i}$ and $s^-_{v,i}$) for each $g_{v,i}(y)$ $(i = 0, \cdots, m-1)$. Here, $s_{v,i}$ is a bit-string extracted from $g_{v,i}(u)$ in the same way as node u extracts $s_{u,i}$ from $g_{u,i}(v)$ (detailed in Step (1)). $s^+_{v,i}$ and $s^-_{v,i}$ are also extracted in the same way from $g_{v,i}(u) + 2^r$ and $g_{v,i}(u) - 2^r$, respectively.

(4) Similar to the cases explored in Section 3, $s_{u,i}$ $(i = 0, \cdots, m-1)$ could be equal to one of $s_{v,i}$, $s^+_{v,i}$ and $s^-_{v,i}$ with the same probability. Therefore, $K_{u,v}$ must be equal to one of the following 3^m strings:

$$s'_{v,0} \mid s'_{v,1} \mid \cdots \mid s'_{v,m-1},$$

where each $s'_{v,i} \in \{s_{v,i}, s^+_{v,i}, s^-_{v,i}\}$. Specifically, let us suppose $m = 2$, all the $3^2 = 9$ strings are:

$$\begin{array}{ll}
K_{v,u,0} = s_{v,0} \mid s_{v,1}, & K_{v,u,1} = s_{v,0} \mid s^+_{v,1}, \\
K_{v,u,2} = s_{v,0} \mid s^-_{v,1}, & K_{v,u,3} = s^+_{v,0} \mid s_{v,1}, \\
K_{v,u,4} = s^+_{v,0} \mid s^+_{v,1}, & K_{v,u,5} = s^+_{v,0} \mid s^-_{v,1}, \\
K_{v,u,6} = s^-_{v,0} \mid s_{v,1}, & K_{v,u,7} = s^-_{v,0} \mid s^+_{v,1}, \\
K_{v,u,8} = s^-_{v,0} \mid s^-_{v,1}. &
\end{array}$$

To find out $K_{u,v}$, node v computes $H(K_{v,u,i})$ for each $i \in \{0, \cdots, 8\}$. For example, $H(K_{v,u,4}) = h(s^+_{v,0} \mid R_0) \bigoplus h(s^+_{v,1} \mid R_1)$. A string $K_{v,u,i}$ is equal to $K_{u,v}$ *iff* $H(K_{v,u,i})$ is equal to the received $H(K_{u,v})$.

4.4 Constructing S and Φ

How to construct the id set S and perturbation polynomial set Φ is vital for the RPB scheme, and the construction process should satisfy the following *Requirements*:

(a) For the RPB scheme to work, *limited infection* property should be satisfied for S and Φ.

(b) The size of S should be large in order to support a large-scale sensor network.

(c) Φ should include more than one randomly constructed perturbation polynomial. As to be shown in Theorem 2 of Section 5, this is important for the security of the RPB scheme because the time complexity to break the system is $\Omega(m* \mid \Phi \mid^{t+1})$.

(d) To efficiently use the RPB scheme, the computation complexity for constructing S and Φ should be as low as possible.

Our approach for constructing S and Φ is based on the following idea. We first initialize $S_1 = \{0, \cdots, q-1\}$ as a set of legitimate ids. A t-degree univariate polynomial, denoted as $\hat{\phi}_1(y)$, is randomly constructed. The polynomial maps all the IDs in set S_1 into multiple groups based on the most significant $l - r$ bits of the mapped value, and the IDs mapped to the largest group form a new ID set denoted as S_2. Then, $\hat{\phi}_1(y)$ is transformed (the detail for the transform will be described later) to another polynomial $\phi_1(y)$ such that $\phi_1(y)$ is a perturbation polynomial regarding S_2 and r. This process continues as follows: another polynomial $\hat{\phi}_2(y)$ is randomly constructed; it maps all the IDs in S_2 into multiple groups; the IDs mapped to the largest group becomes another new ID set S_3; $\hat{\phi}_2(y)$ is transformed to polynomial $\phi_2(y)$ which is a perturbation polynomial regarding S_3 and r. As the above proceeds, more perturbation polynomials are found and the set of legitimate IDs shrinks. Suppose the process stops after the n^{th} step. Then, S_n becomes the legitimate ID set and all the perturbation polynomials generated so far form the set of polynomials regarding S_n and r. A formal description of the algorithm is presented as follows:

(0) Initializations: $i = 1$, $S = S_i = \{0, \cdots, q-1\}$, $\Phi = \emptyset$.

(1) A t-degree polynomial $\hat{h}_i(y)$ is randomly constructed over F_q.

(2) Based on $\hat{h}_i(y)$, S_i is divided into w subsets denoted as $S_{i,0}, S_{i,1}, \cdots, S_{i,(w-1)}$, where $w = 2^{l-r}$, each $S_{i,j}$ $(j = 0, \cdots, w-1)$ is defined as

$$\{y \mid \hat{h}_i(y) = j * 2^r + c, \text{where } c \in \{0, \cdots, 2^r - 1\}\}.$$

As an example, suppose $l = 9$, $q = 457$, $r = 4$, $\{0, 1\} \subset S_i$, $\hat{h}_i(0) = 131 = 8*2^4 + 4$, and $\hat{h}_i(1) = 50 = 3*2^4 + 2$. According to the rule for set division, $0 \in S_{i,8}$, $1 \in S_{i,3}$.

(3) Let $S_{i,k}$ be the largest subset of S_i. If $\mid S_{i,k} \mid < N$, the algorithm terminates. Otherwise, Step (4) is executed. Note that $\mid S_{i,k} \mid$ is checked in this step to guarantee that the generated ID set contains at least N (the desired network size) ids.

(4) Let $h_i(y) = \hat{h}_i(y) - k * 2^r$. Then, for any $u \in S_{i,k}$, we have $h_i(y) \in \{0, \cdots, 2^r - 1\}$ because $\hat{h}_i(u) \in \{k * 2^r + c \text{and } c \in \{0, \cdots, 2^r - 1\}\}$ according to the rule stated in step (2). Therefore, polynomial $h_i(y)$ and ID set $S_{i,k}$ satisfy the limited infection property; i.e., $h_i(y)$ is a perturbation polynomial regarding $S_{i,k}$ and r. So, $h_i(y)$ is added to the perturbation polynomial set Φ. Note that the polynomials previously added to Φ are perturbation polynomials regarding r and S_i, which is a superset of $S_{i,k}$. So, they must also be perturbation polynomials regarding r and $S_{i,k}$.

(5) Let $S = S_{i+1} = S_{i,k}$, $i = i+1$, and repeat steps (1)-(5).

It is easy to see that the constructed S and Φ satisfy requirements (a)-(c). Also, at most $q-1 < 2^l$ polynomial evaluations are needed for each execution of Step (2), and Step (1) and Step (3) have lower computation complexity than Step (2). So the complexity for finding out a perturbation polynomial is $O(2^l)$ evaluations of t-degree polynomials. Note that the algorithm can be run in advance by the offline authority, which has much more computation power. Table II shows some examples for setting parameters l, r and m to construct at least two perturbation polynomials, given the desired network size. For example, to support a network size of $N = 2^{16}$, the parameters can be set as follows: $q = 2^{40} - 87$, $l = 40$, and $r = 28$.

5. SECURITY ANALYSIS

After an adversary has compromised n_c nodes, denoted as $u_1, u_2, \cdots, u_{n_c}$, and captured the polynomial shares preloaded to these nodes, the adversary can attack the system based on the captured shares.

5.1 Breaking $f_i(x,y)$ $(i = 0, \cdots, m-1)$

In the polynomial-based scheme proposed by Blundo et al. [8], the polynomial used for generating all pairwise keys can be broken after the number of compromised nodes (n_c) exceeds the degree (t) of the polynomial. Since the RPB scheme is also polynomial-based, it is important to study whether the scheme has similar limitations. In RPB, pairwise keys are constructed based on multiple polynomials $f_i(x,y)$ $(i = 0, \cdots, m-1)$. An adversary must compromise all these polynomials in order to break down the system. Each $f_i(x,y)$ is a t-degree bivariate and symmetric polynomial. Recall that the adversary can find out at most one perturbed share of $f_i(x,y)$, i.e., $g_{u_k,i}(y)$, from each compromised node u_k. Therefore, the adversary cannot break $f_i(x,y)$ if $n_c \le t$. In the following, we only consider the case that $n_c \ge t+1$.

LEMMA 1. *The probability for the adversary to break any $f(x,y) \in \{f_i(x,y) \mid i = 0, \cdots, m-1\}$ in one attempt is $\frac{1}{|\Phi|^{t+1}}$.*

PROOF. (sketch) Because $g_{u_k}(y) = f(u_k, y) + \phi_{u_k}(y)$, we obtain a system of linear equations as follows:

$$\sum_{i=0}^{t} (u_k)^i \cdot A_{i,j} + B_{k,j} = D_{k,j}, j = 0, \cdots, t \text{ and } k = 1, \cdots, n_c. \tag{3}$$

where,

- $f(x,y) = \sum_{0 \le i,j \le t} A_{i,j} x^i y^j$. Each $A_{i,j}$ is unknown and $A_{i,j} = A_{j,i}$. So the number of unknowns in $f(x,y)$ is $(t+1) * t/2$.
- $\phi_{u_k}(y) = \sum_{j=0}^{t} B_{k,j} y^j$, and each $B_{k,j}$ is unknown.
- $g_{u_k}(y) = \sum_{j=0}^{t} D_{k,j} y^j$, and each $D_{k,j}$ is known.

In linear system (3), the total number of linear equations (i.e., $n_c * (t+1)$) is less than the total number of unknowns (i.e., $n_c * (t+1) + (t+1) * t/2$). So, a unique solution for $A_{i,j}$ $(0 \le i,j \le t)$ cannot be found if the number of unknowns is not reduced.

Due to the arbitrariness in constructing polynomial $f(x,y)$, any two A_{i_0,j_0} and A_{i_1,j_1} ($i_0 \ne j_1$ or $j_0 \ne i_1$) are independent. So, the number of distinct $A_{i,j}$ is $(t+1) * t/2$ and cannot be reduced. However, if the adversary knows that the same perturbation polynomial is preloaded to a group of nodes (e.g., $u_0, \cdots, u_w$, without loss of generality), i.e., $\phi_{u_0}(y) \equiv \cdots \equiv \phi_{u_w}(y)$, then $B_{0,j} = \cdots = B_{w,j}$ for $j = 0, \cdots, t$. In this case the number of unknowns in linear system (3) is reduced by $w * (t+1)$; i.e., the number of distinct $\langle B_{i,0}, \cdots, B_{i,t} \rangle$ $(i = 1, \cdots, n_c)$ is reduced by w. In the following, we denote $\langle B_{i,0}, \cdots, B_{i,t} \rangle$ as $\hat{B}_i$. Furthermore, only when the number of distinct $\hat{B}_i$ is reduced by $(t+1)$, can the unique solution to $A_{i,j}$ $(0 \le i,j \le t)$ in system (3) be found. This can be achieved by identifying one or more groups of nodes such that the nodes in the same group are preloaded with the same perturbation polynomial.

Next, we study the probability for the adversary to correctly group nodes based on the perturbation polynomials preloaded to them. Recall that in the RPB scheme: (a) The polynomial share preloaded to each node is perturbed with a perturbation polynomial randomly picked from Φ. That is, each node has the same probability to have its share perturbed by any perturbation polynomials. (b) For any two nodes, they cannot find out whether they are preloaded with the same perturbation. Considering two arbitrary nodes u_0 and u_1, $g_{u_0}(u_1) - g_{u_1}(u_0) = (f(u_0,u_1) - f(u_1,u_0)) + (\phi_{u_0}(u_1) - \phi_{u_1}(u_0)) = \phi_{u_0}(u_1) - \phi_{u_1}(u_0)$. Because $\{\phi_{u_0}(u_1), \phi_{u_1}(u_0)\} \subset \{0, \cdots, 2^r - 1\}$, $\phi_{u_0}(u_1) - \phi_{u_1}(u_0)$ can be any element in $\{0, \cdots, 2^r - 1\} \bigcup \{q - (2^r - 1), q - 1\}$, no matter $\phi_{u_0}(y)$ and $\phi_{u_1}(y)$ are the same or not. Note that $\phi_{u_0}(y) \equiv \phi_{u_1}(y)$ does not imply that $\phi_{u_0}(u_1) = \phi_{u_1}(u_0)$. Due to the above reasons, the adversary has to guess (without any other knowledge) whether a group of nodes are preloaded with the same perturbation polynomial.

Let us suppose the adversary guesses that nodes $u_0, \cdots, u_w$ are preloaded with the same perturbation polynomial. Because the number of perturbation polynomials is $|\ \Phi\ |$, the content of $\langle \phi_{u_0}(y), \cdots, \phi_{u_w}(y) \rangle$ has $|\ \Phi\ |^{w+1}$ possibilities, among which the number of cases that $\phi_{u_0}(y) \equiv \cdots \equiv \phi_{u_w}(y)$ is $|\ \Phi\ |$. So, the probability that this grouping is correct is $\frac{1}{|\Phi|^w}$.To reduce the number of distinct $\hat{B}_i$ $(i = 1, \cdots, n_c)$, multiple groups may need to be identified. Assume that N_g groups are identified, each group i has S_i nodes. Note that by putting S_i nodes into a group, the number of the distinct $\hat{B}_i$ is reduced by $S_i - 1$. To break $f(x,y)$, it must hold that $\sum_{i=1}^{N_g}(S_i - 1) \ge (t+1)$, and hence, the probability of correctly identifying these groups (i.e., $\frac{1}{\prod_{i=1}^{N_g} |\Phi|^{S_i - 1}} = (\frac{1}{|\Phi|})^{\sum_{i=1}^{N_g}(S_i-1)} \le \frac{1}{|\Phi|^{t+1}}$) is also the probability for break $f(x,y)$ in one attempt.

□

After an unsuccessful attempt, the adversary can keep on attacking until the polynomial is broken. The expected number of such attempts is $\Omega(|\ \Phi\ |^{t+1})$. Because all these $f_0(x,y), \cdots, f_{m-1}(x,y)$ are independently constructed, we have

THEOREM 2. *The computational complexity for breaking $\{f_i(x,y) \mid i = 0, \cdots, m-1\}$ is $\Omega(m * |\ \Phi\ |^{t+1})$.*

Table 2 shows some numeric results of security analysis. For example, let us assume the desired network size (N) is

Table 2: **Security Level and Supportable Network Size with Various Parameters [Desired key size (L) is 80 bits; BC: Breaking Complexity; N: Supportable Network Size]**

q	l	r	m	t	$\mid\Phi\mid$	BC	N
$2^{32}-5$	32	22	8	≥ 76	2	$> 2^{80}$	2^{12}
$2^{36}-5$	36	24	7	≥ 77	2	$> 2^{80}$	2^{12}
$2^{40}-87$	40	26	6	≥ 77	2	$> 2^{80}$	2^{12}
$2^{40}-87$	40	28	7	≥ 77	2	$> 2^{80}$	2^{16}

2^{12} and the desired key size is $L = 80$ bits. Suppose the offline authority sets $q = 2^{32}-5$, $l = 32$, $t = 76$, and $r = 22$. According to the algorithm for constructing node ID set and perturbation polynomials, $\mid \Phi \mid= 2$. Also, $m = \lceil \frac{L}{l-r} \rceil = 8$. So, the complexity for breaking $f_i(x, y)$ $(i = 0, \cdots, m-1)$ is no lower than $8 * 2^{77} = 2^{80}$.

5.2 Compromising A Partial Set of Pairwise Keys

We have shown that, if the system parameters are chosen appropriately, an adversary has prohibitively high complexity to break $f_i(x, y)$ $(i = 0, \cdots, m-1)$ to compromise all the pairwise keys. However, the adversary may attempt to compromise part of the pairwise keys. For example, it may try to break the polynomial shares associated with a particular non-compromised node v, i.e., $f_i(v, y)$ $(i = 0, \cdots, m-1)$.

First, we analyze the complexity to break a certain $f(v, y) \in \{f_i(v, y) \mid i = 0, \cdots, m-1\}$. From each compromised node u_k $(k = 1, \cdots, n_c)$, polynomial share $g_{u_k}(y)$ is captured, and thus $g_{u_k}(v)$ is known by the adversary. Also since $g_{u_k}(v) = f(u_k, v) + \phi_{u_k}(v)$, it can construct the following system of linear equations:

$$\sum_{i=0}^{t} u_k^i A_i + B_k = D_k, k = 1, \cdots, n_c, \tag{4}$$

where,

- $f(v, y) = \sum_{0\leq i\leq t} A_i x^i$, and each A_i is unknown.
- $B_k = \phi_{u_k}(v)$, and each B_k is unknown.
- $D_k = g_{u_k}(v)$, and each D_k is known.

Here, the number of equations is n_c and the number of unknowns is $(t + 1) + n_c$. Similar to the proof of Lemma 1, $f(v, y)$ can be broken only if the number of unknowns can be reduced by identifying the nodes that are preloaded with the same perturbation polynomials. Also, the probability for successfully grouping the nodes (based on the preloaded perturbation polynomials) is no higher than $\frac{1}{|\Phi|^{t+1}}$. Therefore, the computational complexity for breaking $f(v, y)$ is $\Omega(\mid \Phi \mid^{t+1})$, and the complexity to break $f_i(v, y)$, $i = 0, \cdots, m-1$, is $\Omega(m * \mid \Phi \mid^{t+1})$.

5.3 Other Attacks

The simplest attack to break the pairwise key shared by two non-compromised nodes is to directly guess the key. Since each bit of the key can be 1 or 0 with the same probability, the probability for correctly guessing the key in one attempt is $\frac{1}{2^L}$, and the computational complexity for finding out the key is $\Omega(2^L)$. Therefore, as long as the size of a key is large enough (e.g., 80 bits), the time complexity is prohibitively high.

Recall that in the course of key establishment, when the sender or the receiver finds a key fragment $K_{u,v}$ be 0, it changes it to the most significant $l-r$ bits of $q-1$. This may be utilized by the adversary, who always guess each key fragment to be the most significant $l-r$ bits of $q-1$. However, this does not significantly improve effectiveness of the attack due to the following reasons: In practice, $q > 2^l - 2^r$ (for example, in our experiments $q = 2^{32}-5, r = 24$ when $l = 32$, and $q = 2^{40}-87, r = 26$ when $l = 40$), and hence the most significant $l-r$ bits of $q-1$ is equal to $2^{l-r}-1$. Thus, $K_{u,v}$ can be any integer between 0 and $2^{l-r}-1$, and it is uniformly distributed in $\{0, \cdots, 2^{l-r}-1\}$ due to the arbitrariness in constructing polynomials $f_i(x, y)$ $(i = 0, \cdots, m-1)$ and the perturbation polynomials. If the adversary guesses each key fragment $K_{u,v}$ to be the most significant $l-r$ bits of $q-1$, the probability of successfully guessing the key fragment is $\frac{1}{2^{l-r}} * 2$. Therefore, the complexity to successfully guess the whole key is $\Omega(2^{(l-r-1)*m})$, which is still on the same order as $\Omega(2^L)$ because $2^L = 2^{(l-r)*m}$ and m is typically small.

By eavesdropping, an adversary may find out know the set of legitimate node IDs (S). Then, it may attempt to find out all possible perturbation polynomials. This attack, however, does not work because in our RPB scheme because the authority constructs the perturbation polynomial set in a random way. To find out the perturbation polynomials, the adversary has to check each n-degree $(n \leq t)$ polynomial, and evaluate it for each legitimate ID. The required computational complexity is equivalent to evaluating $2^{(t+1)\times l}$ polynomials. As long as t and l are appropriately selected, e.g., $t = 80$ and $l = 40$, the number is as high as 2^{360}, which makes the attack infeasible.

The adversary may attempt to disrupt the PKE process by randomly generating a hash value other than the one sent by the sender. If the hash value is the same as any of the valid $3^m - 1$ ones other than the one specified by the sender, the receiver will generate a different pairwise key. The probability of success is $(3^m - 1)/2^L$, which is very small because m is small, for example, 6 or 7.

6. IMPLEMENTATION AND EVALUATION

We have implemented a prototype of the RPB scheme on the 8-bit, 7.37-MHz MICA2 mote [25] running TinyOS [29]. The implementation uses the RC5 function of TinySEC [27] as the secure hash function. The system parameters q, l, r, t and m can be tuned to achieve the desired size of pairwise keys (L), the desired size of networks (N) and the desired level of security. In the following, we presents the experiments that we have conducted to evaluate the performance of RPB.

6.1 Experiment Setup

The parameters we use for evaluating the RPB scheme are shown in Table 3. In these settings, a network with $N = 2^{12}$

Table 3: Parameters of RPB in the Experiments

q	l	r	t	m
$2^{32}-5$	32	22	80	8
$2^{36}-5$	36	24	80	7
$2^{40}-87$	40	28 or 26	80	7 or 6

or 2^{16} nodes can be supported, the size of generated pairwise key is 84 bits, and the computational complexity to break the secret polynomials for key generation in this system is at least $2^t = 2^{80}$ according to the analysis in Section 5.

The RPB scheme has very low communication overhead: only a hash value of the pairwise key needs to be sent between two nodes. This value can be piggybacked in the first data message they exchange. Therefore, in the experiments we only study the computational overhead and the storage overhead of the sensor nodes. Note that the overhead of the offline key server is not considered since the server can be much more powerful than sensor nodes.

Two metrics are used in our experiments: (a) *The computational overhead per node* — the total number of CPU cycles and the CPU running time that are required to find a shared key divided by the number of communicating nodes. In particular, the reported computational overhead for RPB is the overhead of the receiver side because the sender has lower computational overhead than the receiver. (b) *The storage overhead* — the size of the program and data in ROM and RAM.

We use the tools provided by Shnayder *et al.* [30] for counting CPU cycles. To focus on the effectiveness of the key establishment schemes with regard to the two metrics, we only use two communicating nodes in our experiments.

6.2 Experiment Results

We now present the experiment results. Note that all the data presented in the figures or tables are the averaged results over 100 independent runs.

6.2.1 Computational Overhead

Table 4: Computational Overhead of RPB

Scheme	Time (in seconds)	Cycle Count
RPB	0.13	$9.59 * 10^5$

Table 4 shows the computational overhead of RPB and other two schemes in terms of CPU cycles and CPU running time required for establishing a pairwise key. Here, the parameters for RPB are as follows: $q = 2^{40} - 87$, $l = 40$, $r = 28$, $t = 80$ and $m = 7$. Therefore, pairwise keys of size $(l - r) * m = 84$ bits can be computed.

6.2.2 Storage Overhead

Table 5: Storage Overhead of RPB (in Byte)

	Full program (Comm. + RPB)	Comm. module	RPB
ROM	22,302 (code) +2,835 (data)	10, 130	12,170 (code) +2,835 (data)
RAM	714	389	325

For running key establishment schemes, each sensor node needs memory space for holding program code and data such as the coefficients of polynomials in RPB. The program and data are initially uploaded into the EPROM, and they will then be loaded into its RAM for computing pairwise keys. We develop a standalone program for testing RPB. In this program, one mote computes a pairwise key and sends it to another mote. The receiver computes candidate pairwise keys and finds out the one matching the key sent by the sender. We measure the ROM and RAM consumption of RPB, and the results are shown in Table 5. Considering the sizes of RAM and ROM in MICA2 are $4KB$ and $128KB$, the space requirements of about $0.33KB$ RAM and about $15KB$ ROM are affordable.

7. CONCLUSIONS

In this paper, we proposed a novel *random perturbation-based scheme*, which utilizes random perturbation polynomials to guarantee that any two nodes can directly compute and agree on a pairwise key; at the same time, any number of compromised colluding nodes have negligible probability to break the pairwise key shared by a pair of non-compromised nodes. Through analysis and prototype implementation, we showed that the scheme is highly secure and computationally efficient. Furthermore, it has pretty low storage requirement, and can be implemented in the current generation of sensor nodes.

8. REFERENCES

[1] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "Spins: security protocols for sensor netowrks," in *Proceedings of ACM Mobile Computing and Networking (Mobicom'01)*, 2001, pp. 189–199.

[2] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, 2003, pp. 62–72.

[3] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *ACM International Conference on Mobile Computing and Networking (MOBICOM'02)*, pp. 148–159, September 2002.

[4] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *IEEE Symposium on Research in Security and Privacy*, 2003.

[5] L. Eschenauer and V. Gligor, "A Key-management Scheme for Distributed Sensor Networks," *The 9th ACM Conference on Computer and Communications Security*, pp. 41–47, November 2002.

[6] W. Du, J. Deng, Y. Han, and P. Varshney, "A Pairwise Key Pre-distribution Schemes for Wireless Sensor Networks," *The 10th ACM Conference on Computer and Communications Security*, 2003.

[7] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *The 10th ACM Conference on Computer and Communications Security*, 2003.

[8] 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*, vol. 740, pp. 471–486, 1993.

[9] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *Proceedings of Security Protocols, LNCS 1361*, 1998.

[10] H. Handschuh, P. Pailer, and J. Stren, "Probing attacks on tamper resistant devices," in *Proceedings of Cryptographic Hardware and Embedded Systems, CHES'99*, 1999.

[11] S. Skorobogatov, "Low temperature data remanence in static ram," in *University of Cambridge, Computer Laboratory, Technical Report UCAM-CL-TR-536*, June 2002.

[12] D. Samyde, S. Skorobogatov, R. Anderson, and J. Quisquater, "On a new way to read data from memory," in *Proceedings of First International IEEE Security in Storage Workshop*, December 2002.

[13] "Cotsbots: The mobile mote-based robots," http://www-bsac.eecs.berkeley.edu/projects/cotsbots/.

[14] R. Pietro, L. Mancini, and A. Mei, "Random key assignment for secure wireless sensor networks," *Proceeding of ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[15] W. Du, J. Deng, Y. Han, and S. Chen, "A key management scheme for wireless sensor networks using deployment knowledge," *IEEE INFOCOM'04*, March 2004.

[16] D. Liu and P. Ning, "Location-based pairwise key establishment for static sensor networks," *Proceeding of ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[17] Haowen Chan and Adrian Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Proceedings of IEEE Infocom*, Mar. 2005.

[18] Arno Wacker, Mirko Knoll, Timo Heiber, and Kurt Rothermel, "A new approach for establishing pairwise keys for securing wireless sensor networks," in *Proc. of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, November 2005, pp. 27–38.

[19] D. Malan, M. Welsh, and M. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," *First IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, October 2004.

[20] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: Securing sensor networks with public key technology," *ACM SASN'04*, October 2004.

[21] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," *Third IEEE International Conference on Pervasive Computing and Communication (PerCom 2005)*, March 2005.

[22] G. Gaubatz, J. Kaps, E. Ozturk and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 146–150, 2005.

[23] A. Shamir, "How to share a secret," *Communications of the ACM*, 1979.

[24] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach," *IEEE Infocom 2005*, March 2005.

[25] "Crossbow technology inc," http://www.xbow.com 2004.

[26] R. Rivest, "The rc5 encryption algorithm," in *Proceedings of the 1st International Workshop on Fast Software Encryption*, 1994, pp. 86–96.

[27] C. Karlof, N. Sastry, U. Shankar, and D. Wagner, "Tinysec: Tinyos link layer security proposal, version 1.0," 2002.

[28] M. Bellare, R. Guerin, and P. Rogaway, "XOR MACs: New Methods for Message Authentication Using Finite Pseudo-random Functions.," *Proc. of Crypto*, 1995.

[29] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104.

[30] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, November 2004, pp. 188–200.

Appendix A: Proof of Theorem 1.

Let $f(u,v) = f(v,u) = A$, $\phi_u(v) = B_0$, $\phi_v(u) = B_1$, and C be the most significant $l-r$ bits of A. Regarding the range of A, there are three cases as follows:

Case 1: $A \in \{2^r, \cdots, (q-1)-(2^r-1)\}$.

Since $B_0 \in \{0, \cdots, 2^r-1\}$ and $B_1 \in \{0, \cdots, 2^r-1\}$, $g_u(v) = A + B_0 \in \{2^r, \cdots, q-1\}$ and $g_v(u) = A + B_1 \in \{2^r, \cdots, q-1\}$. Note that the additions in this proof are not modular additions unless particularly mentioned. Regarding whether each addition produces a carry from bit $r-1$ to bit r, there are four sub-cases as follows:

- **Case 1.1:** neither $A+B_0$ or $A+B_1$ produces a carry from bit $r-1$ to bit r. B_0 and B_1 do not affect the most significant $l-r$ bits of $A+B_0$ or $A+B_1$. Therefore, the most significant $l-r$ bits of these two are determined solely by A and hence are the same. That is, $K_{u,v} = C = K_{v,u}$.
- **Case 1.2:** only $A+B_0$ produces a carry from bit $r-1$ to bit r. In this case, $K_{u,v} = C+1$ in node u. As for node v, $K_{v,u} = C$ since there is no carry from bit $r-1$ to r in $A+B_1$. Therefore, $K^+_{v,u} = C+1 = K_{u,v}$.
- **Case 1.3:** only $A+B_1$ produces a carry from bit $r-1$ to bit r. In this case, $K_{u,v} = C$ since $A+B_0$ does not produce a carry from bit $r-1$ to r; in node v, $K_{v,u} = C+1$. Therefore, $K^-_{v,u} = C = K_{u,v}$.
- **Case 1.4:** both of them produce a carry from bit $r-1$ to bit r. In this case, $K_{u,v} = C+1$ at node u and $K_{v,u} = C+1$ at node v. So, $K_{u,v} = C+1 = K_{v,u}$.

Case 2: $A \in \{q-(2^r-1), q-1\}$.

Regarding whether the addition results, i.e., $A+B_0$ and $A+B_1$, are greater than $q-1$, there are four cases:

- **Case 2.1:** $A+B_0 \le q-1$ and $A+B_1 \le q-1$.
  This is the same as Case 1. Therefore, a match can be found.
- **Case 2.2:** $A+B_0 \ge q$ and $A+B_1 \le q-1$. At node u, since $A+B_0 \in \{q, \cdots, q-1+(2^r-1)\}$, $g_u(v) = (A+B_0 \bmod q) \in \{0, \cdots, 2^r-2\}$. So, the most significant $l-r$ bits of $g_u(v)$ is 0. At node v, $A+B_1 \in \{q-(2^r-1), \cdots, q-1\}$. Thus, $A+B_1+2^r \in \{q+1, \cdots, q+2^r-1\}$. Therefore, $(g_v(u)+2^r \bmod q) = (A+B_1+2^r \bmod q) \in \{1, \cdots, 2^r-1\}$. That is, the most significant $l-r$ bits of $(g_v(u)+2^r \bmod q)$ is also 0. So, $K^+_{v,u} = K_{u,v}$.
- **Case 2.3:** $A+B_0 \le q-1$ and $A+B_1 \ge q$. In this case, $B_0 < B_1$. Also due to $\{B_0, B_1\} \subset \{0, \cdots, 2^r-1\}$, it holds that $B_1 - 2^r < B_0$. Thus, $A+B_1-2^r < A+B_0 \le q-1 < A+B_1$. At node u, $g_u(v) = A+B_0 \in \{q-(2^r-1), \cdots, q-1\}$. So, $K_{u,v}$ is the most significant $l-r$ bits of $A+B_0$, and $K_{u,v}$ must be the same as either the most significant $l-r$ bits of $A+B_1-2^r$ or those of $q-1$. At node v, since $A+B_1 \in \{q, \cdots, q-1+(2^r-1)\}$, it holds that $g_v(u) = (A+B_1 \bmod q) \in \{0, \cdots, 2^r-2\}$. So, the most significant $l-r$ bits of $g_v(u)$ is 0. According to the algorithm, $K_{v,u}$ is set to be the most significant $l-r$ bits of $q-1$, $K^-_{v,u}$ be the most significant $l-r$ bits of $g_u(v) - 2^r \bmod q = A+B_1-2$. Therefore, $K_{u,v} = K_{v,u}$ or $K_{u,v} = K^-_{v,u}$.
- **Case 2.4:** $A+B_0 \ge q$ and $A+B_1 \ge q$. $g_u(v) = (A+B_0 \bmod q) \in \{0, \cdots, 2^r-1\}$ and $g_v(u) = (A+B_1 \bmod q) \in \{0, \cdots, 2^r-1\}$. So, the most significant $l-r$ bits of both $g_u(v)$ and $g_v(u)$ are 0, and thus, according to the algorithm, $K_{u,v} = K_{v,u}$.

Case 3: $A \in \{0, \cdots, 2^r-1\}$.

$g_u(v) = A+B_0 \in \{0, \cdots, 2^{r+1}-2\}$ and $g_v(u) = A+B_1 \in \{0, \cdots, 2^{r+1}-2\}$. That is, the most significant $l-r$ bits of $g_u(v)$ and $g_v(u)$ can only be either 0 or 1. Therefore, $K_{u,v}$ can be either 1 or the most significant $l-r$ bits of $q-1$. As for node v, there are two sub-cases:

- **Case 3.1:** The most significant $l-r$ bits of $g_v(u)$ is 0. Then, $K_{v,u}$ is equal to the most significant $l-r$ bits of $q-1$ and $K^+_{v,u} = 1$. So, it holds that $K_{v,u} = K_{u,v}$ or $K^+_{v,u} = K_{u,v}$.
- **Case 3.2:** The most significant $l-r$ bits of $g_v(u)$ is 1. Then, $K_{v,u} = 1$. On the other hand, the $l-r$ bits of $g_v(u) - 2^r \bmod q$ must be 0, and thus $K^-_{v,u}$ is the most significant $l-r$ bits of $q-1$. Therefore, it holds that $K_{v,u} = K_{u,v}$ or $K^-_{v,u} = K_{u,v}$.