A Secure and Energy-Efficient Key Generation* Mechanism for Wireless Sensor Networks

Murat Al1, Kenji Yoshigoe2
1Department of Applied Science, University of Arkansas at Little Rock, Little Rock, AR, U.S.A.
2Department of Computer Science, University of Arkansas at Little Rock, Little Rock, AR, U.S.A.

Abstract- This paper enhances the data confidentiality and energy saving of an existing solution. Our approach utilizes encrypted message as a key generator (KG) for the next message. Thus, a KG that used to be exposed is now encrypted. A masking policy is incorporated to an encryption technique to further improve the data confidentiality. Furthermore, the proposed method no longer requires a transmission of extra information in a form of KGs reducing energy consumption incurred by data transmission. Analysis shows that the proposed method significantly increases encryption strength and minimizes the damage area from the entire network down to a single node. At the same time, our method reduces the traffic volume, excluding network protocol headers, down to 50%. Thus, our solution is ideal for extending the life-span of a WSN while providing a solid security level.

Keywords: WSN, key generation, energy efficiency, confidentiality.

1 Introduction

Wireless Sensor Networks (WSNs) are widely used in both military and civil applications to monitor physical or environmental conditions including magnetism, temperature, sound, motion, vibration, pressure, and chemical elements. For instance, military surveillance applications, which in fact motivated the development of sensor network technology, can detect biological and chemical weapons [1]. Some of the civil applications are habitat monitoring, traffic monitoring, object tracking, and fire detection. A secure communication among sensor nodes is essential for many of these applications especially if sensitive information is exchanged in the network.

Various solutions on secure data transmission in sensor networks are discussed in literatures [2, 3, 4, 5, 7, 8, 9, 10]. A list of 18 different key establishment techniques is provided in [2]. These key management protocols are categorized as pre-deployed, arbitrated, and self-enforcing autonomous keying protocols. Kerberos, Otway-Rees, (Elected) Simple Key Distribution Center and Burmester-Desmedt Conference Keying are among the analyzed protocols. The paper compares these protocols based on the size of the exchanged messages as well as the computational resources required for key calculation on a number of different microprocessors.

A key management scheme consisting of key pre-distribution, shared-key discovery, and path-key establishment is proposed in [3]. Due to the memory constraints of sensor nodes each node is not capable of storing a large number of keys. Thus, every node is assigned a randomly chosen set of m keys out of a large pool of P keys where m << P. The paper provides extensive calculations for the probabilities of having shared keys for different network sizes and varying values for P.

Per-hop key exposure problem is addressed in [4]. When sending a common key over a path, the key is exposed to intermediate nodes along the path. An adversary needs to compromise only one of the nodes on this path to get the key. Here, the solution is to use multiple disjoint paths for negotiating a key so that each piece of information for negotiation takes a different path.

Many key exchange, key distribution, and key management protocols have been proposed [7, 8, 9, 10]. Key pre-distribution [10] is the most energy efficient scheme among these solutions. The drawback of this solution is that the communication in the whole network may be compromised once a message is decrypted. Key encryption in its various forms is one possible security mechanism, which can be used to improve the security of this scheme.

A confidentiality mechanism based on key encryption technique is proposed in [5]. Instead of maintaining a large key pool, only one key per tier is used in the network. Using a time stamp as a key generator and a hidden

---

* This material is based in part, upon work supported by the National Science Foundation, under Grants nos. CNS-0619069 and EPS-0701890. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
operation, a different key is produced for each message. The benefit of this approach is it saves memory, which would have been allocated for storing the key pool. However, the drawback is the exposed time stamp making the mechanism open to attacks.

In this paper, we propose a solution that enhances the existing key encryption method described in [5] in twofold. First, we increase the provided security level by hiding the key generator and adding masking policy to the hidden operations. Second, we reduce the energy consumed for data transmission.

The remainder of this paper is structured as follows. Section 2 describes the used network model and key predistribution scheme. In section 3 we explain the existing method and introduce our proposed encryption method. Section 4 is a theoretical evaluation of our method, and Section 5 summarizes the paper.

2 Background

This section describes three tiered network architecture that scales up well for large networks and key predistribution schemes used in WSNs.

2.1 Three Tiered Network Model

A three tiered network architecture is often used in WSNs. Figure 1 shows a star-mesh topology, an example of a three tiered network topology.

The first tier consists of sensor nodes with exceptionally limited resources and computational capacity. Nodes in this tier simply referred to as sensor nodes mainly perform processing, transmission, reception, and forwarding of data in addition to sensing of physical property within its proximity. Nodes in the second tier referred to as supernodes are the head of their cluster, consisting of a group of sensor nodes. They are typically equipped with more resources than sensor nodes.

They communicate with every sensor within their cluster and perform tasks including key management and data aggregation and forwarding. The node in the third tier also referred to as base station is a powerful device, which most likely has unlimited power supply and an Internet connection. It aggregates and processes data of the WSN, and is the interface to a remote operations center.

For the star-mesh topology, data transmission from sensor nodes to the according supernode is unicast. From the supernode to sensor nodes, the data transmission can be unicast or multicast. Data transmission in the second tier among supernodes can be unicast or multicast, and that to the base station is unicast. Data transmission from the base station to supernodes can be unicast or multicast.

2.2 Key Pre-assignment

Keys are assigned in the manufacturing phase. Each tier has its own secret key. Thus, there are three different keys when the network is deployed. Every sensor node in the first tier would be assigned the key $K_{T1}$. Similarly every supernode in the second tier would be assigned the key $K_{T2}$, and the base station in the third tier would be assigned the key $K_{T3}$. To be able to communicate with their cluster head, sensor nodes also store the key of the second tier ($K_{T2}$). To communicate with tier one and tier three, supernodes also store the keys $K_{T1}$ and $K_{T3}$. To communicate with the supernodes, the base station also stores the key of the second tier ($K_{T2}$).

We assume the mechanism described in [5]. A sensor node has a message $M$ of $x$ bits to transmit to its supernode in form of $k$ sub-messages $M_1, M_2, ... , M_k$ of $x/n^2$ bits each where $n$ is a predefined integer $0 < n^2 < x$. Thus, $n^2$ is fixed and known to all nodes. It can be assigned at the network construction phase. When $x$ is divided by $n^2$, the size of the last sub-message is less than $n^2$ in the most cases. Thus, the last sub-message $M_k$ will contain stuffed bytes to fill the message up to $n^2$ bits. All the pre-assigned keys $K_{T1}, K_{T2}$, and $K_{T3}$ are $n^2$ bits long. These sub-messages $M_i, i = 1, 2, ... , k$ from the sensor node to its supernode will be encrypted using the key $K_{T1}$. The encrypted sub-messages $\{M_i\}_{K_{T1}}$ are obtained by using the following equation:

$$\{M_i\}_{K_{T1}} = M_i \times K_{T1} \mod p$$  \hspace{1cm} (1)

where $p$ is a large prime number (usually in the order of 512 bits). It can be assigned in the construction phase of the network and is known to all nodes. Since $K_{T1}$, the pre-assigned key for all nodes in tier one is the same for all sub-messages $M_i$, this encryption method is too weak and not suitable for secure data transmission. After intercepting a packet, an intruder will be able to find the secret key $K_{T1}$ by utilizing some decryption algorithms. Here for example Euclid’s rule: Given $x$ and $n$, it finds $y$ such that $x^y \mod n = 1$. In this case, the term will be defined as follows:

![Figure 1. Star-Mesh Topology](image-url)
\[ \frac{\{M_1\}_{K_{T1}}}{\{M_1\}_{K_{T1}}} = \frac{M_i \times K_{T1}}{\{M_1\}_{K_{T1}}} \mod p \quad (1a) \]

The x in Euclid’s rule is a plaintext message \( M_i \), i.e. the attacker needs to know the message to be able to run this “known plaintext attack”. If he does not have a plaintext example, he will apply a brute force attack, to deduce the key \( K_{T1} \). The vulnerability of this method is that it uses the same key for all sub-messages. Once the key is detected, the network will become useless since adversaries can change original information (loss of data integrity) or feed the network with any misleading information.

3 Method

In this section, we describe two methods. In Section 3.1 we discuss the data confidentiality method proposed by [5], referred to as Existing Method. The terminology has been partly modified to match our work. In Section 3.2 we introduce our method, referred to as Proposed Method, which improves Existing Method.

3.1 Existing Method

The vulnerability described in section 2 can be mitigated by protecting the pre-distributed key. Such a key encryption mechanism is proposed in [5]. The main idea is to generate a new encrypted key, using the pre-distributed key, for every sub-message to be transmitted. A new encrypted key \( K_{T1}^{i} \) (in this example for the first tier), \( i = 1, 2, \ldots, k \), is gained by applying a hidden operation to the preinstalled key \( K_{T1} \) and a variable. This variable can be any physical value like time, temperature, or the concentration of a chemical element at the time the packet is being sent. The advantage of this system is that if a sub-message is intercepted and decrypted by an intruder, only the key \( K_{T1}^{i} \), to which we refer as one-time key, for this specific sub-message is revealed, not the preinstalled key \( K_{T1} \). This is true when we assume that the intruder does not know the hidden operation.

The variable used in [5] is a timestamp matrix \( T_i \) of \( n^2 \) bits, \( i = 1, 2, \ldots, k \). For each sub-message a different timestamp matrix \( T_i \) is used to generate a new encrypted key \( K_{T1}^{i} \), which is known to the sender only:

\[ K_{T1}^{i} = K_{T1} \circ T_i \quad (2) \]

The hidden operation \( \circ \) is a reversible binary operation like XOR or XNOR and is known to sender and the receiver. For a cluster, the supernode updates the operation from time to time. The cipher message \( C_i \) is obtained as follows:

\[ C_i = M_i \times K_{T1}^{i} \mod p, \quad i = 1, 2, \ldots, k. \quad (3) \]

Each packet sent in the network has the format shown in Figure 2. Every ciphered sub-message \( C_i \) is encrypted using the according key \( K_{T1}^{i} \).

The receiver needs to decrypt the new key \( K_{T1}^{i} \) first in order to decrypt the cipher message \( C_i \). To decrypt \( K_{T1}^{i} \), the receiver needs the value of the time stamp \( T_i \). Therefore, \( T_i \) is sent with each packet.

<table>
<thead>
<tr>
<th>( C_i = M_i \times K_{T1}^{i} \mod p )</th>
<th>Timestamp ( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n^2 ) bit</td>
<td>( n^2 ) bit</td>
</tr>
<tr>
<td>( i = 1, 2, \ldots, k )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Format of a packet with \( T_i \)

3.2 Proposed Method

The majority of energy consumption in a sensor node is found in transmission and receiving mode [6]. To minimize the energy consumption for sensor nodes we use a star-mesh topology in which a sensor node can reach its supernode in one hop. However, the described method would also work with other topologies. In this architecture, sensor nodes only communicate with their cluster heads and do not need to route any traffic from other sensor nodes, and routing only occurs in the second tier. Supernodes aggregate the information from sensor nodes in their cluster; they may perform some other operation to the gathered information and send the result to the base station. Since the base station might be out of transceiving range for some supernodes, nodes in the second tier provide routing capabilities. In addition, traffic from the base station to remote supernodes can be routed via intermediate supernodes.

The communication in the second and third tier is more sensitive and valuable than in the first tier, since the transmitted data may have been aggregated and may be consolidated compressed data from the network. Thus, the supernodes and the base station use a key size of 128 bit for their keys \( K_{T2} \) and \( K_{T3} \) respectively. Since the sensor nodes in the first tier have less energy, shorter keys of size of 64 bit can be used for \( K_{T1} \). Since the supernodes use a larger key size than the sensor nodes we assume the transmission of the key generators (see next section) is highly secure.

Encryption, decryption and transmission are performed on sub-messages only; we will refer to sub-messages as messages or packets for simplicity for the rest of the paper.

Confidentiality

We introduce a key generator, \( KG \), which itself is a key of \( n^2 \) bit of size. Like the time stamp \( T_i \), \( KG \) is used to generate a one-time key for a message as shown in (4) with the following differences from \( T_i \): \( KG \) is

1. distributed by a supernode to its cluster nodes after network setup,
2. known to the nodes in a particular cluster only,
3. used for a node authentication within a cluster,
4. not exposed, and
5. used to generate only the first one-time key.

In our example we will have traffic from the first tier to the second tier, i.e. a sensor node will send packets to the supernode. We use \( KG \) and the pre-distributed key \( K_{T_1} \) to generate the first one-time key \( K^{T_1}_{I} \) as shown in (4). The role of the key generator \( KG \) is limited to generating the first one-time key. The first encrypted key \( K^{T_1}_{I} \) is obtained using the following equation:

\[
K^{T_1}_{I} = K_{T_1} \circ KG
\]  

This key is used to encrypt the first cipher message \( C_1 \) using (3). The sender then transmits the ciphered message \( C_1 \) to the supernode. Upon receiving the first packet, the receiver will first build the encrypted key \( K^{T_1}_{I} \) by using key generator \( KG \) and (4).

\( K_{T_1} \) and \( KG \) are already known to the receiver, thus it can build \( K^{T_1}_{I} \). The receiver will obtain the first message \( M_1 \) from \( C_1 \) by using (3).

Since the receiver now knows \( M_1 \), for the second packet the sender (receiver) uses \( M_1 \) instead of \( KG \) to encrypt (decrypt) the key \( K^{T_2}_{I} \). The sender (receiver) will use \( M_1 \) to encrypt (decrypt) the third key and so on. Thus, the Proposed Method does not require a transmission of extra information like \( T_1 \) in the Existing Method, hereby reducing transmitted packet size and energy consumption incurred by the data transmission. In particular, the size of the data being transmitted becomes a half of that of the Existing Method (see Figure 2 and Figure 3).

Beginning with the second packet, we obtain the encrypted keys using the following equation:

\[
K^{T_i}_{I} = K_{T_i} \circ M_{i-1}, \quad i = 2, 3, \ldots, k
\]  

The general formula for obtaining all ciphered messages including the first message is (3). The new format for all packets is shown in Figure 3. It is half the size of the packet produced by [5].

\[
C_i = M_i \times K^{T_i}_{I} \pmod{p} \quad i = 1, 2, \ldots, k
\]

Once a one-time key is compromised, it is a straightforward computation (shown in Table 1) to obtain the pre-installed key \( K_{T_1} \), which enables an attacker to create new one-time keys using e.g. XOR operation.

### Table 1. Deducing \( K_{T_1} \) in Existing Method

| \( K^{T_1}_{I} \) | 0 | 0 | 1 | 1 | \ldots |
| \( T_i \) | 0 | 1 | 0 | 1 | \ldots |
| \( K_{T_1} \) | 0 | 1 | 1 | 0 | \ldots |

We enhance the strength of one-time keys \( K^{T_i}_{I} \) by introducing a masking policy that is used as a part of the hidden operation. Once in a while, according to the network policy, the supernode informs the sensor nodes in which (of any permutation of bit sequence) pattern the hidden operation is to be applied. For example the pattern could require the hidden operation (e.g. XOR) to be applied to every second bit (see Table 2). On bit pairs where the hidden operation is not applied, the value of \( KG \) (or \( M_{i-1} \)) remains.

Table 2. Creating \( K^{T_i}_{I} \) with Proposed Method

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K^{T_i}_{I} )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( KG )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( K^{T_i}_{I} )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
In this example, XOR operation is not applied to the first, third, fifth, and seventh bit-pair \((K_{T1}, KG)\) so that the value of these bits in \(K_{T1}\) equals the value of the bits in \(KG\). For \(n\)-bit variables, \(K_{T1}\) and \(KG\), \(2^n\) possible bit-pairs of \(K_{T1}\), \(KG\) exist. As it was the case with \(KG\), this masking pattern is multicast to the cluster using a strong encrypted key so that the pattern is not easily accessible to an adversary.

### Authentication

A low complexity public key encryption for authentication, in which unique node IDs, distributed by the cluster head, serving as public keys was investigated in [5]. No further explanation exists for how secure the distribution of these unique IDs is (i.e., the same security level as the sensed data). Furthermore, the availability of a public key of a sender at potential receivers is not mentioned. Instead, the study describes that a receiver authenticates the sender by calculating a value “\(n\)” by using a public key cryptographic mechanism and two values “\(a\)” and “\(b\)” referred to as unknown variables. The performance evaluation of this authentication method is not possible from the explanations, and the authors did not present such an analysis in their study.

The \(KG\) in our approach is distributed and updated by the supernode using a stronger encryption than used for sending sensed data by sensor nodes. Thus we assume the transmission of \(KG\) is secure and a node that uses \(KG\) is an authenticated member of the cluster. If an adversary comes into possession of a sensor network ID as used in [5] he can authenticate himself for the rest of the life time of the network. In the Proposed Method, by using \(KG\) in the first message, a sensor node is considered to be authenticated for subsequent messages until a new KG is issued. Furthermore our \(KG\), which is already used to achieve confidentiality, also can be used to achieve authentication without any additional overhead.

### 4 Evaluation

There is no encryption algorithm that is unconditionally secure, i.e. the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext [11].

There are two general approaches to attack such a conventional encryption scheme:

- **Cryptanalysis:** The attacker knows the algorithm, some characteristics of the plaintext and perhaps has some plaintext-ciphertext pairs. It is used to deduce a specific plaintext or the key [11].

- **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained [11].

Let us assume that the messages sent in the network consist of intelligible English text. The equation for obtaining the cipher text in the existing method can be expressed as (3). If an adversary was able to intercept a packet, he can see \(C_i\) and \(T_i\). Since \(T_i\) is not encrypted and therefore is known, according to the above given definition he can use a cryptanalysis attack to decipher \(C_i\). A cryptanalysis attack takes less time than a brute force attack; however, estimating the amount of effort required to cryptanalyze ciphertext successfully is a great challenge [11]. Therefore, in the following we will apply brute force attacks to compare the effort required to deduce a key in both the existing and the proposed methods.

For both methods, we assume that the adversary knows the used equations to create encrypted keys and encrypted messages, and also that he knows the hidden operation \(\circ\). Furthermore, we assume the attacker can guess \(10^{12}\) keys per second. Finally, we decide to use a key size of 64 since larger key sizes as used in public cryptography are generally not feasible for WSNs due to their computational and power constraints. However, a 128-bit AES key or a 168-bit 3DES key can be used if desired or required. The chosen key size will make no difference for the results of relative comparison between these two methods.

When we use a 64-bit key size, the number of alternative keys is \(2^{64} \approx 18 \times 10^{18}\). Statistically he needs to try out 50% of the keys. Assuming that the attacker uses a distributed computer system which can try out \(10^{12}\) keys per second, the time to obtain the key is:

\[
9.223 \times 10^{18} / 10^{12} = 9.223,000,000 \text{ sec} \approx 106.5 \text{ days}.
\]

This is a theoretical evaluation where our goal is to perform a relative comparison between the two encryption algorithms. Hence, the time calculated above for breaking the key will vary depending on different decryption speeds but the relative difference between them will be the same.

#### 4.1 Existing Method

From an intercepted message (for indexes let us say the first message), the attacker obtains the cipher message \(C_i\) and the time stamp \(T_i\). He applies a brute force attack on \(C_i\) using (3), \(C_i = M_i \times K_i^{T1} \mod p\). This attack reveals \(M_i\) and \(K_i^{T1}\) after 106.5 days. The attacker is interested in the preinstalled key \(K_{T1}\). He puts the obtained value of \(K_i^{T1}\) and \(T_i\) into (2), \(K_i^{T1} = K_{T1} \circ T_i\) to immediately derive \(K_{T1}\). For the hidden operation the attacker tries XOR and XNOR. In this scenario, \(K_{T1}\) is revealed after a total time of 106.5 days. It should be noted that the damage area of a single attack is not restricted to the compromised node or even to its cluster. Instead, once \(K_{T1}\) is revealed the attacker can read every message he intercepts throughout the entire network for its lifetime since every node uses \(K_{T1}\) and a timestamp that is exposed.

#### 4.2 Proposed Method

Analog to the existing method, when the attacker applies a brute force attack on the first cipher message \(C_i\) using (3)
$C_I = M_I \times K_{T_I}^{T_I} (\text{mod} p)$, he will obtain the first one-time key $K_{T_I}^{T_I}$ and the first message $M_I$ after 106.5 days. Here, the first one-time key $K_{T_I}^{T_I}$ was built using (4), $K_{T_I}^{T_I} = K_{T_I} \bigotimes KG$. In this equation there are two unknowns namely $K_{T_I}$ and $KG$, and both are keys. Even a brute force attack would not make sense because no intelligible text exists, which could help him decide whether or not a tested key is correct. On the other hand, if the attacker applies the brute force attack on the second cipher message, $C_2$ using (3), he will obtain the second one-time key, $K_{T_2}^{T_2}$, and the second message $M_2$ after 106.5 days. $K_{T_2}^{T_2}$ can be obtained using (5), $K_{T_2}^{T_2} = K_{T_2} \bigotimes M_{i-1}$. Again, we have two unknowns in this equation namely $K_{T_2}$ and $M_{i-1}$. However, a second brute force attack now would make sense, since $M_{i-1}$ is intelligible text. Under same conditions, deducing $K_{T_2}$ would take another 106.5 days. Thus, a total of 213 days are required for the attacker to deduce $K_{T_2}$ and $K_{T_1}$.

In the previous evaluation, we did not consider the range of masking patterns that is used to determine the location of the hidden operation to be applied. For 64-bit variables, $K_{T_I}$ and $M_{i-1}$, the number of possible masking patterns is $2^{64}$. However the added complexity is not of factor $2^{64}$. To evaluate the benefit of our masking scheme, we consider two cases, Table 3, if no masking was applied, and Table 4, if masking was applied to every bit. The hidden operation used is $\text{XOR}$. An attacker would see, in Table 3 the bit-pairs $K_{T_I}^{T_I}=0$, $M_{i-1}=0$ and $K_{T_I}^{T_I}=0$, $M_{i-1}=1$ produce the same results for $K_{T_I}$ as in Table 4. That is, masking has no effect on these two bit-pairs, and it affects only two out of four possible outcomes for $K_{T_I}$. Thus, the added complexity is only a factor of half of $2^{64}$.

Table 3. No masking

<table>
<thead>
<tr>
<th>$K_{T_I}^{T_I}$</th>
<th>$M_{i-1}$</th>
<th>$K_{T_I}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4. Masking

<table>
<thead>
<tr>
<th>$K_{T_I}^{T_I}$</th>
<th>$M_{i-1}$</th>
<th>$K_{T_I}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The probability for $K_{i}^{T_I}=0$ is $\frac{1}{2}$, and the same applies to $M_{i-1}$. Thus, the probability for a bit-pair of $[K_{i}^{T_I}, M_{i-1}]$ to be $[0,0]$ is $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. Similarly, the probability for the bit-pair to be $[0,1]$ is also $\frac{1}{4}$. For these bit combinations the attacker knows the outcome, whether the hidden operation is applied or not, assuming that he also knows the used hidden operation is $\text{XOR}$. The probability that bit-pairs $[K_{i}^{T_I}, M_{i-1}]$ have the combination $[0,0]$ or $[0,1]$ out of all possible combinations $[0,0],[0,1],[1,0],[1,1]$ is $\frac{1}{4}$. The worst case for the security of $K_{T_I}$ is that all bit combinations of $[K_{i}^{T_I}, M_{i-1}]$ consist of pairs of either $[0,0]$ or $[0,1]$, thus combinations $[1,0]$ and $[1,1]$, for which masking makes a difference do not occur. For a sequence of $m$ bits, this probability is $(\frac{3}{4})^m$. Thus the probability for the worst case (for our 64 bit key) is $(\frac{3}{4})^m = 1/18446744073709551616 \approx 0.542 \times 10^{-21}$.

On average half of the bit-pairs will be $[0,0]$ or $[0,1]$ and another half will have the combination of $[1,0]$ or $[1,1]$. Thus for a key of 64 bit, an attacker will need to try out a half of the possible $2^{64}$ bit-pairs. Thus, the total time for cracking $K_{T_I}^{T_I}$ and $K_{T_I}$ is derived as

$$106.5 \times \frac{2^{64}}{2} \approx 2.69 \times 10^{18} \text{ years}$$

Another benefit of using $KG$ instead of $T_i$ is the limitation of the lifetime of an attack. For the Existing Method, if $K_{T_I}$ is compromised the adversary can create one-time keys as long as the network exists. Let us assume a sensor node creates messages every minute and the network exists for one month. The number of messages created by one node is $60 \times 24 \times 30 = 43200$. For example, if an adversary cracks $K_{T_I}$ after 10 days (instead of 106.5 days by using more computational power); he can send 28800 messages into the network ($60 \times 24 \times 20 = 28800$).

In the Proposed Method, besides $K_{T_I}$ also $KG$ or $M_{i-1}$ is needed to create one-time keys. How often $KG$ is to be updated by the supernode depends on the network policy and can be adaptive to current power level of the cluster nodes or the required security level at different locations of the network.

For example we choose a $KG$ life cycle of 10,000 messages. That is, after receiving a total of 10,000 messages from the nodes in its cluster, the supernode will issue a new $KG$. In this time some sensor nodes may have sent no packets to the supernode and some may have sent many packets. For a cluster size of 100 nodes the average number of sent packets per node is 100. The first message built using $KG$ is not suited for an attack as mentioned earlier in this section. The number of previous messages, $M_{i-1}$, that can be used to generate the next key is 99. As for the worst case, taking the second message, if an attacker can break $K_{T_I}^{T_I}$, $M_I$ and $K_{T_I}$ using (3) and (5), within one minute or before the sender of this message sends its third message, the attacker can read the next 98 messages. With our approach this scenario is extremely unlikely since the average time required to do so is $2.69 \times 10^{18}$ years. The result will be very similar for an attack with higher computational power.

For the Existing Method, revealing $K_{T_I}$ will lead to decryption of every intercepted message in the whole network since every node uses $K_{T_I}$ and a timestamp that is exposed. In our method, the hidden operation and the masking pattern are used per cluster; however, every sensor node has a different $K_{T_I}$. Thus, he can read a set of messages sent only by the sensor node to which the deduced $K_{T_I}$ belongs.

## 5 Conclusion

In this paper, we proposed a secure and energy-efficient key exchange mechanism that improves the data confidentiality and energy saving of a previous solution in wireless sensor networks (WSNs). It utilizes an encrypted message instead of an exposed timestamp as a key generator for the next message. Since a key generator that
used to be exposed is now encrypted, the proposed key exchange mechanism improves the data confidentiality. A masking policy incorporated to the hidden operations further improves the data confidentiality.

Moreover, the proposed method reduces energy consumption incurred by data transmission because a preceding message is used as a key generator, and no additional key generator needs to be transmitted.

Analysis shows that the proposed method significantly increases the encryption strength (from $10^{6.5}$ days to $2.69 \times 10^{18}$ years) in addition to reducing the damage area, when being compromised, from an entire network down to a single node. Furthermore, the amount of data transmission required by the existing approach in [5] is reduced to 50% with our method. Considering the fact that receiving and transmitting of one bit consumes as much energy as a processor needs to execute 100 instructions [12], the energy savings with our method are significant.

In general, the higher the provided security level is, the higher the energy requirements of such a system are; however, the proposed key generation and distribution mechanism is very efficient in terms of storage and energy consumption while achieving excellent confidentiality. Therefore, our solution is ideal for extending the life-span of a WSN while providing a solid security level.

6 References


