An Efficient and Polynomial based Key Distribution Approach for Wireless Sensor Networks

Ramu Kuchipudi, Dr.A.A.Moiz Qyser, Saith P

Abstract—Wireless Sensor Networks are highly vulnerable to attacks because, it consists of various resources constrained devices and they communicate via wireless links. The most fundamental security problem in WSN is key management that covers the establishment, distribution, renewing and revocation of cryptographic keys. Several key management protocols were proposed in the literature. Unfortunately, most of them are not resilient to nodes capture. This means that an attacker compromising a node can reuse the node’s key materials to populate any part of the network with cloned nodes and new injected nodes. In this article, we present a simple polynomial-based key management protocol using a group-based deployment model without any necessary predictable deployment location of nodes. That solution achieves high resilience to nodes compromising compared with other protocols.

Index Terms—Key Distribution schemes, Sensor Networks, Polynomials, WSN Security, Key Management, Nodes Compromising, Intrusion Detection

I. INTRODUCTION

A wireless sensor network (WSN) is a network formed by a large number of sensor nodes, each equipped with sensor(s) to detect physical phenomena such as heat, light, motion, or sound. Using different sensors, WSNs can be implemented to support many applications including security, entertainment, automation, industrial monitoring, public utilities, and asset management. However, many WSN devices have severe resource constraints in terms of energy computation, and memory, caused by a need to limit the cost of the large number of devices required for many applications and by deployment scenarios that prevent easy access to the devices. Such resource limitations lead to many open issues — including WSN security — which have been studied actively by researchers.

Wireless sensor networks (WSN) are infrastructure-less and self-organizing networks, which can be deployed anywhere, and work without any assistance [1]. These characteristics motivated their deployment, but introduced critical security issues like network control access, authentication, confidentiality and nodes compromising. WSN are very sensitive to those issues since sensors are known to be tamper-vulnerable devices [2], and deployment of them is mostly done in open area that should be assimilated as hostile area for security consideration.

Current WSN security solutions rely on secret keys but today an efficient key management protocol is still needed to generate, distribute, renew and revoke cryptographic keys. In the last few years, several key management protocols for WSN have been defined, but they do not satisfy the protocol efficiency requirements Low storage, computation, and transmission overheads. Compromised nodes or injected false nodes, by reusing their key materials. Resistance to nodes compromising, so keys established between non compromised nodes remain confidential even in case of nodes compromising. No on-line key management server, that would be the point of failure in case of DoS attacks.

Resistance to nodes compromising that prevents attackers to populate the network with clones of Most of the key management protocols [2] [3] [4] [5], satisfy one or more of the three rst requirements. Unfortunately the last requirement is rarely or not enough investigated. Most of the key management protocols are poorly resilient to nodes compromising, and the few ones achieving an acceptable level of resilience, either rely on strong assumptions (i.e. nodes are tamper-resistant) [6] [4], introduce heavy overheads (the use of public key cryptography) [6], or require prior knowledge on nodes deployment [4].
In this paper, we propose a simple key management protocol for static WSN, based on the well-known polynomial-based key generation protocol of [7] for pairwise keys establishment, and our proposed group-based deployment model to ensure resilience to nodes compromising. Our protocol requires no prior knowledge on the locations of deployed nodes. It relies on realistic assumptions, and introduces no significant overhead.

II. ASSUMPTIONS AND NETWORK MODEL

A. Assumptions

First, we suppose that the Base Station (BS) is a trusted and a powerful entity in the network that cannot be compromised. Second, we suppose that sensors are static, so once they are deployed they do not leave their locations. In many scenarios (i.e. perimeter monitoring), WSN are considered as static, either because sensors are fixed or because sensors are not asked to be mobile for achieving their tasks.

Third, we suppose that sensors are deployed progressively in successive generations (groups). This assumption is adopted in most group-based deployment key management protocols like [3] and [8]. However, unlike the other group-based key management protocols, we do not suppose that nodes of the same generation are deployed in the same neighborhood. In our protocol, nodes of the same generation might be deployed anywhere in the network. Therefore, our protocol is not based on any prior knowledge on deployment location of nodes, but if such information was available, our protocol will achieve better resilience. Fourth, we suppose that an attacker needs a minimum time $T_{comp}$ in order to compromise a node after it is deployed. $T_{comp}$ is greater than the time $T_{test}$, which is the maximum time needed by a newly deployed node to establish pair-wise keys with its one-hop neighbors. This assumption is present in several works like [9] and [5], and is likely to be true, because an attacker must first have a physical access to a sensor, and then use some programming tools in order to retrieve sensor's key materials. However, in [9] and [5], deployed devices are initially loaded with some common secrets that nodes use to establish different keys (pair-wise keys, cluster keys) with their neighbors. In addition, [9] and [5] require that each newly deployed node erases these common secrets after a time $T_{test}$ from its deployment, to prevent that an attacker can get them if it is compromised.

In our protocol, no such assumption exists, because each node only needs its unique secret polynomial share for pair-wise key establishment. Fifth, we suppose that sensors are synchronized with the BS. This could be done through an authenticated beacon periodically broadcasted by the BS, to keep sensor's clocks synchronized with the BS's one. Authentication can be guaranteed using the µTesla protocol [10].Finally we suppose that an attacker can only get a partial control over the network. In case of full control on all deployed nodes, security solutions will be inoperative to stop the attacker.

B. Network Model and Security Considerations

The BS deploys nodes in multiple generations numbered successively from 1 to $n$, where $n$ is the maximum number of deployed generations. The order of deployment must be respected $G_1, \ldots, G_i, \ldots, G_n$, where $G_i$ is the $i^{th}$ deployed generation. Each node belongs to a unique generation.

Because nodes are not mobile in our network, it is logical that only nodes of the newly deployed generation ask for key establishment with their neighbors, which may belong either to the same generation, or to former deployed generations.

Nodes of former generations can not request for key establishment, and even if they do request, their requests must be rejected. Based on this assumption, we can state that any key establishment request originates from either a node from the newly deployed generation, or a node deployed by an attacker, which is either a node having a false Id, or a cloned node having the Id of a compromised node.

For security reasons, we suppose that any newly deployed node $u$ sets timer to the value $T_{test}$ straight after deployment. Once the timer expires, node $u$ rejects any key establishment request originating from a node of the same newly deployed generation.
A. Location-aware Key Establishment in Wireless Sensor Networks

Liu et al. [4] propose a distributed location-aware key establishment protocol, based on bivariate symmetric polynomials. The protocol assumes that the network is formed by simple nodes, and some sufficiently dedicated nodes called the service nodes, which are elected amongst sensors after deployment. These nodes play the role of trusted key servers in the network and are assumed to be non-compromised. The protocol also assumes that once nodes are deployed, they know their exact location coordinates \((x, y)\).

After deployment, each service node creates a distinct t-degree bivariate polynomial, and then securely initializes each neighbor node with its secret polynomial share, using the unique location coordinates of the node. The protocol is resistant and resilient to node compromising, as long as the service nodes are not compromised, and there are fewer than \(t+1\) compromised nodes initialized by the same service node. However, if a service node is compromised - which is a current threat because a service node is just a non-tamper resistant sensor node - an attacker can inject clones and new nodes with new positions, deploy them in the neighborhood of the compromised service node, and establish secure links with any nodes of the network.

IV. LIGHTWEIGHT KEY MANAGEMENT IN WIRELESS SENSOR NETWORKS

Dutertre et al. [9] suppose that nodes are deployed in \(n\) successive generations, and cannot be compromised in a period of time less than \(T_{\text{comp}} > T_{\text{est}}\). Each generation is loaded with a unique two master keys, used respectively for authentication and key generation between the nodes of the generation. Once a deployed node successfully establishes secure links with its neighbors of the same generation, it erases these two keys to prevent from attacks. In order to establish secret keys between nodes of two different generations, each generation \(i\) is also loaded with a unique secret group key \(GK_i\) that enables nodes to establish secure links with previously deployed generations \(j = 1 \ldots i-1\). In addition, each node \(u\) of generation \(i\) is loaded with a unique random value \(R_u\), and a secret key \(S_{uj} = H(GK_i, R_u)\) for each future generation \(j = i + 1 \ldots n\), allowing it to establish secure links with nodes of newly deployed generations.

The group key \(GK_i\) is also deleted at the end of the key establishment phase. The protocol is poorly resilient to nodes compromising, as an attacker compromising a node of generation \(i\), can establish secure links with nodes of the future deployed generations, using the compromised secret keys \(S_{uj}\). Moreover, if an attacker compromises a node \(u\) of generation \(i\) in a time period less than \(T_{\text{est}}\), he might retrieve the master keys of generation \(i\), and the group key \(GK_i\). As a consequence, he can deduce secure links established between nodes of generation \(i\), and can inject cloned nodes and false nodes anywhere in the network.

A. Distributed Approach to Security in Sensornets

Bhuse et al. propose a key distribution protocol based on the use of the Hughes's variant of the DH protocol with encrypted key exchange (HDH-EKE) and based on the assumption that nodes cannot be compromised, and even if they are, then they self-destroy without revealing their secret cryptographic materials. All nodes are initially loaded with a common password \(P\) used for authentication, and after deployment, nodes self-organize into clusters, and elect one of them to act as a key server. The key server of each cluster generates a cluster key, and securely distributes the key to each one-hop neighbor using the HDH-EKE protocol, which in turn will pursue the distribution of the key to its neighbors in the same manner, until all nodes of the cluster possess the cluster key.

The cluster key is used for encrypting messages and authenticating them inside the cluster. In order to send packets between two different clusters, boarding nodes, which possess the cluster keys of two or more clusters, will act as a gateway by decrypting/encrypting messages from the source cluster to the target cluster. The key server periodically updates the cluster key, by sending a random counter value used along with the secret password, and the current cluster key to produce the new cluster key. The main problem of this protocol is its high computation overhead due to the use of modular exponentiations (public key cryptography), its weak authentication mechanism. The protocol is resilient against nodes compromising as long as an attacker cannot retrieve the secret common password \(P\). However, it's unlikely that sensors can be tamper resistant [10], where memory containing the secret
cryptographic materials is hardware-protected, because this will increase significantly the cost of sensors, and sensor nodes are intended to be very inexpensive.

V. PROPOSED SOLUTIONS
We propose a resilient key management protocol, based on the use of a symmetric polynomial for secure key establishment, and based on our defined group-based deployment model for achieving resilience to nodes compromising.

Our protocol involves three phases:
- Initialization Phase
- Pair-wise key establishment Phase
- Key-path establishment Phase

A. Initialization Phase
Initially, the BS generates a random symmetric bivariate polynomial \( f(x, y) \) The BS then selects a group of nodes to form the next deployed sensors generation. The BS loads each node \( u \) with a unique secret polynomial share as follows:

\[
\begin{align*}
    f(x, y) &= \sum_{i,j=0}^{t} a_{ij} x^i y^j \mod(Q); \\
    f(x, y) &= f(y, x)
\end{align*}
\]

Where \( Q \) is a sufficiently great prime number, \( 1 < a_{ij} < Q - 1 \), and \( t \) is the degree of the polynomial and a security parameter.

Initially, the key server configures each node \( u \) with its unique secret polynomial share:

\[
    f_u(y) = f(Id_u, y) = \sum_{i,j=0}^{t} a_{ij}(Id_u)^i y^j \mod(Q);
\]

Where \( Id_u \) is the unique identifier of node \( u \) in the network. Two nodes of the network \( u \) and \( v \), can easily establish a unique shared secret key by computing:

\[
    K_{uv} = f(Id_u, Id_v) = f(Id_v, Id_u) = K_{vu};
\]

B. Pair-wise Key Establishment Phase
Suppose that the BS deployed some previous generations, say the \( i \) first generations (1, 2... \( i \)), and just deployed generation \( i + 1 \). In our protocol, nodes know the highest deployed generation's number \( i + 1 \) through a mechanism we describe let \( u \in Gj \) a newly deployed node. It is obvious that as a well-behaving node, \( u \in Gi+1 \). Node \( u \) tries to establish secure links with its direct neighbors by locally broadcasting a Hello message:

\[
    u \rightarrow * : \text{Hello, } j, Id_u, N_u
\]

Where \( N_u \) is used to guarantee response freshness. Let \( v \in Gz \), where \( z \leq i + 1 \), a neighbor node of \( u \) receiving its message. For node \( v \) to decide serving node \( u \), node \( v \) follows two steps:

1. \( v \) checks if \( j = i + 1 \), to verify whether \( u \) belongs to the newly deployed generation. If the verification fails, it simply rejects the request of node \( u \), because \( u \) is normally already deployed.
2. If \( v \) verifies that \( j = i + 1 \) then:
If \( v \) belongs to generation \( z \leq i \), then \( v \) computes \( K_{vu} = f_v(i + 1||Id_v) \) and sends back to node \( u \) the following message:

\[
v \rightarrow u : z, Id_v, Nu, MACK_{vu}(z, Id_v, Nu, Nu)
\]

If \( j = z = i + 1 \) (\( u, v \in G_{i+1} \)), then

1. If the timer set by node \( v \) did not expire, do the same treatment as the previous case.
2. If the timer expired, reject the request, because node \( u \) is suspected to be malicious.

Upon receiving node \( v \)'s message, node \( u \) computes \( K_{uv} = f_u(z||Id_v) \), and checks the message authenticity. If the message is authenticated, node \( u \) sets \( K_{uv} \) as the shared pair-wise key with \( v \), and sends to \( v \) the following message to conclude the key establishment process:

\[
u \rightarrow v : ok, MACK_{uv}(ok, Nu)
\]

Upon receiving node \( u \)'s response, node \( v \) checks the message authenticity using \( K_{uv} \) and, if successfully done, node \( v \) sets \( K_{vu} \) as the shared pair-wise key with \( u \), otherwise (failed authentication, or non-received response), it erases \( K_{vu} \).

At the end of this phase, either a pair-wise key is established between two valid nodes, or the pair-wise key establishment fails in case one of the two nodes is suspected of being a clone or a false node. The described protocol guarantees that any served key establishment request originates from a newly deployed node belonging to generation \( i+1 \). However, the current version of the protocol fails to detect two particular attempts of false key establishment. The first attempt is when an attacker compromises a newly deployed node of generation \( i+1 \) and deploys clones in the neighborhood of nodes of older generations, and the second attempt is when an attacker compromises an older deployed node, and tries to respond to the Hello messages of the newly deployed nodes.

**C. Key-Path Establishment Phase**

In our scheme, two non-neighboring nodes might attempt to establish a secret key. The two nodes might belong to the same generation, or to two different generations. Moreover, the node initiating the establishment might be from a higher generation, same generation, or lower generation than the target node.

This raises a problem because we supposed that only newly deployed nodes are eligible for requesting key establishment. We rely on the previously established pair-wise keys in order to overcome this problem, and to guarantee each node that the other node is a valid node and not a cloned node or a false injected node.

Let \( u \in G_i \) and \( v \in G_j \) be two distant nodes that need to establish a secret key, and consider that \( u \) initiates the communication, and \( u \) knows \( Id_v \), and that \( v \in G_j \). First, \( u \) must find a secure path to node \( v \), formed by previously established secure links. Once the path is found, node \( u \) sends to \( v \) the following Key Establishment Request (KER) message:

\[
u \rightarrow v : i, Id_v, Nu, MACK_{uv}(i, Id_v, Nu) \text{ where } K_{uv} = f_u(j||Id_v).
\]

The KER message is sent in a secure path, where each node in the path, including \( u \) and \( v \), authenticates the message with the key it shares with the previous node in the path. Upon receiving the KER message, node \( v \) computes \( K_{uv} = f_v(i||Id_u) \), and then checks the authenticity of the message. If the KER message is authenticated, then \( v \) sends back to \( u \) the Key Establishment Confirmation (KEC) message:

\[
v \rightarrow u : ok, MACK_{uv}(ok, Nu)
\]

To conclude the key-path establishment. Node \( u \) checks the authenticity of the KEC message, and in case of unsuccessful authentication, it erases \( K_{uv} \). Note that thanks to the secure path, each node is ensured that the other node is neither a cloned node nor an injected node. Indeed, suppose that the path is going through nodes: \( u, w_1, \ldots, w_r, w_{r+1}, \ldots, w_s, v \).
According to, each pair-wise key in the network is established between two valid nodes, which are neither cloned nodes nor false injected nodes. As a consequence, the fact that \( u \) found a secure path and received the KEC message means that node \( v \) is a valid node, and the fact that node \( v \) received the KER message through the path, means that node \( u \) is a valid node.

### VI. SECURITY ANALYSIS OF PROPOSED SOLUTION

#### A. Resilience to nodes compromising

Now, let consider the resilience to nodes compromising, which refers to the capability of an attacker to inject cloned nodes and new nodes in the network, using the key materials it gets from the compromised nodes.

#### B. Injecting nodes with false Ids

It is clear that as long as an attacker compromises fewer than \( t+1 \) nodes, new nodes with non-existing Ids can not be injected in the network. In our protocol, each node \( u \) possesses a unique polynomial share bound to its identity \( f_u = f(i||Id_u, y) \), where \( u \) belongs to \( G_i \).

After compromising node \( u \), an attacker can not create node \( u' \), with a new identity \( Id_{u'} \) and a new polynomial share \( f_{u'}(y) = f(i||Id_{u'}, y) \). In addition, an attacker can not use the polynomial share \( f_u \), because node \( u' \) will fail to establish secret keys with this polynomial share using the new identity \( Id_{u'} \). As a conclusion, our protocol is resilient to the injection of false nodes with non-existing identifiers in the network.

#### C. Injecting cloned nodes

In our protocol an attacker is highly unlikely to deploy cloned nodes, and convince his neighbors of the validity of the clones.

### VII. COMPARISON WITH PREVIOUS WORK

As we have seen in the description of some previous works done in the literature, most of them lack to provide resilience to nodes compromising, and those providing some degree of resilience rely on some assumptions, that cannot be met easily. For essence, [6] and [4] suppose that nodes are tamper resistant devices, so they cannot be compromised or they self-destroy when they detect that they are under attacks, and [4] relies also in the assumption that nodes know their locations coordinates, in order to tie each node's secret key material (i.e. its secret polynomial share) to its location coordinates, so even if in attacker succeeds into compromising a node and creates some cloned nodes, it cannot deploy them anywhere in the network. Someone can suppose that the future generation of sensors will be tamper-resistant. However, tamper-resistant devices are expensive, and constructor’s tendency may be to keep sensors at lower prices, with an increase of memory and computation capacities, instead of making them tamper-resistant.

Localization in WSN is still under research, and the actual solutions assume that either nodes are equipped with GPS receivers, or the existence of some trusted nodes (at least three) on the perimeter of the network, that provide nodes with their locations coordinates. The first solution is unlikely to be deployed in sensors, and is energy consuming, and the second solution requires multiple trusted entities, and the resulting location co-ordinates are prone to errors. In our protocol, we don't assume that nodes are tamper-resistant devices, and we don't consider nodes locations. Some other works like [5], [9] suppose that nodes share some common secret key(s) they use for key establishment, which will be erased from their memory straightforward after they finish key establishment when they are first deployed.

### VIII. CONCLUSION

We proposed a key distribution scheme that has a very good resilience to node capture. Any number of node’s capture doesn’t affect even a single link in the remaining network. Also it has many other advantages over probabilistic key distribution scheme and deterministic key distribution scheme. Compared to deterministic scheme, it is more secure, requires less space for keying material, have full network connectivity even after compromising a great fraction of nodes in the network and doesn’t have compromised links between non-compromised nodes.
Our proposed solution improves considerably resilience to nodes compromising compared with other protocols, and does not require any prior knowledge relative to nodes deployment, and any common secret key pre-establishment between nodes. Moreover, the solution does not rely on non-realistic assumptions like supposing that compromised nodes do not divulge their secret keys or that some nodes in the network can not be compromised. Our protocol uses t-degree polynomial-based key generation protocol for achieving resistance to nodes compromising, and the proposed group-based deployment scheme for resilience against nodes compromising, where only nodes of the newly deployed generation ask for key establishment.

REFERENCES


AUTHOR BIOGRAPHY

DR AHMED ABDUL MOIZ QYSER is Professor and Head of Department of Computer Science Engineering in MJCET, Banjara Hills, and Hyderabad. He has completed Ph.D. from OU, Hyderabad. He is a Member of ACM, Life Member of CSI and ISTE

K.Ramu is working as assistant Professor in MJCET, Banjara Hills, and Hyderabad. He has completed M.Tech from JNTUH. He is a Life Member of ISTE.

Saith P is studying B.Tech (IT) in MJCET, Banjara Hills, and Hyderabad. He is a member of IEEE.