A Polynomial Based Key Establishment Scheme for Heterogeneous Sensor Networks

Iuon-Chang Lin
Department of Management Information Systems
National Chung Hsing University
Taichung, Taiwan, R.O.C.
iclin@nchu.edu.tw

Shu-Yan Huang
Department of Management Information Systems
National Chung Hsing University
Taichung, Taiwan, R.O.C.
g9629001@mail.nchu.edu.tw

Abstract—Recently, many researches have focused on a multi-tiered architecture for sensor networks. The advantages of such an architecture are better scalability, energy efficiency and security. Unfortunately, the homogeneous sensor networks lack these advantages. Thus, we use this multi-tiered architecture in this paper. Wireless sensor networks are usually deployed in hostile or dangerous environments, so the communications among nodes have to be encrypted. Furthermore, adversaries may capture or compromise nodes in the sensor networks. The keying material of the compromised nodes may be known by adversaries. Thus, the consequence of node capture attack has to be minimized. In this paper, we employ a polynomial to establish a pairwise key between any two nodes in a cluster. In addition, each sensor can also establish pairwise keys, in all probability, with its neighbors deployed in other clusters. The security analysis shows that our scheme provides better security, compared with the existing key establishment scheme.

Keywords—polynomial; key establishment; sensor networks;

I. INTRODUCTION

In homogeneous sensor networks, many existing researches [1], [3], [9] focus on the pairwise key establishment between any two neighbors by utilizing key pools, space pools of matrix or polynomials. However, these schemes suffer from poor scalability and premature battery depletions of nodes near the base station [4]. In addition, these schemes also have low network’s robustness against node capture attack because the effects of such an attack are usually network-wide. Obviously, we can see that the multi-tiered architecture used in this paper has many benefits. Recently, the authors in [5], [6] used pairing scheme to establish pairwise keys among sensors in homogeneous sensor networks, but these schemes introduce additional computation overhead. They are unsuitable for resource-constrained sensors.

Many existing researches show that there are better performance in heterogeneous sensor networks than in homogeneous sensor networks [2], [4], [10], [11], [12]. The schemes [2], [10], [12] are based on the key pool scheme proposed in [1], so the probability that two neighbors can’t establish a session key may happen when they don’t have any key match. In [11], the authors modified the key predistribution scheme used in [3], and they used the modified version to secure each link of nodes in heterogeneous sensor networks with robust continuity. Furthermore, the authors proposed a group key management protocol for the network. The participants can establish a group key by using the protocol, but this protocol introduces additional communication and computation overhead for nodes. In addition, the authors in [10] used the key pools [1] to secure the link between any two nodes in a cluster, and they also used pairing scheme to secure the link between any two cluster heads. The authors considered that unidirectional links may emerge among nodes, so they exploited unidirectional links to establish keys among nodes.

The rest of this paper is organized as follows. The related work is introduced in Section II. In Section III, we illustrate the $t$–degree trivariate symmetric polynomial and the network model used in this paper. In Section IV, we present our proposed method. Section V is the security analysis, and the conclusion is in Section VI.

II. RELATED WORK

In this section, we describe the related methods used in this paper. In most exiting schemes we describe before [2], [4], [10], [11], [12] for heterogeneous sensor networks, a trusted third party (e.g., the base station) is usually required to generate secret information or keys for the networks. We also assume that there exists a trusted base station for the purpose of generating a $t$–degree trivariate symmetric polynomial used in [13] and distributing the polynomial shares to nodes prior to network deployment.

In [2], [12], the authors used asymmetric key distribution scheme to find the pairwise keys if any two neighbors have the common predistributed keys. The main idea is that each cluster head is preloaded with $l$ keys which is randomly chosen from a large pool without replacement. In the same way, each sensor is also preloaded with $M$ ($M \gg l$) keys. Thus, the probability of sharing one key between a cluster head and a sensor in heterogeneous sensor networks will be larger than in homogeneous sensor networks [1]. Moreover, the security against node capture attack is more resilient. Unfortunately, these schemes may lead to the situation that two neighbors don’t have any common key to secure the link between them. Hence, they have to discover a secure path to help them in the pairwise key establishment through...
multiple nodes. It may introduce additional communication overhead.

The authors in [7] proposed a well-known protocol called \( \mu \)TESLA. They utilized one-way hash chains to generate delayed disclosure of symmetric keys for broadcast authentication. It also requires that the base station and nodes are loosely time synchronized. In this paper, we simply employ a one-way hash chain to generate a series of hash values as an input of the polynomial for each cluster head. Similarly, we also assume that the base station and cluster heads are loosely time synchronized.

### III. Preliminaries

#### A. \( t \) – degree trivariate symmetric polynomial

We use the polynomial proposed in [13] to achieve key agreement among nodes. The \( t \) – degree trivariate polynomial is described as follows.

\[
f(x_1, x_2, x_3) = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \sum_{i_3=0}^{t} a_{i_1,i_2,i_3} x_1^{i_1} x_2^{i_2} x_3^{i_3},
\]

where all the coefficients \( a_{i_1,i_2,i_3} \) are chosen from a finite field \( \mathbb{F}_q \). Suppose that each sensor is given two positive and pairwise different integers. For sensor \( A \), it has \( (a_1, a_2) \). Similarly, sensor \( B \) has \( (b_1, b_2) \). Prior to node deployment, sensors \( A \) and \( B \) are given \( f(a_1, a_2, x_3) \) and \( f(b_1, b_2, x_3) \), respectively. We call \( f(a_1, a_2, x_3) \) the polynomial share for sensor \( A \). Suppose that \( a_1 = b_1, a_2 \neq b_2 \) or \( a_1 \neq b_1, a_2 = b_2 \), sensors \( A \) and \( B \) can establish a session key. For example, if \( a_1 = b_1, a_2 \neq b_2 \), sensors \( A \) and \( B \) can establish a session key by performing the following equation, respectively.

\[
K_{AB} = f(a_1, a_2, b_2) = f(b_2, b_2, a_2).
\]

Similarly, if \( a_1 \neq b_1, a_2 = b_2 \), they can establish a session key by taking the different value of the other as the input.

#### B. The network model

We use a sensor network model, consisting of a base station, a small number of resource-rich cluster heads and a large number of resource-constrained sensors. All nodes are stationary after deployment. The base station is always trustworthy, and all cluster heads are equipped with tamper-resistant hardware. However, sensor nodes may be compromised by adversaries. If adversaries compromise a node in the sensor network, the keying material stored at the node will be exposed to them. The network area is divided into many clusters, and each cluster consists of a cluster head and numerous sensors.

### IV. The proposed method

#### A. Our scheme

1) **Initialization**: Initially, a \( t \) – degree trivariate symmetric polynomial \( f(x_1, x_2, x_3) \) is generated by a trusted third party, e.g., the base station. Prior to network deployment, each node in the sensor network is assigned a unique identity by the base station. For a sensor, say \( N_i \), the base station distributes a polynomial share \( f(h^0(X), ID_{N_i}, x_3) \) to it, where \( h(\cdot) \) is a one-way hash function, \( X \) is a secret value generated by the base station and \( h^i(X) \) denotes the \( i \)th hash value of \( X \). Moreover, the base station preloads a cluster head, say \( CH_j \), with the secret value \( X \) and a polynomial share \( f(ID_{CH_j}, x_2, x_3) \).

2) **Nodes deployment and clustering**: After finishing predistribution, all sensor nodes and cluster heads are uniformly and randomly deployed into a flat network that is a designated area. After network deployment, each cluster head periodically broadcasts a \( hello \) message including its identity to the nearby sensors with a random delay. We assume that most sensors can receive one or more \( hello \) messages in the sensor network. Each sensor will choose an appropriate cluster head as its cluster head according to the signal strength of the \( hello \) message and record the cluster head’s identity. This clustering scheme is similar to [2]. Eventually, the network is divided into many clusters, and each cluster is controlled by a cluster head. Note that the session key between any two neighboring cluster heads can be established by taking their identities as inputs. For example, cluster heads \( CH_1 \) and \( CH_2 \) can establish a session key by computing the equation \( K_{CH_1,CH_2} = f(ID_{CH_1}, ID_{CH_1}, ID_{CH_2}) = f(ID_{CH_2}, ID_{CH_1}, ID_{CH_1}) \). Similarly, each cluster head can also establish a session key with the base station. This method is straightforward, so we don’t describe it here.

3) **Session key establishment in a cluster**: Suppose that sensor \( A \) is deployed in the proximity of cluster head \( CH_j \). After choosing the corresponding cluster head for sensor \( A \), it computes a value \( f(h^0(X), ID_A, ID_{CH_j}) \) as the pairwise key \( K_{A,CH_j} \) shared between \( A \) and \( CH_j \). After that, \( A \) sends a \( join \) message including \( ID_A, ID_{CH_j} \), and a MAC value computed by \( K_{A,CH_j} \) to \( CH_j \). Upon receipt of this message, \( CH_j \) computes a value \( f(ID_{CH_j}, ID_A, h^0(X)) \) and checks if this computed MAC value is equal to the MAC value computed by \( A \). If so, \( CH_j \) responds with an \( accept \) message including \( ID_{CH_j}, ID_A, Nonce_{CH_j}, A’s \) encrypted share \( E_{K_{A,CH_j}}(f(ID_{CH_j}, ID_A, x_3)) \) used in this cluster, and \( MAC(K_{A,CH_j}, Nonce_A \oplus Nonce_{CH_j}, \ldots) \) to \( A \) which wants to join. Otherwise, a \( reject \) message will be sent to \( A \).

If the share possessed by \( A \) is correct, \( A \) will be accepted to join the cluster controlled by \( CH_j \). Otherwise, sensor \( A \) may be a malicious sensor be-
cause it doesn’t possess the correct polynomial share. Thus, each legal sensor can join the corresponding cluster and obtain its share used in this cluster. Finally, each sensor can establish pairwise keys with all sensors in this cluster by exchanging their identities and taking others’ identities as inputs. The reason is that all sensors in the cluster possess the same marginal polynomial \(f(ID_{CH}, x_2, x_3)\), so the equation \(K_{A,other} = f(ID_{CH}, ID_A, ID_{other}) = f(ID_{CH}, ID_{other}, ID_A)\) will be equal. How to find if the sensors in the cluster share the same marginal polynomial? For a sensor, a simple protocol is to exchange its corresponding cluster head’s identity with others. On the other hand, any two nodes can directly or indirectly communicate with each other via the pairwise key shared between them in a cluster.

4) Session key establishment across multiple clusters: At the beginning of network deployment, all legal sensors are preloaded with the same marginal polynomial \(f(h^0(X), x_2, x_3)\), so they can establish pairwise keys with the neighbors which may reside in other clusters. The key establishment scheme has been described before, so we don’t repeat it again. How to identify the shared marginal polynomial? A simple protocol called challenge-response can be used here [1], [9]. After establishing pairwise keys, all sensors have to erase their shares \(f(h^0(X), ID, x_3)\) from their memory immediately. We use the same assumption in [8]. There exists a lower bound on the time interval \(T_{min}\) for an adversary to compromise a sensor. The time to establish pairwise keys is smaller than the time to compromise a sensor. The advantage is that if a sensor wants to communicate with its neighbors residing in other clusters, it can directly communicate with them instead of sending messages to its cluster head which acts as a medium.

5) The hash value update: We use the same assumption in [7] that the base station and all cluster heads are loosely time synchronized. Time is divided into a series of equal time intervals. At the beginning of network deployment, the time interval is \(T_0\). At the time, the base station and all cluster heads keep the value \(h^0(X)\). In the next time interval \(T_1\), the value \(h^1(X)\) is computed by the base station and all cluster heads, and they erase the old value \(h^0(X)\) from their memory. Thus, in time interval \(T_t\), the new hash value \(h^{t-1}(X)\) will be generated by the base station and all cluster heads, and the old hash value \(h^{t-1}(X)\) will be erased by them.

B. Deploying new sensors

During the operation time of the sensor network, sensors may exhaust their energy or be compromised by adversaries. Deploying new sensors to the network is necessary. Suppose that a new sensor, say \(N_{new}\), will be deployed in time interval \(T_t\). First, the base station preloads it with the share \(f(h^t(X), ID_{N_{new}}, x_3)\). Second, sensor \(N_{new}\) is deployed into the sensor network via an airplane or other methods. No matter where \(N_{new}\) is deployed, it can always join a cluster. The reason is that each cluster head keeps the value \(h^t(X)\), so it can establish a pairwise key with sensor \(N_{new}\) according the equation \(K_{N_{new},CH} = f(ID_{CH}, ID_{N_{new}}, h^t(X)) = f(h^t(X), ID_{N_{new}}, ID_{CH})\). The follow-up steps are the same as above-mentioned steps. If adversaries compromise a sensor in time interval \(T_t\), they will fail to use the compromised sensor to join any cluster in the next time interval \(T_{t+1}\) due to the updated hash value \(h^{t+1}(X)\) of each cluster head. The situation to deploy only a sensor is seldom. We consider the situation that a large number of sensors are usually deployed to sensor network.

There is a high probability that these newly deployed sensors are neighbors due to the method of deployment. Thus, each newly deployed sensor can not only join a cluster where it resides, but also establish pairwise keys with its newly deployed neighbors residing in other clusters. The reason is that all newly deployed sensors are preloaded with the same marginal polynomial \(f(h^t(X), x_2, x_3)\).

V. SECURITY ANALYSIS

In this section, we analyze the resilience to node capture attack. Once adversaries capture more nodes, the more keying material is known. The scheme in [2] requires that at least \(q\) keys are needed to secure a link between two neighboring sensors. The probability that a link between two noncompromised sensors when \(x\) sensors have been compromised is

\[
P(l) = \sum_{i=q}^{x} (1 - (1 - l/P)^x)^ip(i)/p,
\]

where \(l\) is the number of keys stored at each sensor, \(P\) is the key pool size, \(p(i) = (M^{l-1})(M_{l-i-2})/(M^l(M_{l-i}))\) is the probability that a cluster head and a sensor have exactly \(i\) keys in common, \(M\) is the number of keys stored at each cluster head and \(p = p(q) + p(q + 1) + \cdots + p(l)\). In addition, assume that there are \(A\) nodes sharing a \(t\) degree marginal polynomial, \(Q\) is the total number of sensors in the sensor network. Thus, the probability that the polynomial is compromised when \(x\) sensors have been compromised is

\[
P_c = \sum_{i=Q}^{A} P_c(i),
\]

where \(P_c(i) = (\binom{A}{t-i}/\binom{N}{t})\) is the probability that \(i\) out of \(A\) sensors are compromised. For the scheme [2], we set: \(q = 1, P = 10,000, M = 125\) and \(l = 45\), since \(p \approx 0.5\). For our scheme, assume that there are \(N_1\) sensors in a cluster and \(N_2\) clusters. If adversaries compromise all \(N_1\) sensors in a cluster, they can construct
compromises once the degree adversaries compromise a sensor, they will fail to launch a wise keys among them if they are neighbors. Furthermore, if only join clusters where they reside, but also establish pairwise keys establishment among nodes in a cluster, since our scheme can ensure that any two nodes in a cluster can establish a pairwise key.

VI. Conclusion

In this paper, we use the \( t - \) degree trivariate polynomial to secure each pair of nodes in a cluster and the links between each sensor and its neighbors residing in other clusters. Our scheme can ensure that no matter where a sensor is deployed, the sensor can always join a cluster. When deploying new sensors into the network, they can not only join clusters where they reside, but also establish pairwise keys among them if they are neighbors. Furthermore, if adversaries compromise a sensor, they will fail to launch a sensor replication attack due to the updated hash value. Our scheme can also ensure the perfect security against node compromises once the degree \( t \) is chosen appropriately.

REFERENCES


