A Network Coding Approach to Secret Key Distribution

Paulo F. Oliveira  
João Barros

Instituto de Telecomunicações,
Departamento de Ciência de Computadores,
Faculdade de Ciências da Universidade do Porto,
Rua do Campo Alegre, 1021/1055, 4169-007 Porto, Portugal
Email:{pvf, barros}@dcc.fc.up.pt
Phone: +351 220 402 917
Fax: +351 220 402 950

Abstract

We consider the problem of secret key distribution in a sensor network with multiple scattered sensor nodes and a mobile device that can be used to bootstrap the network. Our main contribution is a set of secure protocols that rely on simple network coding operations to provide a robust and low-complexity solution for sharing secret keys among sensor nodes, including pairwise keys, cluster keys, key revocation and mobile node authentication. In spite of its role as a key enabler for this approach, the mobile node only has access to an encrypted version of the keys, providing information-theoretic security with respect to attacks focused on the mobile node. Our results include performance evaluation in terms of security metrics and a detailed analysis of resource utilization. The basic scheme was implemented and tested in a real-life sensor network testbed. We deem this class of network coding protocols to be particularly well suited for highly constrained dynamic systems such as wireless sensor networks.

Index Terms

secret key distribution, sensor networks, network coding

Parts of this work have been presented in the International Conference on Security and Cryptography (SECRIPT’07) [1] and in the International Symposium on Information Security (IS’07) [2]. This work was partly supported by the Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) under grants POSC/EIA/62199/2004 and PTDC/EIA/71362/2006, as well as by the European Commission under grant FP7–INFSO–ICT–215252 (N-Crave Project).
I. INTRODUCTION

Highly volatile and constrained systems such as wireless sensor networks, in which low-power, low-complexity nodes coordinate their efforts to collect and transmit physical data to a central collection point, as illustrated in Fig. 1, are by now widely perceived as particularly challenging with respect to secret key distribution.

Currently available proposals can be divided into at least three basic types of secret key distribution schemes [3]: (a) public-key infrastructure, (b) trusted third party and (c) key pre-distribution.

In spite of the fact that public-key infrastructure schemes have been implemented in a few sensor network prototypes [4], it can be argued that the requirements of these schemes in terms of processing and communication often exceed the resources available for large classes of wireless sensor networks. In the case of trusted party schemes, we must rely on a central base station to provide secret keys encrypted individually for each sensor node [5], thus inheriting all the drawbacks of having a single point of attack.

An alternative solution is to use key pre-distribution schemes, such that prior to deployment each node is loaded with a key ring of $k$ keys, chosen randomly from a random pool $P$, as proposed in [6]. A secure link is said to exist between two neighboring sensor nodes, if they share a key with which communication may be initiated. A random graph analysis in [6] shows that shared-key connectivity can be achieved almost surely, provided that each sensor node is loaded with 250 keys drawn out of a pool of roughly 100.000 sequences. A different scheme with pre-installed key rings is described in [7], in which the network key is erased immediately after the pairwise keys are established. Since nodes in that situation can no longer establish pairwise keys, the protocol in [7] is only suitable for static WSNs.

Secret key distribution schemes that exploit specifically the availability of mobile nodes thus far seem elusive. To the best of our knowledge the same could be said about the use of network coding [8], [9] (or equivalently, algebraic mixing of data packets) towards accomplishing secret key distribution tasks. The concept of weakly network coding security is introduced in [10] to describe the level of secrecy that is achievable for message transmission in a multi-cast scenario where an attacker only observes linear combinations of data packets and not the data packets themselves. Our contribution differs from the work in [10] in several important aspects: (a)
A wireless sensor network is a collection of small devices that, once deployed on a target area, organize themselves in an ad-hoc network, collect measurements of a physical process and transmit the data over the wireless medium to a data fusion center for further processing. If a mobile node (represented here by a laptop computer) is available, then it can be used to establish secure links between sensor nodes and thus bootstrap the network.

In the spirit of the Resurrecting Duckling paradigm in ubiquitous computing [11], [12], we consider the scenario in which a mobile node, e.g. a handheld device or a laptop computer, is available for bootstrapping the network and is used to help establish secure connections between the sensor nodes by exploiting the broadcast property of the wireless medium. We shall show that by applying the basic principles of network coding, as illustrated in Fig. 2, it is possible to design power-efficient key distribution schemes that are not probabilistic, while ensuring that the aforementioned mobile node does not constitute a single point of attack — its capture alone is not sufficient to compromise the whole network. In contrast with pure key pre-distribution schemes, we propose the combined use of network coding and mobility and show how these
Fig. 2. A typical wireless network coding example. To exchange messages \( a \) and \( b \), nodes \( A \) and \( B \) must route their packets through node \( S \). Clearly, the traditional scheme in Fig. (a) would require four transmissions. However, if \( S \) is allowed to perform network coding with simple XOR operations, \( a \oplus b \) can be sent, as shown in Fig. (b), in one single broadcast transmission (instead one transmission with \( b \) followed by another one with \( a \)). By combining the received data with the stored message, \( A \) which possesses \( a \) can recover \( b \) and \( B \) can recover \( a \) using \( b \). Thus, network coding saves one transmission.

Tools can be used effectively to establish secure connections between sensor nodes, offering the following non-trivial advantages:

- **Deterministic Security**: The combined use of network coding and a mobile node ensures that links are secured with probability one;

- **Global Efficiency**: in addition to a small number of transmissions and low-complexity processing (mainly XOR operations), each node is required to pre-store only a small number of keys (as many as its expected number of links);

- **“Blind” Key Distribution**: although the mobile node only sees encrypted versions of the secret keys, it is capable of using network coding to ensure that each pair of sensor nodes receives enough data to agree on a pair of secret keys.

In addition to the basic secret key distribution scheme, we also include a collection of relevant extensions, specifically:

- **Key Renewal for Robustness and Scalability in Dynamic Environments**: if the network topology changes rapidly or new nodes enter the network, new keys can be safely distributed
with a simple procedure even when the sets of pre-stored keys have been depleted;

- **Authentication, Clustering and Key Revocation**: we provide additional protocols that cover authentication of the mobile node, generation of cluster keys and revocation in the case of compromised sensor nodes;

- **Performance Evaluation**: we provide a thorough analysis of the security performance of our scheme by discussing its behavior under various attack models and proving mathematically that the encrypted keys stored by the mobile node are information-theoretically secure;

- **Implementation and Testing**: as a proof-of-concept we implemented and tested the basic XOR based key distribution scheme on TelosB motes, running the TinyOS 2.0 operating system.

The rest of the paper is organized as follows. Section II provides a detailed description of the secret key distribution scheme. The basic methodology is scrutinized in Section III, which explains all the features and extensions, such as requesting extra keys, generating cluster keys and revoking compromised keys. Section IV then elaborates on the consequences of having compromised sensor nodes and proves that the mobile node is indeed ignorant about the pre-stored keys. After some notes on a practical implementation described in Section V, the paper concludes with Section VI.

### II. Mobile Secret Key Distribution

In this section we describe a key distribution scheme that exploits the existence of a mobile node, requires only a small number of pre-stored keys, and provides a level of security similar to probabilistic private key-sharing schemes such as [6], [3], while ensuring that shared-key connectivity is established with probability one. Table I shows a comparison of the advantages and disadvantages of our scheme in comparison with various existing alternatives.

#### A. A Basic Key Distribution Scheme

For clarity, we start by describing the key distribution scheme in the case of two nodes. Later, in Section II-B, we will show that with a few simple extensions the same method can be used for a larger number of sensor nodes. Suppose that two sensor nodes $A$ and $B$ want to establish a secure link via a mobile node $S$. Although $A$ and $B$ own different keys that are unknown to $S$, the latter is capable of providing $A$ and $B$ with enough information for them to recover
TABLE I

COMPARISON OF SECRET KEY DISTRIBUTION SCHEMES

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic key-predistribution [6]</td>
<td>Low computational complexity.</td>
<td>Heavy memory requirements and extra communication to identify shared secrets. Links are not secured with probability one.</td>
</tr>
<tr>
<td>Trusted third party [5]</td>
<td>Low computational complexity.</td>
<td>Requires a super-node, which can be viewed as a single point of attack.</td>
</tr>
<tr>
<td>Proposed Network Coding Protocols</td>
<td>Low requirements in terms of memory, processing and communication. Provides means of authenticating the mobile node. Offers extra line of defense, because mobile node does not know the keys. Links are secured with probability one.</td>
<td>Requires mobile node for bootstrapping security and tamper-resistant sensor nodes.</td>
</tr>
</tbody>
</table>

each other’s keys based on their own pre-stored keys. The basic scheme, which is illustrated in Fig. 3, can be summarized in the following tasks:

(i) Prior to sensor node deployment:

– Generate a large pool \( \mathcal{P} \) of statistically independent keys \( K_i \) and their identifiers \( i \in \{0, ..., |\mathcal{P}| - 1\} \);

– Produce a Vernam cipher [13] \( R \), i.e. a binary sequence of size equal to the key size and consisting of bits drawn randomly according to a Bernoulli \( \left( \frac{1}{2} \right) \) distribution;

– Store in the memory of \( S \) a list with all identifiers \( i \) and an encrypted version of the corresponding key \( K_i \oplus R \) (it shall be proven in Section IV that in this case it is perfectly safe to use the same Vernam cipher \( R \) for all the keys, specifically because they are drawn uniformly at random);

– Let \( C \ll |\mathcal{P}| \) be the expected number of links that each node intends to use during its lifetime (it can be increased after deployment); load \( C \) keys from \( \mathcal{P} \) and their
corresponding identifiers into the memory of each sensor (the subsets of keys possessed by the different sensor nodes are disjoint).

(ii) After sensor node deployment:

1) $S$ broadcasts HELLO messages, once it reaches the target area;
2) Each sensor node within the wireless transmission range of $S$ replies by sending one key identifier;
3) Upon receiving the identifiers $i(A)$ from node $A$ and $i(B)$ from node $B$, the mobile node $S$ performs a simple table look-up and runs an XOR network coding operation over the corresponding protected keys, i.e. $K_{i(A)} \oplus R \oplus K_{i(B)} \oplus R$. Since $R$ cancels out, $S$ sends back $K_{i(A)} \oplus K_{i(B)}$;
4) Based on the received XOR combination $K_{i(A)} \oplus K_{i(B)}$, $A$ and $B$ can easily recover each other’s key ($A$ knows $K_{i(A)}$ and computes $K_{i(A)} \oplus K_{i(A)} \oplus K_{i(B)}$, thus obtaining $K_{i(B)}$; and $B$ proceeds similarly).

Once this process is concluded, sensor nodes $A$ and $B$ can communicate using the two keys $K_{i(A)}$ and $K_{i(B)}$. $E_{K_{i(A)}}(m_{A \rightarrow B})$ denotes a message sent by $A$ to $B$, encrypted with $K_{i(A)}$, and $E_{K_{i(B)}}(m_{B \rightarrow A})$ corresponds to a message sent by $B$ to $A$, encrypted with $K_{i(B)}$.

B. Large-Scale Secret Key Distribution

We are now ready to describe how this scheme can be used in large-scale sensor networks. Fig. 4 highlights the required modifications. For simplicity, each global key identifier $i$ is assumed to result from the concatenation of the node identifier $n$ and the local key identifier $j$ (e.g. $|n| = 24$ bit and $|j| = 8$ bit). Each sensor node knows both its own identifier $n$ and the local key identifiers $j$ (substituting the key identifiers $i$ used in the simplified two-node scheme presented in the previous Section). The general protocol can be described as follows:

1) The sensor nodes perform standard neighborhood discovery by broadcasting their identifiers $n$ and storing in a list $L_n$ the identifiers announced by their neighbors;
2) $S$ broadcasts HELLO messages that are received by any sensor node within wireless transmission range. Each sensor node sends a reply message containing \{n, $L_n$\};
3) When $S$ receives \{n(A), $L_{n(A)}$\} from a node $A$ and \{n(B), $L_{n(B)}$\} from a node $B$, it checks whether $n(A) \in L_{n(B)}$ and $n(B) \in L_{n(A)}$, $n(A) \neq n(B)$. If this is the case,
Fig. 3. Secret key distribution scheme. Sensor nodes $A$ and $B$ want to exchange two keys via a mobile node $S$. The process is initiated by a HELLO message broadcasted by $S$. Upon receiving this message, each sensor node sends back a key identifier $i(\cdot)$ corresponding to one of its keys $K_i(\cdot)$. Node $S$ then broadcasts the result of the XOR of the two keys $K_i(A) \oplus K_i(B)$. After recovering each other’s keys by simple XOR operations, the nodes communicate securely by encrypting their messages.

$S$ sends back \{\(n(A) \ast j(A), n(B) \ast j(B), K_{n(A)\ast j(A)} \oplus K_{n(B)\ast j(B)}\}\), where \((n(\cdot) \ast j(\cdot))\) denotes the concatenation of node and local key identifiers; the local key identifier $j$ (for each node) is initially set at 0 and increases with the number of established links;

4) Upon receiving this message, $A$ and $B$ can recover each other’s keys by performing an XOR operation using the lowest local key identifier that corresponds to an unused key.

Thus, each pair of nodes shares a pair of keys which is kept secret from $S$.

C. Usage of Keys

There are several ways to make use of the established pair of keys beyond the straightforward solution in which each node encrypts messages with its own pre-stored key. One possibility is to encrypt the messages using both keys in a double cypher albeit at the cost of non-negligible computational overhead. A less onerous alternative is to form a common session key as a function of the two shared keys and use that session key to secure the communication. Session keys can also be generated locally, for example node $A$ generates a random value $x$, encrypts it using
Fig. 4. Example of the general key distribution scheme for the topology shown above. Sensor nodes A, B, C and D want to exchange keys with their neighbors via a mobile node S. Although we use a line network in this example, the scheme is suitable for any topology. Initially, the nodes exchange their identifiers and wait for an HELLO message from S (transmission 1). After this step, each node sends a key request message to the mobile node (transmissions 2,3,5,7) and waits for the latter to send back a key reply message (transmissions 4,6,8).

one of the shared keys and sends it to node B, which generates a random value y and sends it back to A, encrypted with the other key). Naturally, the availability of suitable random number generators is a relevant issue to be taken under consideration. One possible approach to fulfill this gap could be using the pseudo-random generator presented in [14], which is particularly well suited for wireless sensors.

It is important to point out that sharing keys a and b through our protocol does not allow the nodes to double the key size by concatenating the two keys (or the two session keys). The main
Fig. 5. Sensor node $A$ wants to authenticate the mobile node $S$. To accomplish this task, $A$ sends to $S$ its identifier and an even number of local key identifiers that it possesses; after receiving this information, $S$ sends back the local key identifiers announced by $A$, encrypted with the XOR of the corresponding keys.

reason is that the eavesdropper can overhear $a \oplus b$ over the channel and thus only has to guess one of the keys to recover the other one.

III. OTHER FEATURES AND EXTENSIONS

A. Authentication of Mobile Node

The single most essential feature of the mobile node is that it can generate all the XOR combinations of pairs of keys. This feature unveils a simple way for a sensor node to check the legitimacy of the mobile node (see Fig. 5):

1) the sensor node transmits an even number$^1$ of key identifiers;

2) the mobile node sends back the key identifiers encrypted by the XOR of the corresponding keys (instead of sending back a message containing the same key identifiers, the mobile node can answer in the challenge-response mechanism with the result of an operation, e.g. the addition of the used local key identifiers);

3) the sensor node verifies the authenticity by decrypting and comparing the key identifiers.

Similarly, the mobile node and an arbitrary sensor node can agree to encrypt their messages using as key the XOR of the even number of keys indicated by the sensor node, thus ensuring confidentiality.

B. Request for Extra Keys

Since each sensor node is initialized with only a limited number of keys, a situation could occur in which secure links must be established although all the keys have already been depleted.

$^1$An even number of keys ensures that the sequence $R$ disappears after the XOR operation carried out by the mobile node.
Fig. 6. Sensor node $A$ wants to obtain more keys. For this purpose $A$ sends the mobile node its identifier, along with an even number of local key identifiers; after receiving this information, the mobile node sends back data encrypted with the XOR of the keys announced by the node; the provided information contains the chosen local key identifier that the sensor node possesses, a new valid key identifier and the XOR of the corresponding keys.

One way to solve this problem, is for the sensor node to request the mobile node for more keys using the following protocol (see Fig. 6):

1) the sensor node transmits an even number of key identifiers and requests a new key;
2) the mobile node sends back a packet with two key identifiers and the XOR of the corresponding keys; the first key identifier corresponds to a key already available to the sensor node, whereas the second one refers to a new key; for security this packet is encrypted using the XOR of the even number of keys proposed by the sensor node;
3) the sensor node decrypts the packet and recovers the new key by performing an XOR operation with the already known key, identified in the received packet.

Since the messages are secure against eavesdropping (even from a legitimate node that shares a key with the requesting node), the latter can securely repeat this process several times up to the total number of admissible local key identifiers. In order to have enough space to save new keys, the requesting node can delete obsolete keys from its memory.

C. Cluster Keys

Beyond the secret keys that are shared by pairs of nodes, it is often useful to have special keys shared between $k$ nodes in the same cluster. This can be achieved by having nodes exchange their identifiers, build a list of nodes that want to share a cluster key and agree on a common cluster identifier. Once the mobile node learns which nodes want to share a cluster key, it broadcasts $k-1$ independent pairwise XOR combinations of keys for each sensor node to be able to recover one key for each one of the other nodes in the cluster — all it has to do is to solve the resulting linear system of equations. Consider the example shown in Fig. 7, where it can be seen that
upon receiving message 6 node $A$ cannot recover any key. However, after receiving message 7, node $A$ can compute the key owned by $B$ and consequently recover the key possessed by $C$. Furthermore, upon receiving message 8 node $A$ can also compute the key owned by $D$. All the $k$ nodes proceed similarly until they have $k$ keys in common from which they can compute the cluster key. It can be shown that an eavesdropper will not have enough degrees of freedom to solve the linear system and recover the keys, since $k-1$ independent pairwise XOR combinations of $k$ keys are not sufficient to recover any key.

It is fair to say that this approach for generating cluster keys is effective but not necessarily efficient: $k-1$ transmissions are required from the mobile node for $k$ nodes to be able to communicate with the same cluster key. We leave such improvements for future work.

D. Revocation

When a node is captured, it must be possible to revoke its entire key ring. Once the mobile node knows the identifiers of compromised nodes or keys, these can be neutralized in the following manner. First, the mobile node broadcasts a revocation message with the list of node or key identifiers to be revoked. Secondly, the compromised nodes or keys are deleted from the mobile node’s memory to prevent the use of exposed keys. Upon receiving the key revocation message, the sensor node verifies if it has any of the revoked keys in its memory. Similarly, upon receiving a node revocation message, a sensor node verifies if it is connected to any of the compromised nodes. After verifying the authenticity of the mobile node (e.g. through the authentication process described in Section III-A), the warned sensor node blocks all the connections initiated by the exposed nodes or keys and removes them from its key ring. Revocation affects only the links currently connecting the compromised node to other nodes, thus isolating it from the network.

IV. Performance Evaluation

To determine the security level of the presented solutions, we shall analyze their main vulnerabilities, identify whether the mobile node is a single point of attack, and determine the general requirements in terms of memory.

A. Attacker Model

We consider the threat posed by an attacker with the following characteristics:
Fig. 7. Cluster key scheme. Sensor nodes A, B, C and D are connected to each other and want to form a cluster key via a mobile node S. The required exchange of messages is similar to the one in Fig. 4 with the difference that here the nodes also agree on a cluster identifier G. Thus, when the nodes receive an HELLO message from the mobile node (transmission 1), they send back a key request message that includes the cluster identifier (transmissions 2,3,4,5), their own identifiers and the identifiers of the nodes which were announced to them. After the mobile node receives this information, it broadcasts a chain of reply messages (transmissions 6,7,8). Each of these messages contains the cluster identifier G, the identifiers of two nodes and XOR results of the keys they own. The value of j is not incremented, in other words the key corresponding to a particular node remains the same for all XOR operations carried out by the mobile node.

<table>
<thead>
<tr>
<th>msg</th>
<th>sender</th>
<th>receiver</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>A,B,C,D</td>
<td>HELLO</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>S</td>
<td>{G, n(B), [n(A), n(C), n(D)]}</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>S</td>
<td>{G, n(C), [n(A), n(B), n(D)]}</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>S</td>
<td>{G, n(A), [n(B), n(C), n(D)]}</td>
</tr>
<tr>
<td>5</td>
<td>D</td>
<td>S</td>
<td>{G, n(D), [n(A), n(B), n(C)]}</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>A,B,C,D</td>
<td>{G, n(B) * j(B), n(C) * j(C), K_{n(B)*j(B)} \oplus K_{n(C)*j(C)}}</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>A,B,C,D</td>
<td>{G, n(A) * j(A), n(B) * j(B), K_{n(A)*j(A)} \oplus K_{n(B)*j(B)}}</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>A,B,C,D</td>
<td>{G, n(C) * j(C), n(D) * j(D), K_{n(C)*j(C)} \oplus K_{n(D)*j(D)}}</td>
</tr>
</tbody>
</table>
1) the attacker can listen to all the traffic over the wireless medium;
2) the attacker is able to inject bogus traffic in the network;
3) the attacker can gain access to the memory of the mobile node or to the memory of a
limited number of sensor nodes (but not to both);
4) the attacker is computationally bounded (polynomial in the security parameter) and thus
unable to break hard cryptographic primitives.

We are now ready to comment on each one of these characteristics. The first characteristic
grants that attacker access to all communications. Although this is the strongest possible
eavesdropping assumption in a wireless network [15], the threat to our schemes is negligible
because the shared keys cannot be decoded from the XOR messages transmitted through the
wireless medium. Fake transmissions resulting from characteristic 2, which typically amount to a
Bizantine modification attack [16], can be detected by the legitimate nodes, who may choose to
ignore any messages that are corrupted by an invalid key. Characteristic 3 amounts to a physical
attack similar to the ones considered in [6]. The implications to our protocol of the threat posed
by an attacker who gains access to the memories of both the mobile node and at least one sensor
node shall be discussed in Section IV-C. Finally, the limitation on the computational power of
the attacker of characteristic 4 is the typical assumption that allows sensor nodes to use standard
cryptographic primitives to cipher the sent messages and achieve computational security.

B. Impact of Compromised Sensor Nodes

It is frequently assumed that the nodes are equipped with tamper-detection technology and
sensor node shielding that erases the keys when nodes are captured [12]. If this assumption
cannot be met, it is still possible to evaluate the impact of compromised sensor nodes. Starting
with the trivial case of a single key for the entire $N$-node network, if one node is compromised
then all communication links are insecure. On the other hand, end-to-end schemes that establish
pair-wise key sharing between any two nodes confine the impact to the subset of $N - 1$ links
of the compromised node. The probabilistic scheme presented in [6] assumes that $k \ll |P|$ 
keys from a single ring are stolen, which gives the attacker a probability of $\frac{k}{|P|}$ of successfully
attacking any link. In our scheme, each node initially possesses $C < k$ keys. By exploiting the
protocol, an attacker who succeeds in compromising one node is able to acquire at most $C^*$ new
keys. Thus, assuming that $C^*$ is very limited (as explained in Section III-B, $C + C^*$ is at most
equal to the total number of local key identifiers), an attacker will not be able to expose more
than \(2(C + C^*) \approx k \ll |\mathcal{P}|\) pairwise keys (for each key available, the node may get a key from
the key ring of its neighbors). Since one pair of keys is used to secure one link, the number
of links that can be compromised is \(C + C^*\). Similarly, for the case of cluster keys, although
the attacker can obtain \(m\) keys per cluster (where \(m\) is the number of nodes of each cluster), it
cannot compromise more than \(C + C^*\) links to clusters (\(m\) keys are used to secure the links of
each cluster). Given the fact that the secret keys are statistically independent, the attacker cannot
compromise other links except by a brute-force attack.

C. Impact of a Captured Mobile Node

The mobile node in our system could be perceived as a single point of attack. If it is captured
prior to key activation, the attack will be trivially detected because the sensor nodes are not able
to communicate. In this case, the attacker will nevertheless have access to a table containing the
XOR results of every key with \(R\). Fortunately, as proven in Section IV-D, the attacker cannot
recover any of the keys \(K_i\) — its options are once again limited to brute-force attack.

A successful attack requires the adversary not only to gain access to the memory of the mobile
node but also to know either one identifier \(i\) and the corresponding key \(K_i\) or the encryption
sequence \(R\). In this case, the adversary has sufficient information to decode all the keys used by
the network. It is thus fair to state that the exposed memory of one regular sensor node together
with the data present on the memory of the mobile node allows the attacker to know all the
keys. However, to increase the difficulty of an attack on the mobile node, physically unclonable
functions (PUFs) [17] can be used to protect the cryptographic data, even in the event that the
attacker gains physical access to the hardware.

D. Information-Theoretic Security

The keys stored in the mobile node are protected by a Vernam cipher. It is well-known that the
Vernam cipher can be proven to achieve information-theoretic security (or perfect secrecy, [18])
for any message statistics if the key is (a) truly random, (b) never reused and (c) kept secret. In
our case, the messages correspond to keys drawn from a uniform distribution and, consequently,
the requirement that the Vernam cipher is never re-used can be dropped, as stated in the following
theorem (the full proof of which can be found in the appendix).
**Theorem 1:** The knowledge of \( \{ K_1 \oplus R, K_2 \oplus R, \ldots, K_m \oplus R \} \) does not increase the information that the attacker has about any key, i.e., \( \forall i \in \{1, \ldots, m\} \),

\[
P(K_i = x | K_1 \oplus R = y_1, \ldots, K_m \oplus R = y_m) = P(K_i = x).
\]

**Sketch of proof** First, notice that \( P(K_i = x) = \frac{1}{2^n} \). We shall prove that \( P(K_i = x | K_1 \oplus R = y_1, \ldots, K_m \oplus R = y_m) = \frac{1}{2^n}, \forall i \in \{1, \ldots, m\} \), which yields the result.

\[
P(K_i = x | K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m)
= \prod_{j=1}^m P(K_j = z_j \oplus x) = \frac{1}{2^{n(m-1)}}
\]

where the event \( K_i \oplus K_i = y_i \oplus y_i \) is not present, because it is redundant. Let \( z_j = y_j \oplus y_i \), for \( 1 \leq j \leq m \) and \( j \neq i \). Let \( A \) denote the event \( \{ K_i = x \} \) and \( B \) denote the event \( \{ K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m \} \). Then, \( P(K_i = x | K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m) = P(A | B) \).

We already have seen that \( P(A) = P(K_i = x) = 1/2^n \). We have that:

\[
P(B | A) = P(K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m | K_i = x)
= \prod_{j=1}^m P(K_j = z_j \oplus x) = \frac{1}{2^{n(m-1)}}
\]

\[
P(B) = P(K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = Z_m)
= \prod_{j=1}^m P(K_j \oplus K_i = z_j) = \frac{1}{2^{n(m-1)}}
\]

Therefore, we have that \( P(A | B) = \frac{1}{2^{n(m-1)}} \).

**E. Exposed Information to an Eavesdropper/Active Attacker**

The only kind of information that passes through the wireless medium are HELLO messages, identifiers and XORs of keys. HELLO messages clearly do not compromise the network security, therefore the sole information that an eavesdropper can acquire are pairs of identifiers and the result of XOR operations on the corresponding keys (belonging to different nodes). An active attacker that is not a legitimate node is able to obtain the XOR of any two keys (each one of them belonging to different nodes) simply by exploiting the basic protocol. However, it cannot get XORs of keys of the same node (which are used in the authentication of the mobile node and in the request for extra keys) nor can it get XORs of an odd number of keys (which would
invalidate the whole protocol if the mobile node is captured). Given the fact that the attacker has to guess at least one random key to recover the sent keys, the protocols can be deemed secure against eavesdropping/active requesting. Moreover, even if the attacker discovers some keys, it could only expose at most one connection per discovered key: the combined information available through the wireless medium and contained in an exposed key does not leak any additional information pertaining keys used in other links.

\section*{F. Brute-Force Attack Analysis}

A well-known physical argument states that a 256 bit symmetric key is secure against a brute force attack \cite{19}. On the other hand, it is proven in \cite{20} that a brute force attack on a sensor node using a 40 bit key will succeed on the average after 128 years, which is beyond the expected lifetime of current sensor nodes \cite{20}. We conclude that the result of a brute force attack directly on the sensor nodes is restricted to a Denial of Service attack and if a large enough key is used, an offline attack over captured messages will be useless.

\section*{G. Memory Requirements}

We recall that each node $n$ has $C$ keys $K_i$ in memory, each one identified by $|i| = |n| + |j|$ bits, where $| \cdot |$ denotes the size of the argument. To store the protocol data, each node requires

$$|n| + C \times (|j| + |K_i|)$$

bits of memory space and the mobile node needs

$$2^{|i|} \times (|i| + |K_i|) = |P| \times (\lceil \log_2(|P|) \rceil + |K_i|)$$

bits. For example, if we assign $n = 24$ there is space for 16,777,216 different node identifiers. For $j = 8$, each sensor node can obtain 256 keys (e.g. if each node initially has $C = 20$ keys in its memory, there is space for 236 extra keys to be requested from the mobile node). Table II illustrates the required resources, which we deem very reasonable under current technology.
### TABLE II

**Required memory for each sensor node (SN) and required memory for the mobile node (MN), for fixed values of** $n = 24$, $j = 8$ **and** $C = 20$.

<table>
<thead>
<tr>
<th>Key size</th>
<th>Size on SN</th>
<th>Size on MN</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 bit</td>
<td>663 Bytes</td>
<td>144.0 GB</td>
</tr>
<tr>
<td>128 bit</td>
<td>343 Bytes</td>
<td>80.0 GB</td>
</tr>
<tr>
<td>64 bit</td>
<td>183 Bytes</td>
<td>48.0 GB</td>
</tr>
<tr>
<td>32 bit</td>
<td>103 Bytes</td>
<td>32.0 GB</td>
</tr>
</tbody>
</table>

### V. Implementation

We implemented the basic secret key distribution scheme on a sensor networking testbed, consisting of TelosB motes (UC Berkeley, Crossbow) running the TinyOS 2.0 operating system. Random sequences of 64 bit were pre-distributed on four motes along with the corresponding 8-bit local key identifiers. We also stored in the memory of each mote its own identifier $n(\cdot)$ of 8 bit and a list $L_n(\cdot)$ containing the identifiers of the nodes with which it wants to communicate. Another mote played the role of the mobile node. In its memory, we stored the keys used by the other four motes, in all cases encrypted with a Vernam cipher, and included the corresponding identifiers.

In the experiment, the mobile node periodically broadcasts its HELLO messages. Upon receiving these HELLO messages, each mote sends back a message with its $n(\cdot)$ and $L_n(\cdot)$. The mobile node, upon receiving this message, verifies if each sensor node whose the identifier is listed in $L_n(\cdot)$ already informed the mobile node that it wants to communicate with $n(\cdot)$. For the nodes that do not satisfy this statement, the mobile node stores the information that $n(\cdot)$ wants to communicate with them. For the other nodes, the mobile node sends back a message containing the identifiers of the two nodes ($n(\cdot)$ and the one contained in $L_n(\cdot)$), the local key identifier that each node has to use and the XOR of the corresponding keys. This message is received by the pair of sensor nodes, and each one of them recovers the key of the neighboring node by running an XOR of the received data with its own key (corresponding to the received local key identifier). To check if the key was well decrypted by the two nodes, each of them sends the neighbor’s key in a message that is captured by an observer mote, programmed as a base station and connected to a standard personal computer.
We monitor all the communication steps using the “net.tinyos.tools.Listen” application. Our experiment shows that the scheme can be easily implemented in real sensor nodes and it works also for large-scale sensor networks.

VI. CONCLUSIONS

We presented a secret key distribution scheme for large sensor networks. Unlike [6] and [3], this is not a probabilistic scheme, i.e. any two nodes that can reach each other can communicate securely with probability one, using a small number of pre-stored keys albeit at the expense of a mobile node for bootstrapping. We presented several security extensions that exploit network coding to provide secret key distribution in large and dynamic sensor networks. Since our protocol and its extensions can easily accommodate for additional nodes, new keys and secured links, we deem the proposed network coding approach to be well suited for dynamic sensor networks with stringent memory and processing restrictions. Our conceptual results and practical implementations show that this approach leads to effective ways of generating pairwise and cluster keys, revoking compromised keys, authenticating and defending the mobile node used for bootstrapping the network. We believe that some of the proposed techniques will find natural applications also in other classes of mobile ad-hoc networks composed of devices with limited processing and transmission capabilities.

Although our use of network coding was so far limited to XOR operations, using linear combinations of symbols is likely to yield more powerful schemes for secret key distribution. Thus, part of our ongoing work is devoted to exploiting random linear network coding [21] and extending these ideas to multi-hop secret key distribution in highly dynamic networks.

APPENDIX A

PROOF OF THEOREM 1

Proof: First, notice that \( P(K_i = x) = \frac{1}{2^m} \). We shall prove that \( P(K_i = x|K_1 \oplus R = y_1, \ldots, K_m \oplus R = y_m) = \frac{1}{2^m}, \forall i \in \{1, \ldots, m\} \), which yields the result. Because \( K_i \oplus R = y_i \), we have that \( R = y_i \oplus K_i \). Thus, replacing \( R \) by \( y_i \oplus K_i \) in \( K_1 \oplus R = y_1, \ldots, K_m \oplus R = y_m \), we have that
\[ P(K_i = x | K_1 \oplus R = y_1, \ldots, K_m \oplus R = y_m) = P(K_i = x | K_1 \oplus K_i = y_1 \oplus y_i, \ldots, K_m \oplus K_i = y_m \oplus y_i), \]

where the event \( K_i \oplus K_i = y_i \oplus y_i \) is not present, because it is redundant. Let \( z_j = y_j \oplus y_i \), for \( 1 \leq j \leq m \) and \( j \neq i \). Let \( A \) denote the event \( \{ K_i = x \} \) and \( B \) denote the event \( \{ K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m \} \). Then, \( P(K_i = x | K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m) = P(A | B) \). By Bayes’ theorem, we have that

\[
P(A | B) = P(B | A) \cdot \frac{P(A)}{P(B)}. \tag{1}
\]

We already have seen that \( P(A) = P(K_i = x) = 1/2^n \). For \( P(B | A) \), we have that:

\[
P(B | A) = P(K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = z_m | K_i = x) = P(K_1 = z_1 \oplus x, \ldots, K_m = z_m \oplus x)
\]

\[
= \prod_{j=1}^{m} P(K_j = z_j \oplus x)
\]

\[
\overset{(a)}{=} \frac{1}{2^{n(m-1)}}
\]

where

(a) follows from the fact that \( K_1, \ldots, K_m \) are chosen independently;

(b) follows from the fact that the keys \( K_j \) are chosen uniformly from the set of words of length \( n \) of the alphabet \( \{0, 1\} \), denoted by \( \{0, 1\}^n \), resulting in \( P(K_j = z_j \oplus x) = 1/2^n \).

To compute \( P(B) \), let \( f : \{0, 1\}^n \rightarrow \{0, 1\}^n \) be defined by \( f(s) = s \oplus K_i \). We have that:

\[
P(B) = P(K_1 \oplus K_i = z_1, \ldots, K_m \oplus K_i = Z_m)
\]

\[
= P(f(K_1) = z_1, \ldots, f(K_m) = z_m)
\]

\[
\overset{(a)}{=} \prod_{j=1}^{m} P(f(K_j) = z_j)
\]

\[
= \prod_{j=1}^{m} P(K_j \oplus K_i = z_j)
\]

\[
\overset{(b)}{=} \frac{1}{2^{n(m-1)}}
\]
where

(a) follows from the fact that, because $K_1, \ldots, K_m$ are independent and $f$ is an injective function, $f(K_1), \ldots, f(K_m)$ are also independent;

(b) in this step, we use the fact that $P(K_j \oplus K_i = z_j) = 1/2^n$ (where $\{s_1, \ldots, s_{2^n}\}$ are the elements of $\{0, 1\}^n$), such that:

$$P(K_j \oplus K_i = z_j)$$

$$= P(K_j \oplus K_i = z_j | K_i = s_1) \cdot P(K_i = s_1) + \cdots$$

$$+ P(K_j \oplus K_i = z_j | K_i = s_{2^n}) \cdot P(K_i = s_{2^n})$$

$$= P(K_j = z_j \oplus s_1) \cdot P(K_i = s_1) + \cdots$$

$$+ P(K_j = z_j \oplus s_{2^n}) \cdot P(K_i = s_{2^n})$$

$$= \frac{1}{2^n} \cdot 2^n = \frac{1}{2^n}.$$

Therefore, replacing $P(A)$, $P(B)$ and $P(B|A)$ in (1), we have that

$$P(A|B) = \frac{\frac{1}{2^{n(m-1)}}, \frac{1}{2^n}}{2^{n(m-1)}},$$

and the result follows.

**ACKNOWLEDGMENT**

The authors gratefully acknowledge useful discussions with Rui Costa (University of Porto) and Prof. Virgil Gligor (University of Maryland).

**REFERENCES**


May 29, 2008


