Adaptive Key Management for Wireless Sensor Networks

Tony Cheneau
National Institute of Standards and Technology
Emerging and Mobile Network Technologies Group
Gaithersburg, Maryland, USA
Email: tony.cheneau@nist.gov

Mudumbai Ranganathan
National Institute of Standards and Technology
Emerging and Mobile Network Technologies Group
Gaithersburg, Maryland, USA
Email: mudumbai.ranganathan@nist.gov

Abstract—Network security protocols for wireless sensor networks (WSN) require each node to store context information, such as cryptographic keys, for each active security association that it maintains with its peers. While the storage requirements for such context information is generally small enough to be overlooked in regular networks, it becomes an important issue in memory-constrained WSN. Under such constraints, a WSN node may become memory saturated and thereby become unable to accommodate new nodes dynamically. What can be done to keep expanding the network with newly joining nodes under such memory constraints? How many security associations are ‘enough’ and how does the choice of the number of associations affect performance and coverage? To address these issues, we introduce a novel authentication and key management mechanism called Adaptive Key Management (AKM) that is robust and scalable under limited memory constraints and works along with the RPL routing protocol to provide strong security guarantees. We present a simulation study of AKM to demonstrate its scalability and to provide insight on how to allocate a node’s memory to store a sufficient number of security associations.

I. INTRODUCTION

A simple key management scheme for Wireless Sensor Networks (WSN) may involve each node establishing a secure link with every other node in its neighborhood. In a dense network where each node may have many neighbors if a such a simplistic scheme were adopted, the storage requirements for storing a large number of certificates, keys and additional session-related data would result in an inefficient use of the available resources and can lead to memory saturation (i.e. inability to establish additional security associations). If a sensor node incorporates a tamper-proof security module for storing cryptographic material, memory capacity for storing cryptographic material can be limited. Memory limitations are a good reason for designing key management protocols that explicitly limit the number of security associations that a node maintains. Such a restriction also results in lessened vulnerability as each node stores fewer session keys. Hence if a node is compromised, less information is divulged about the other nodes with which it has established sessions. Further, embedded operating systems for sensor networks (e.g. Contiki) usually offer no virtual memory facilities and impose restrictions that require pre-allocation of memory space, requiring that the maximum number of such associations be known a priori.

This paper presents a novel protocol for Adaptive Key Management (AKM) based on certificates, with the goal of limiting the number of secure associations per node to a predefined limit. Our protocol, is robust, secure and scalable under such limits. To the best of our knowledge, this is the first work that explicitly addresses the issue of memory limits for peer-to-peer link-layer security in WSN.

AKM works by exploiting topology information provided by the Routing Protocol for Low-Power and Lossy Networks [1] (RPL). RPL is the routing protocol of choice for the various applications loosely termed the “Internet of Things” (IoT), such as smart power grid and home automation, because of its ability to form optimal routes to the sink (a central collection point).

Following are the major contributions of this paper:

• We define a distributed, adaptive, online key management scheme, with the goals outlined above.
• We provide insights on the minimum number of security associations that need to be maintained by each node such that the entire network may be securely connected and reasonable routing performance be achieved.
• We show how the security association limit affects performance metrics such as the number of hops to the sink and the overall time for a network-wide secure routing topology establishment.

The remainder of this paper is organized as follows: in Section II, we present existing authentication and key management protocols for WSN and characterize the storage requirement for these protocols (Section III). The RPL routing protocol is briefly presented in (Section IV). We next detail our proposed scheme and comment on how it can be applied to a typical Smart Grid application (Section V). We present the simulation of our proposed scheme (Section VI) and analyze the output of this simulation (Section VII). Finally, we conclude the paper in Section VIII.

II. RELATED WORK

In Wireless Sensor Networks, the cost of asymmetric cryptography is often so prohibitive that symmetric cryptography based solutions are favored. This fact imposes a new paradigm: keying material needs to be pre-deployed on the nodes before the network starts. This is the main role of Key Management System (KMS). Basic forms of KMS, such as the probabilistic key sharing scheme proposed by Eschenauer and Gligor [2], allocates small random subsets of a bigger set of keys to nodes and relies on the probability that two neighboring nodes share a common key to securely communicate. More sophisticated KMS, such as PIKE [3] try to minimize the amount of keys required.

Some authors assume the keying material to be already
deployed and use it to derive additional keys. One such example is SPINS [4], where the authors define a framework for data confidentiality, authentication and integrity based on pre-deployed keys shared between the base station, acting as a Key Distribution Center (KDC), and the nodes. In SPINS, the base station is a trusted entity through which new keys are derived for node-to-node communications. In LEAP+ [5], the authors define a protocol that enables nodes to perform in-network processing (e.g. data is aggregated as it moves toward the sink) and supports multiple communications patterns (one-to-one, cluster, broadcast), each pattern being protected by a different key. In this scheme, the additional keys are derived from a pre-distributed key shared between the node and the base station.

When the topology of the network is not known in advance, efficiently pre-deploying keys might not be possible, and public key cryptography can not be avoided. TinyECC [6] shows that it is possible to use Elliptic Curve Cryptography (ECC) on sensor nodes. Hubaux et al. [7] describes self-organized Public Key Infrastructure (PKI) where nodes establish trust graphs.

More recently, due to the increase of the computational power of sensors and presence of Hardware Cryptographic Accelerator porting some ‘mainstream’ security protocol to the WSN realm has become possible. In EAP-Sens [8], the authors extend EAP and IEEE 802.1X to work on IEEE 802.15.4 enabled-radios. Oualha et al. [9] describes a solution based on the AAA-protocols PANA and RADIUS/DIAMETER, with a functioning similar to our scheme, where the authenticated nodes expends in rings outward from a central trusted source. However, in contrast to our approach, they do not constrain the security association space.

Among all the previously mentioned contributions, only LEAP+ [5] provides information on memory requirement. However, none of the cited works define the behavior of their solution when the node memory becomes saturated nor do they define how to manage and bound the space allocated to security associations. This is the major focus of our work.

In the next section, we motivate the need for our work by presenting the memory requirements of popular cryptographic primitives and protocols.

### III. The Case for AKM: Storage Considerations for Cryptographic Protocols

Sensor deployments may require unattended operation for decades. The cryptographic strength of the whole system must therefore be decided accordingly. To that end, NIST Special Publication 800-57 [10] provides guidelines for choosing the appropriate key length of cryptographic material being deployed. For example, if the keying material needs to be used until 2030 and beyond, a minimal cryptographic strength of 128 bits is recommended. In practice, this translates into the following requirements for the most popular primitives:

- AES, in its AES-128 variant, must be used
- RSA modulus size (n) must be at least 3072 bits.
- Elliptic Curve Cryptography key size (f) must be at least 256 bits.

In Table I, we present the storage requirements of the key. In other words, from the specification, we computed how much memory is be required to store the key without wrapping applied to the structure. This provides us a lower bound of the storage space required to store the keys. We computed the theoretical storage requirement of asymmetric cryptography encryption data, because this data can be cached between sessions. We compared these values against the size of the data structures defined in the popular sensor oriented Relic Cryptographic Library [11]. This shows how design choices influence real-world storage space requirements.

In Table II, we consider the storage space for an active session for LEAP+, SPINS and IEEE 802.1X [12].

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Theoretical Storage Req.</th>
<th>Relic Storage Req.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES key</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td>RSA-3072 public key</td>
<td>387</td>
<td>6288</td>
</tr>
<tr>
<td>RSA-3072 private key</td>
<td>~1728</td>
<td>6288</td>
</tr>
<tr>
<td>ECDSA-256 public key</td>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>ECDSA-256 private key</td>
<td>32</td>
<td>270</td>
</tr>
</tbody>
</table>

**TABLE I:** Cryptographic material storage requirement for some well-known cryptographic primitives (in bytes)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Required cryptographic storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAP+</td>
<td>1 pairwise key (16 bytes) + 2 encryption keys (32 bytes) + 1 replay counter (varying)</td>
</tr>
<tr>
<td>SPINS</td>
<td>1 pairwise key (16 bytes) + 2 MAC key (32 bytes) + 2 encryption keys (32 bytes) + 1 replay counter (varying)</td>
</tr>
<tr>
<td>IEEE 802.1X - 4-Way Handshake</td>
<td>PTK (64 bytes) + GTK (32 bytes)</td>
</tr>
</tbody>
</table>

**TABLE II:** Storage space required per ‘active’ session on some major security protocols (in bytes)

Based on the information presented in the tables above and considering the limited memory availability of popular WSN nodes, such as the RedBee Econotag (96kB of RAM) and the T-Mote Sky (10kB of RAM), we motivate the need for a protocol that can run effectively under memory constraints.

### IV. Routing Protocol for Low Power and Lossy Networks (RPL)

The Routing Protocol for Low Power and Lossy Networks [1] (RPL) works on IPv6 using ICMPv6 messages and is designed to operate on low data rate and lossy networks where nodes have a low processing power and limited memory. RPL is ideally suited for one-to-many and many-to-one communication patterns - a pattern that is typical for a broad class of WSN applications. Because our protocol retrieves topology information from RPL, we briefly present the salient points of RPL in this section.

RPL builds a Destination Oriented Directed Acyclic Graph (DODAG) that directs all the ‘upward’ traffic toward a single destination: the DODAG root. In most WSN scenarios, the sink is a sensible choice for the DODAG root. This is because the DODAG’s topology favors the point-to-multipoint communications originating from the root and multipoint-to-point communications destined to the root. Both schemes are optimal for command dissemination and data collection.

In RPL, upward routing is initiated by the DODAG root. The root broadcasts a DODAG Information Object (DIO) message to its neighbors, which in turn rebroadcast the DIO message to their neighbors, and so on, until the whole network is covered. To speed up the joining process of a new node, a node wishing to join the network can send a Destination Information Solicitation (DIS) in order to receive DIO messages from its neighbors.

---

1. When public exponent e is equal to \(2^{16} + 1\).

2. We assume that the original protocol is updated to use AES-128.

3. The DODAG root may have multiple IP interfaces and act as a gateway to a backbone network where collected data need to be sent.
neighbors. When rebroadcasting a DIO, the rank information within the message is altered so as to reflect the sender node’s own rank. This rank value provides an indication on a node’s position relative to other nodes and relative to the DODAG root, the lowest rank values indicating the closest nodes to the root. Because the rank strictly increases as the nodes are further apart from the root, a node can build up a list of one or more parent nodes - that is, nodes whose rank is less than its own. Later, when a node wants to send a packet upward (i.e. to the sink), it sends the packet to one of its parents which in turn forwards the packet to its own parent. This process is repeated until the packet reaches the root. The salient points to note are: (1) A given node only maintains information about its parents - not its children. (2) A RPL node maintains at least one parent in its parent list but can potentially keep more than one parent for routing redundancy.

Our protocol relies on information provided by RPL in order to choose a strategy for link replacement when needed as described in the following sections.

V. ADAPTIVE KEY MANAGEMENT

In this section, we describe the core operations that implement our scheme.

A. Assumptions

AKM is link layer agnostic. We make the following fairly general assumptions:

- Communication channels between nodes (i.e. ‘links’) are bidirectional.
- Each node of a given network is provisioned with a unique certificate belonging to a network-wide Public Key Infrastructure (PKI). However,
- The ‘sink’ (the RPL DODAG root) has significantly superior computational and memory capabilities when compared with the regular nodes.

B. Protocol overview

The main goal of AKM is to build a secure communication overlay encompassing as many nodes as possible on top of an existing WSN while limiting the number of security associations per node to a pre-defined limit. Initially, the only ‘authenticated’ node is the sink. A secure overlay expends outward from the sink in ‘rings’ until the whole network is covered by the overlay. Concurrently, a routing topology is established on top of this secure overlay.

Because the secure overlay network grows from the sink, there is always a secure path to it. This allows for any authenticator that joined the secure overlay to reach the sink at any time during the authentication procedure and provides a path for data so that the authenticator can retrieve information such as certificate revocation status and other certificate validation information.

We assume that mutual authentication is performed through the use of certificates. Abstractly, we model the authentication as a three-way handshake, which is a reasonable description of the messaging behavior of well known authentication methods such as TLS/EAP-TLS [13], [14].

As the network expands, each authenticated node can act as an authenticator to accommodate new nodes. In order to grow the network new nodes need to be added dynamically, while simultaneously maintaining the security association limit and the IP path to the DODAG root. To achieve this, existing authenticated links may need to be torn down and new links established with newly joining nodes as the network expands, while at the same time being aware of the local routing topology requirements to ensure that authenticators have a path to the sink. We define two operations to accomplish this: ‘redundant parent pruning’ and ‘node insertion’. Note that since each node has only a localized view of the graph (i.e. can only see its neighbors), this is an online algorithm and hence we do not aim for optimizing global metrics such as path length to the sink.

C. States of a Link and Node

As defined in IEEE 802.1X [12], a node can be an authenticator or supplicant. An authenticator is part of the secure overlay and is expected to participate in the routing protocol which implies it always has a secure route to the sink. A supplicant can be a new node that is not already part of the secure overlay that wants to join the routing topology and send messages to the sink or it can be a node that is already part of the secure overlay, in the neighborhood of an authenticator, that has not yet been authenticated by this authenticator (which is also a part of the secure overlay). As the latter case outlines, depending on the context, a node must be able to act both as a supplicant and as an authenticator.

A node can be in one of the following states:

- **Authenticated-unsaturated:** the node has enough memory to make new security associations.
- **Authenticated-saturated:** the node does not have enough storage to take on new security associations. Two sub-states exist for this state:
  - Redundant parent available: there is more than one parent in the DODAG.
  - No redundant parent available: the node has only one parent in the DODAG.
- **Unauthenticated:** the node has no authenticated link to the secure overlay.

Correspondingly, we have link states. As illustrated in Figure 1, a link’s state can take one of the following values **Authenticated Links**, **Unauthenticated Links** or **Pending Links**. Pending links are temporary associations and are exclusively used for attempting to accommodate new node when the current authenticator is ‘saturated’.

![Fig. 1: Link states: a link transitions between different states in the process of being inserted into the secure overlay.](image)

D. Authentication procedure

When a node is turned on, it is in the Unauthenticated state and starts beaconsing. A beacon message triggers an authenticator to send the newly joining node an authentication challenge. Authentication challenges are sent after a random delay, dependent on the node states defined above, as illustrated in Figure 2.
A beacon message contains the following information:

- **Status of the node**: Authenticated or Unauthenticated
- **Identifier** of the node sending the message.

Beaconing occurs at two different rates. Unauthenticated nodes transmit beacons at a regular interval, while Authenticated nodes transmit beacons at a substantially less frequency. Eventually, an authenticated node stops beaconing if it fills up all its security association storage space (i.e., it becomes saturated). When that occurs, the authenticator node will answer. Subsequent challenges from saturated nodes will simply be dropped by the newly joining node.

The abstracted authentication procedure is composed of 3 messages, exchanged in the following order:

1) **Authentication Challenge**: the authenticator sends a challenge (e.g., a cryptographic puzzle) to the supplicant; a challenge message provides an additional status message indicating the storage condition of authenticator. i.e., the authenticator indicates in the challenge if it is saturated, or has available space.

2) **Authentication Challenge Response**: the supplicant solves the challenge and responds.

3) **Authentication Acknowledgment**: the authenticator acknowledges the response and indicates the status in the acknowledgment. The status can be one of the following:

   - **SUCCESS**: authentication is successful, the link becomes an authenticated link. The supplicant becomes an authenticator, if it was not one already.
   - **PENDING**: authentication has succeeded, however, the authenticator cannot accommodate the supplicant. Additionally, the parent node of the Authenticator is provided in the Acknowledgment message. After this message has been received, the link becomes an authenticated link for AUTH_PENDING_TIMEOUT seconds. If the time period expires before the link is promoted to an authenticated link then the link expires, the authenticator destroys the association and returns the link to the UNAUTHENTICATED state.

The authentication procedure runs even when the authenticator is saturated. This is in order to allow the supplicant to possibly insert itself by one of the link pruning algorithms described below. In order to facilitate this, the acknowledgment may carry additional information such as the parent node ID of the authenticator that sends the acknowledgment (in case only a single parent exists).

### E. Storage Management

The security association storage is conceptually split in ‘slots’, each of which is able to store credentials for an authenticated link. For authenticator nodes, a slot belongs to one of the two following categories:

- **Permanent slot**: Permanently store cryptographic material associated to an authenticated link.
- **Temporary slot**: Temporary store cryptographic material associated to an authenticated link. A temporary slot can only be held for AUTH_PENDING_TIMEOUT seconds.

An authenticator divides its storage space in \( k \) permanent slots (with \( k \geq 2 \)) and one temporary slot, thus having \( k + 1 \) slots in total. The temporary slot is designed for authenticating joining nodes in the event all the permanent slots are taken, so that secure insertion and pruning algorithms (explained below) can be run. Thus an authenticator may establish a temporary link with only one supplicant at a time.

### F. Link pruning algorithm

Consider a new node wanting to join the secure overlay. After sending out a beacon, the newly joining node will receive challenges from one or more of its authenticated neighbors. The challenge presented to the newly joining node contains the state of the authenticator node. The newly joining node answers the challenge immediately. The timing delays for launching the challenge from the authenticators to the new node are set up so that the least expensive strategy is tried first. For example, if there is a node in the neighborhood of authenticated nodes that have available storage space, such nodes will launch a challenge to the newly joining node first, which the newly joining node will answer. Subsequent challenges from saturated nodes will simply be dropped by the newly joining node.

A difficulty arises when the only neighbors that the newly joining node has are all saturated. When that occurs, the newly joining node and its neighbors can collaborate to try to accommodate an authenticated link to the overlay without violating the security association limit storage constraint. Based on the local knowledge provided by RPL, we implement two strategies: **Redundant Parent Pruning** and **Node Insertion**.

1) **Redundant Parent Pruning**: The saturated (authenticator) node queries the RPL routing protocol in order to know if a backup parent exists. Note that RPL maintains a parent list, but only one parent is used at a time for upward routing from node to sink.
If such a parent exists, the saturated node can send its authentication challenge, indicating in that challenge that a redundant parent exists. If no such parent exists, the procedure fails and we fall back to the ‘Node Insertion’ procedure which is described below.

Upon reception of a valid challenge response message, the Authenticator queries the RPL routing protocol a second time to confirm that there is still at least one valid backup parent and asks for its identifier (an IPv6 address). If this condition is met, the Authentication Acknowledgment is sent to the supplicant with the status SUCCESS. Then, the saturated node tears down the authenticated link associated to its backup parent and moves the cryptographic material storage in the temporary slot to the permanent slot that has just been freed. If this condition is not met, the authentication terminates with the status PENDING and returns its (single) parent in the Authentication Acknowledgment. This triggers the ‘Node Insertion’ procedure which we now describe.

2) Node Insertion: Insert Node is an operation that ‘injects’ a joining node between two authenticators. It involves collaboration between a parent-child pair that are both authenticators and the supplicant that is trying to insert itself into the secure overlay. It becomes necessary to perform this operation when the only available option to join the secure overlay involves cutting the parent-child link and inserting the newly authenticated node in the middle. This situation occurs when a joining node has only parent child pairs in its neighborhood and both the parent and the child have only a single parent. In order to accomplish this, the joining node detects that two nodes in its secure overlay neighborhood have a parent-child relationship in the RPL routing tree. i.e. the joining node keeps track of the neighboring nodes with which it authenticates in order to detect their parent-child relationship. This can be done because an authenticator attaches its parent’s identifier to its acknowledgment during the authentication procedure. When a parent-child relationship has been detected, the joining node signals the child node to insert itself between the child and its parent. Because this operation affects the secure overlay topology, the RPL DODAG requires an update. This update can be sped up by having the child broadcast a DIO ‘poison’ message and getting the joining node to send DODAG Information Solicitation (DIS) upon joining the secure overlay. The former effectively cancels the route that goes through the child. The latter prompts the parent to send a DIO message so that the joining node can participate in the routing.

In a graph theoretic sense, we are adding and deleting edges to accommodate a new node into the secure overlay. We want to accomplish this while allowing existing nodes that are already part of routing DODAG to form a maximally connected graph subject to the key storage limit and be able to reach the sink, thereby providing maximal path redundancy to the sink. While we performed our analysis based on the information provided by RPL protocol, we note that other routing protocol could provide similar information on the network topology and a similar reasoning could be applied.

We now proceed to examine its performance using detailed simulation studies.

VI. SIMULATION

In this section, we apply AKM to Smart Grid application and analyze, through a high-level discrete event simulation, the performance of our proposed algorithm.

A. Connecting the Smart Meters to the Smart Grid

In urban areas, population density is such that Smart Meters are often in close range. In this specific setup, we believe that short range and low power wireless radio, such as IEEE 802.15.4, can enable Smart Meters to communicate with each others in order to form an Ad Hoc network.

We show how AKM enables secure operations in an Advanced Metering Infrastructure (AMI) oriented application, where a fleet of Smart Meters report meter reading to a collecting entity named Data Aggregation Point (DAP). In a typical deployment, a DAP is expected to collect reading from hundreds or even thousands of Smart Meters and then transfer this data to the utility company backbone network it is connected to. This means that the DAP actually acts as a sink to which Smart Meters send data to.

Unlike conventional WSN deployment, when it is acceptable for few nodes not to participate in the ‘main’ network operations, we expect all the Smart Meters to be connected. This is because Smart Meters report meter reading information that is crucial to the billing aspects of a utility company. Also, in order to ease the deployment process, nodes must be deployable without external assistance (i.e. without requiring human intervention).

B. Simulation setup

We extracted the population density from the US census data for two cities New York City (Manhattan island) and Baltimore, and extrapolated the data for simulation purposes as presented in Table III. We generated random graph topologies using NetworkX4. Edges between vertices, symbolizing that two nodes can talk to each others, are formed when the geographical distance between them is within their communication radius (or range). For the New York City simulation, we experimented with various coverage radii, 15, 30 and 50 meters. These small radius values aims to mimic capabilities of IEEE 802.15.4 radios that we observed through experimentation. For the Baltimore simulation, in order to evaluate the scalability of our approach, we chose to study a bigger area with more nodes connected to the DAP. We then convert the resulting graphs in network topologies for use within the OMNet++5 Discrete Event Simulator. Because no open-source RPL implementation was available, we wrote a minimalistic RPL module for OMNet++.

<table>
<thead>
<tr>
<th># of random networks</th>
<th>100</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>area of study</td>
<td>100m²</td>
<td>1000m²²</td>
</tr>
<tr>
<td>total # of nodes</td>
<td>135</td>
<td>1450</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nodes coverage radius</th>
<th>15m</th>
<th>30m</th>
<th>50m</th>
</tr>
</thead>
<tbody>
<tr>
<td>average # of neighbor</td>
<td>8.2</td>
<td>28.5</td>
<td>64</td>
</tr>
<tr>
<td>corresponding standard deviation</td>
<td>3.1</td>
<td>8.5</td>
<td>18.3</td>
</tr>
</tbody>
</table>

Table IV shows the simulation parameters for the simulation. We perform multiple runs of our algorithm on each networks so as to better characterize the effects of the random delays in various timers of our protocol.

VII. RESULT ANALYSIS

A careful analysis of the New York City and the Baltimore datasets showed that the Baltimore dataset, despite having a larger number of nodes, showed few differences when compared

4 http://networkx.lanl.gov/
5 http://omnetpp.org
with the New York City dataset. Hence, we focus mostly on analyzing the first dataset and only introduce elements of the second one when necessary. For the sake of comparison, we also introduce a ‘Simple’ algorithm. This Simple algorithm does not implement any link pruning strategy nor does it implement any beacon throttling. A node implementing the Simple algorithm will try to establish security association with its neighbors until its security association storage limit is reached. Similar to our approach, the Simple algorithm also expands outwards from the sink.

### A. Secure overlay coverage

The Figure 3 illustrates the performance of our proposal against the Simple algorithm under low storage constraints. In this figure, we compare the percentage of network being fully covered by the secure overlay. Here, we see that AKM offers good performance even with low association limit \((k)\) values under low connectivity.

![Figure 3: Comparison of the percentage of networks completely covered by the secure overlay between the Simple algorithm and AKM on the New York dataset.](image)

Note that even when nodes are provided a large amount of storage space (high \(k\) values), the Simple scheme still fails to connect a few nodes, which is not acceptable when deployment should be done without external assistance.

### B. Coverage time

Figure 4 shows the average time until the whole network is covered by the secure overlay\(^6\). As we can see, increasing the \(k\) value greatly decreases the time it takes for the whole network to be covered. This number is important in case the secure overlay needs to be rebuilt from scratch (e.g. after a power outage). We note that \(k\) value over 7 does not yield any significant changes in the covering times, thus indicating that keeping 7 security associations in memory suffices. Additional associations do not provide significant benefits from the perspective of this metric.

![Figure 4: Average time elapsed for the whole network included in the secure overlay](image)

### C. Effectiveness of link pruning algorithms

In Figure 5, we show how often the two link pruning algorithm are run on average on a network. The Baltimore dataset provides a larger number of nodes, thus, more nodes need to be connected to the secure overlay.

![Figure 5: Average number of runs per networks for each links pruning algorithm on the Baltimore dataset](image)

We note that the ‘redundant parent pruning’ algorithm runs less often on small \(k\) values. This is because security association is restricted. Thus a node establishes fewer parents in the RPL DODAG.

The steadily decreasing number of runs of the ‘node insertion’ algorithm as the number of ‘redundant parent pruning’ runs increase confirms the proper functioning our beacon mechanism scheduling.

### D. Number of hops to the sink

In Figure 6, we compare the average path lengths obtained on the RPL DODAG with the unconstrained shortest path on the network topology (i.e. without being constrained by the level of security offered by the path). For each nodes that is part of the secure overlay, we measure the length of the paths (i.e. the DODAG path and the shortest path) to the DAP.

We note that our emphasis is not on reducing the path length to the DAP but on covering all nodes. However, we wanted to ensure that our algorithm does not do badly in path length.

![Figure 6: Average number of hops to the sink on the Baltimore dataset](image)

For low \(k\) values, in Figure 6 the shortest path metric appears small because less nodes are connected, thus most of the nodes

---

\(^6\)The figure does not show any point for \(k = 2\) on the Baltimore dataset because the secure overlay was never completely covered.
are closer to the ‘sink’. For these low \( k \) values, AKM behaves worse than the Simple algorithm. This results from two facts: AKM can insert nodes in the middle when required. This effectively increase the path length by one hop. Also, the Simple algorithm connects less nodes, which are located closer to the DAP.

As the \( k \) value is increased, the DODAG path becomes closer to the shortest path (which is one of the features of the RPL routing protocol).

\[\text{Fig. 6: Comparison between the average path length from the nodes to the DAP on the RPL DODAG and the corresponding shortest path on the physical network on the New York City dataset.}\]

\[\text{Fig. 7: Comparison of the percentage of networks completely secured by the secure overlay between the Simple algorithm and AKM on the New York dataset when nodes joining time varies.}\]

\section*{E. Adding new nodes once the network is saturated}

Another interesting scenario occurs when nodes join the network at different times. When this happens, nodes that have been started earlier are likely to be already saturated when new nodes join. In Figure 7, we study a scenario where Smart Meters are started at a random time after a power outage and try to re-establish a secure network. In such a scenario, we can see that the Simple algorithm does not always make it possible for the new nodes to join. As expected, AKM outperforms the Simple algorithm and behaves well, even for low \( k \) values and low network density. This is because, by design, AKM can prune existing authenticated links in order to accommodate new joining nodes. This demonstrates AKM’s robustness against changing topologies.

\section*{VIII. Conclusion and Future Work}

In this paper, we presented a memory and routing topology aware authentication and key management protocol for wireless sensor networks. We demonstrated through extensive simulation, that our scheme works well under memory constraints on a wide range of network topologies. We showed that our proposal provides the flexibility required when the network topology changes (e.g. nodes join the network at random times).

Our simulation results show that our scheme is scalable and offers a better coverage than the Simple scheme which does not obtain topology information from the routing layer.

The choice of maximum number of security associations is dependent upon the density of the network. For a representative metropolitan area with a radio coverage of 15 meters, we concluded that 7 slots sufficed to give full coverage in all cases.

While it can be argued that memory is plentiful in general purpose processors, it is still at a premium in Wireless Sensor nodes especially when tamper proof components such as Hardware Security Modules are incorporated. Our results are particularly relevant for these scenarios.

Based on our encouraging simulation results, we are in the process of implementing this scheme on real IEEE 802.15.4 enabled devices.

\section*{Acknowledgment}

This work was supported by the Secure Smart Grid project at NIST. The authors thank Nada Golmie, manager of the Emerging and Mobile Networking Technologies for her input and support.

\section*{References}


