Use of Symmetric And Asymmetric Cryptography in False Report Filtering in Sensor Networks

Aleksi Toivonen
Helsinki University of Technology
Aleksi.Toivonen@tkk.fi

Abstract

Sensor networks are easily deployable networks of inexpensive sensor nodes. The sensors detect events of interest, for example forest fires, battlefield events, and so on. The sensor nodes form an ad-hoc network by communicating with each other using short-range radio. However, since the sensors are “in the field”, they may be compromised by adversaries. Compromising a sensor node gives the attacker keys and any other stored material, making it possible for him to inject false reports to the network. So far, mainly symmetric methods have been proposed to detect and filter out such false reports - the low computational resources available to the nodes set constraints on the use of cryptography. However, recent research has shown that asymmetric methods may also be feasible. The purpose of this paper is to review both methods; so far, symmetric methods seem to tackle the problem of false reports more efficiently than asymmetric ones, but the issue remains not yet adequately solved.

KEYWORDS: Pervasive Networks, Sensor Networks, Node Compromise, Symmetric, Asymmetric, Cryptography, False Report Filtering

1 Introduction

1.1 Sensor Networks

Sensor networks deploy sensor nodes that can communicate with each other and with a base station called the sink. The sink is a data collection center that collects reports about events detected by the nodes, and has laptop-level computational resources. Because the nodes have low power, they can only communicate over short-range radio, which means that they have to forward other nodes’ reports. In some schemes a sensor node acts both as a sensor and a forwarding node, in others the roles are separated.

The dense deployment of sensors makes it possible for several sensors to detect the same event, adding reliability to the system. The nodes select one node from their neighborhood to be the cluster head, and all sensor nodes in the cluster send their reports to this cluster head. The cluster head collects reports and forwards them to the sink, possibly via many forwarding nodes, see Fig. 1. If a node in the route fails to forward messages, the network may be rerouted via other nodes.

The nodes can encrypt and authenticate their communication with each other and with the sink using cryptography. Secure communication is needed when two neighbor nodes communicate or when the sink authenticates itself to the nodes. Before the network is deployed, the nodes and the sink can have their keys predistributed in a safe manner. Later, if they need to reset keys, a key agreement or distribution protocol is needed.

The rest of the paper is organized as follows. The rest of the introduction gives a short overview of symmetric and asymmetric cryptography and defines the problem setting. We define research methodology and criteria by which methods are evaluated in Section 2. In Section 3 we evaluate various symmetric methods in regard to our problem setting. Section 4 does likewise to asymmetric methods. Section 5 presents a short comparison of these two method categories. Finally, in Section 6, we draw our conclusions.

1.2 Symmetric And Asymmetric Cryptography

Symmetric cryptography relies on a shared secret key between two parties to enable secure communication. Asymmetric cryptography, on the other hand, employs two different keys, a secret one and a public one. The public key is used for encryption and can be published. The secret private key is used for decryption. From computational point of view, asymmetric cryptography requires orders of magnitude
more resources than symmetric. Until recently, only symmetric cryptosystems have been proposed because of this [4].

1.3 False Data Injection Attack
Malicious outsiders who wish to compromise the network’s operation may launch attacks against it. All attacks that plague wireless networks in general work here as well. Wireless transmissions may be eavesdropped on, jammed, messages injected and replayed. There may be transmissions originating from outside the network. The man-in-the-middle -attack and denial of service attack are examples. These can be classified as outsider attacks.

In addition, sensor networks have additional challenges. The sensors have low power and other resources, limiting the use cryptography. They are deployed in possibly hostile environments, making sensor damage, destruction, and capture possible. When a malicious outsider gets cryptographic material from a captured node, he may launch attacks that fall under the category of insider attacks, which are especially dangerous.

Tackling the problem with tamper-proof or tamper-resistant hardware is not economically feasible [11] for sensor networks. Therefore the problem must be addressed by cryptography.

The attack scenario we are considering is as follows: the attacker gains access to unattended sensor nodes and extracts cryptographic material from them. Later, the attacker launches an attack injecting bogus information to the sensor network pretending that it was sent from the original nodes. This may cause wrong upper level decisions, e.g. sending rescue workers to wrong location. Zhu et al. call this type of attack a false data injection attack [11]. This attack is especially interesting, because it is dangerous and quite feasible to execute from the attacker’s point of view. Our intent here is to examine, how recent symmetric and asymmetric methods can detect and prevent such an attack.

2 Methodology and Evaluation Criteria
Our examination of the methods is based on a literature survey. The criteria by which we evaluate each method are:

- The method’s ability to detect the false data injection attack.
- The method’s ability to prevent the false data injection attack.
- The method’s feasibility for sensor networks from resource consumption point of view.

Detecting the actual node capture is out of scope of this paper.

3 Evaluation of Symmetric Methods
In this section, we examine security schemes based on symmetric methods and their effectiveness against the false data injection attack (Sec. 1.3). To date, the scientific community has concentrated mainly on symmetric methods [2, 3, 5, 7, 8, 9, 11, 10].

3.1 LEAP
LEAP (Localized Encryption and Authentication Protocol) is a key management protocol for sensor networks. It divides keys into four categories: individual keys, pairwise keys, cluster keys and group keys. Individual keys are used in node and sink communication, pairwise keys between neighboring nodes, cluster keys between nodes in the same cluster and group keys globally in the system [10].

LEAP is resistant to outsider attacks, but it succumbs to insider attacks, as only one node compromise is needed to inject false data into the system [11]. However, LEAP does localize the compromise to the neighborhood of the node, as gaining access to the compromised node’s keys does not help compromising other nodes. LEAP does not help identifying that a node sends false data, but it is an efficient protocol in sensor networks, and can be used for key management in other security schemes.

3.2 SEF
The SEF mechanism (Statistical En-Route Filtering) [8] detects and drops bogus reports from compromised nodes. It employs MACs (Message Authentication Codes) in the detecting and forwarding sensor nodes. A MAC is produced by an algorithm, that takes as input a secret key and a message of arbitrary length, and produces a tag (called MAC), that is used to verify the message’s integrity and authenticity.

Each node that has detected the same event, verifies that the MACs of the reports it has received about the event are consistent with its own detection, and drops wrong ones, saving resources in the system. The verifying of MACs is done probabilistically. The nodes have predistributed keys from a global pool, and each node gets a random subset of the keys, to be used in producing the MACs. Because nodes have common keys with some probability, many nodes are needed to produce a common, verified event report.

SEF can’t detect which nodes are compromised, because reports are filtered en-route probabilistically, but it can prevent the false data injection attack with 80 - 90 percent probability within 10 hops. In other words, it is not very efficient in this regard.

3.3 A Fault-localized Scheme by Zhou et al.
Zhou et al. propose a fault-localized scheme [9] to localize the impact of node compromise. Their central idea is to separate the roles of detecting and forwarding nodes. Detecting nodes only generate reports about events, and forwarding en-route nodes verify the correctness of these reports. Therefore en-route nodes can’t generate false reports in the first place.

Each detecting node has its own key that it shares with the sink as well as a commitment. The key is used by the sink to verify the report’s authenticity, the commitment is used in a hash chain to verify a report’s freshness in en-route nodes.
A hash is fixed-length output produced by a one-way function. A new commitment from the hash chain is computed as follows:

\[ c_i^j = F(c_i^{j+1}) \]

Where \( c \) is commitment, \( i \) is sensor index, \( j \) is jth commitment, and \( F \) is a one-way function.

When a node generates a report, it encrypts the report and the next commitment from hash chain with the key it shares with the cluster head, as well as a MAC of the key and the report, and sends these to the cluster head. The cluster head collects reports from at least \( t+1 \) nodes and sends the sink the IDs of the detecting nodes, their commitments, commitment indices, and a XOR of the MACs (\( t \) is a system parameter).

If an en-route node tries to inject false data, it is immediately detected, because it doesn’t have the shared key with the sink. At least \( t+1 \) nodes need to collude together to produce a false event report, otherwise the report is discarded at the cluster head. Malicious en-route nodes need to forge \( t+1 \) hash values to generate a false report. An uncompromised en-route node detects this and drops the report. An individual or few compromised nodes can thus be detected.

This scheme works for sensor networks, because en-route nodes need only to store commitments, not keys. However, if many routes pass through an en-route node, the storage requirement becomes preventive, as it needs other nodes’ commitments to verify freshness. In other words, this scheme does not scale well.

### 3.4 An Interleaved Hop-By-Hop Scheme

The paper by Zhu et al. [11] introduces a scheme where the sink can detect false information if less than \( t \) nodes are compromised. All nodes are detecting nodes and forwarding nodes, generating reports about events, forwarding them, and verifying report correctness. At least \( t+1 \) nodes must agree on a report for it to be considered valid. Reports are sent to the cluster head, which collects them and forwards them to the sink.

Report validity is guaranteed by MACs. All nodes compute two key MACs on a report, one with the key they share with the sink and another with the key they share with their neighbor.

Unlike the fault-localized scheme by Zhou et al. [9], this scheme does not separate the roles of detection and forwarding. Forwarding nodes might also fabricate reports. Faults cannot be localized, because any node may be generating false data.

Each node must act on its “own behalf”, because of the authentication of each node’s neighborhood. A node knows the set of its authentic neighbor nodes after the initialization and deployment phase. The correct association may not be guaranteed, however, and localizing node compromise may fail [9].

The interleaved hop-by-hop scheme supports in-network processing (aggregation), reducing the need for required computational efforts. It also scales well, for a symmetric method.

### 4 Evaluation of Asymmetric Methods

Asymmetric methods, i.e. public key cryptography (PKC), provide many benefits in comparison with symmetric cryptography, but the burden they put on low-resource sensor nodes has discouraged their use. That said, some recent research seems to validate the use of PKC in sensor networks [1, 6].

Symmetric algorithms face the key agreement problem - the nodes must agree on a secret key to exchange information. This is a problem in open environments. In addition, pairwise keys are not scalable. And lastly, symmetric cryptosystems do not provide nonrepudiation [6]. Asymmetric methods solve these problems.

In the following, the effectiveness of some proposed asymmetric methods against the false data injection attack is evaluated.

#### 4.1 Public Key Algorithms

Gaubatz et al. [1] compare three different public key algorithms securing wireless sensor networks to show that PKC can be used effectively in such environments. They are Rivin’s Scheme, NtruEncrypt and NtruSign, and ECC (Elliptic Curve Cryptography). These algorithms are presented in the paper as proof-of-concept, showing that it might be possible to introduce PKC to sensor networks. The challenge that we face here, however, is an insider attack, namely, the false data injection attack. Public key cryptography alone cannot solve this problem, because node compromise in any event causes all cryptographic material to leak, including private keys. Private keys being the basis for authentication, PKC systems should somehow tackle the same issue as symmetric systems: how to prevent a malicious insider from injecting false data into the system? PKC is powerful for preventing outsider attacks, but suffers from the same weakness as symmetric methods.

In short, PKC localizes node compromise to the immediate neighborhood, because attacker gains only the compromised node’s private key, but does not detect or prevent the false data injection attack in a foreseeable way.

From resource consumption point of view, PKC requires a lot more power consumption, but scales well, because each node needs only the public keys of its neighbors to communicate. PKC can used to by two neighbor nodes to negotiate a secret key for symmetric encryption.

#### 4.1.1 Identity-Based Encryption

The authors of [6] maintain that identity-based encryption (IBE) and wireless sensor networks have great synergy. In IBE, nodes use their IDs as their public keys, so a PKI (Public Key Infrastructure) is unnecessary. In general use a PKI is required, so that communicating parties can authenticate their public keys. A third, trusted party issues digital signatures for keys. Without a PKI, PKC in sensor networks is at least a possibility - no additional storage and communication overhead is generated. Also IBE’s requirement of an unconditionally trusted entity is solved by the sink. In other words, IBE might prove that PKC is feasible in sensor networks, at least from resource consumption point of view. In an actual
5 Comparison of Symmetric And Asymmetric Methods

Symmetric methods solved the false data injection attack by introducing MACs, hash chains and commitments, separated roles for detecting and en-route nodes, different keys for different communication types, and a required threshold of reports. Symmetric methods are feasible, because their energy consumption is low, but suffer from key management issues. They scale less well than asymmetric methods.

Asymmetric methods have the status of a new proposal in relation to sensor networks. No actual PKC system has yet been proposed for sensor networks. Research needs to be done to make PKC practical; PKC’s energy consumption is high and insider attacks, including the false data injection attack, need particular attention. All things considered, PKC may help symmetric methods to form secret key pairs between two parties.

Table 1 presents comparison between symmetric and asymmetric methods.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Symmetric Methods</th>
<th>Asymmetric Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability To Detect</td>
<td>LEAP does not detect, SEF and Interleaved Hop-by-Hop Scheme can detect the attack, but not which nodes are compromised, Fault-localized Scheme detects individual nodes</td>
<td>Not yet implemented</td>
</tr>
<tr>
<td>Ability To Prevent</td>
<td>LEAP cannot prevent, SEF can prevent with 80-90 percent certainty within 10 hops, Interleaved Hop-by-Hop Scheme and Fault-localized Scheme can prevent if at most compromised nodes</td>
<td>Not yet implemented</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Low power consumption; high overhead; scales poorly</td>
<td>High power consumption; low overhead; scales well</td>
</tr>
</tbody>
</table>

Table 1: Comparison of symmetric and asymmetric methods.

6 Conclusion

The false data injection attack, and insider attacks in general, remain a central issue in making sensor networks secure. Insider attacks are especially dangerous, and their detection and prevention is a challenge, especially because the use of cryptography has some constraints in sensor networks. So far, symmetric methods seem to be the best available solution to the false data injection attack, as research in their field has been more extensive, and some solutions have solved the problem adequately to some degree, e.g. (Sec. 3.3) sets a limit to the number of compromised nodes. The field is quite new, and proposes many challenges yet to be addressed.

References
