

SMART: Secure Multi-pAths Routing for wireless sensor neTworks

Noureddine Lasla, Abdelouahid Derhab, Abdelraouf Ouadjaout, Miloud Bagaa, Yacine Challal

▶ **To cite this version:**

Noureddine Lasla, Abdelouahid Derhab, Abdelraouf Ouadjaout, Miloud Bagaa, Yacine Challal. SMART: Secure Multi-pAths Routing for wireless sensor neTworks. 13th International Conference, ADHOC-NOW, Jun 2014, Benidorm, Spain. pp.332-345, 10.1007/978-3-319-07425-2_25 . hal-01308973

HAL Id: hal-01308973
https://hal.science/hal-01308973

Submitted on 28 Apr 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SMART: Secure Multi-pAths Routing for wireless sensor neTworks

Noureddine Lasla[1], Abdelouahid Derhab[2], Abdelraouf Ouadjaout[1], Miloud Bagaa[1], and Yacine Challal[3]

[1] Department of Theories and Computer Engineering, CERIST, Algiers, Algeria
[2] Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh, Saudi Arabia
[3] Laboratoire de Méthodes de Conception des Systèmes (LMCS), Ecole nationale Supérieure d'Informatique, Algiers, Algeria

Abstract. In this paper, we propose a novel secure routing protocol named Secure Multi-pAths Routing for wireless sensor neTworks (SMART) as well as its underlying key management scheme named *Extended Two-hop Keys Establishment* (ETKE). The proposed framework keeps consistent routing topology by protecting the hop count information from being forged. It also ensures a fast detection of inconsistent routing information without referring to the sink node. We analyze the security of the proposed scheme as well as its resilience probability against the forged hop count attack. We have demonstrated through simulations that SMART outperforms a comparative solution in literature, i.e., SeRINS, in terms of energy consumption.

1 Introduction

Wireless sensor networks (WSNs) [1] are defined as a large collection of tiny sensor nodes, which have scarce resources regarding energy, bandwidth, processing capacity and storage. Such networks are designed to gather data in inhospitable places and might be involved in critical applications meant for civil and military use. The main task of a wireless sensor network is to collect/aggregate data from the sensor nodes and transmit them towards the sink node using a hop-by-hop communication. In these critical applications, establishing a reliable path free of compromised nodes is an important security concern.

The single-path routing is not resilient to attacks as it is sufficient to compromise one node along the path to cause path failure. To deal with this failure, a path maintenance process is initiated to find a new path, which is costly in terms of time, control overhead, and energy consumption. The use of multi-path routing can be a good solution against attacks that target the reliability of the network. As data are transmitted redundantly through multiple paths, the packets are likely to reach the sink even in the presence of some compromised nodes.

The attacks against wireless sensor networks can be either *insider* or *outsider* according to whether or not the adversary retrieves the information stored in

the sensor nodes. Using cryptography mechanism, the outsider attacks can be avoided as there is no way for an attacker to inject or read information from the network. The insider attack, however, is more powerful as by compromising a set of sensor nodes, an adversary can get access to the security materials of these nodes, change their running codes, and inject false information. For example, in routing construction protocols, an adversary can succeed at launching the Sinkhole attack by simply injecting a faked shortest path RREQ message using the cryptographic materials of the compromised nodes.

To construct the routing topology, different metrics are employed. Among them, we can find the hop count, sequence number, path identifier, etc. The hop count for example is used to select the shortest path leading to the sink and avoid routing loops, where each node should increment it by one before relaying it to its next hop. However, these metrics are mutable information, meaning that every node could manipulate it during the relay. In a security context, this mutable information is attractive for adversaries who want to compromise the network and can be exploited by many attacks like the sinkhole and the wormhole attacks. Using only cryptography techniques to ensure the integrity of route construction information (hop count, sequence number, etc.) is not sufficient especially when the adversary compromise a set of nodes in the network, as mentioned earlier. For these reasons, detecting compromised nodes is an important security concern that should be considered when designing a secure communication protocol.

In the literature, some solutions [2–5] have been designed to build a reliable routing topology. Authors in [5] propose a protocol to secure tree construction, based on braodcast key to authenticate neighboring nodes. However, although this protocol is resilient to node replication attack, the protocol cannot protect the transmitted routing information from being altered or forged when an adversary compromise a node. SEIF [3] allows the construction of more alternative disjoint paths belonging to different sub-branches. Each path from different sub-branches can be only intersected at nodes that are at one hop from the sink, and each sub-branch is tagged with a unique identifier that guarantees the construction of such topology. Furthermore, to ensure the security of the sub-branch identifiers, authors use a set of one-way hash chains to authenticate messages from the sink and the sub-branches origin. Therefore, any attempt of injecting faked sub-branch identifier can be immediately detected even if the adversary make an inside attack. However, SEIF cannot detect a Wormhole attack, when an adversary captures a valid hash chain of sub-branches from one end and replay them at the other end [6]. In addition, SEIF does not consider any metric to carry out the routing decision.
SeRINS [2] is a semi-distributed solution based on the hop count metric to select routes. This protocol provides a mechanism to protect the hop count information at the sensor node level with the help of the sink. Each sensor, first, chooses its first parent that will be used as a reference to verify the correctness of any received alternate route. After that, when a node suspects on one alternate route, it sends an alert to the sink, which makes a decision about whether the suspected or the alert sender node is malicious. However, involving the sink in the verifi-

cation process of suspected nodes, causes considerable communication overhead and affects negatively the network scalability.

In this paper we propose a novel multi-path routing protocol called *Secure Multi-pAths Routing for wireless sensor neTworks* (SMART), relying on a extended version of our previous key management scheme EPKE [7], we call it *Extended Two-hop Keys Establishment* (ETKE). The main contributions of the paper are the following: Firstly, the establishment of two-hop broadcast keys enables the authentication of two-hop neighbors, which allows applying the watchdog mechanism to check the well-behaving of one-hop neighbors during the relay of RREQ messages. Secondly, we show that if two consecutive nodes along the path are not compromised, SMART ensures an immediate detection of inconsistent routing information without referring to the sink, using a two-hop verification mechanism. Thirdly, if the two consecutive nodes are compromised, an analysis of detection evasion is provided.

The remaining of this paper is organized as follows: Section 2 provides system model and basic idea. In Section 3, description of ETKE is presented. Section 4 describes SMART. Security analysis and simulation results are given in Section 5. Finally, Section 6 concludes the paper.

2 System model and basic idea

2.1 Attack model

We consider the node capture attack, in which an attacker can capture a legitimate node and turn it into a malicious one by extracting cryptographic keys from the captured node and makes it run a malicious code. The compromised node then broadcasts a fake RREQ with false hop count in order to attract the traffic that is destined to the sink. This mechanism is used by some attacks such as : Sinkhole and Wormhole.

2.2 Basic idea

In the tree-based construction process, each node forwards a RREQ message initiated by the sink node. The RREQ message contains a mutable information (hop count) that should be incremented by one at each relay level. To secure the routing topology construction, we should ensure a correct alteration of the hop count value during the relay.
The idea behind our solution is to provide a secret that is shared between each node and its two-hop neighbors. This secret is unknown for the one-hop neighbors and allow to verify their behaviour during the relay by the two-hop neighbors.

For instance, let us consider the network in Fig. 1(a), where the dashed lines represent the communication links between neighboring nodes and the solid lines represent the selected routing paths. The number besides each node represents the hop count information relayed by that node. In Fig. 1(b), if c (the child node of parent p) is a compromised node, without any security mechanism, it can

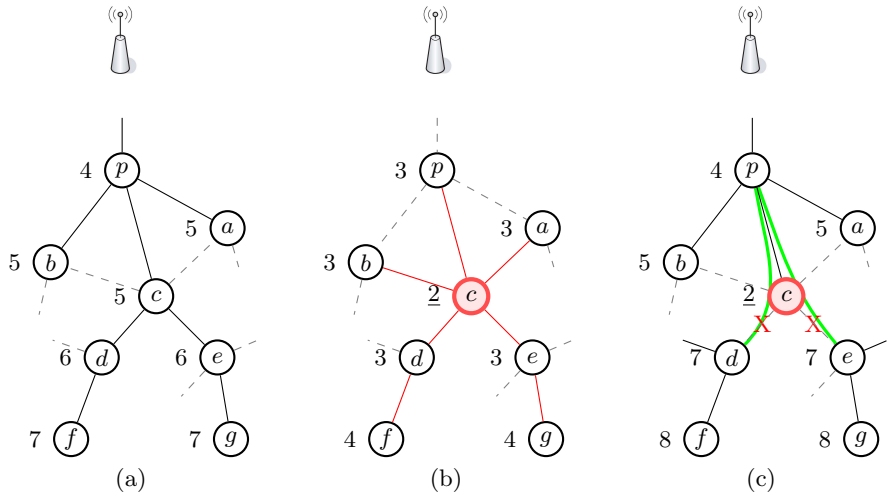


Fig. 1. The two-hop verification mechanism used in SMART

forge inconsistent hop count with value 2, in order to make most of the network traffic pass through it without being detected. Our idea, as shown in Fig. 1(c), is to share a secret (virtual tunnel) between node p and its two-hop neighbors d and e. This secret is used by node p to generate a proof (encryption of the current hop count vlaue). This proof is forwarded by node c without being able to decrypt it, and allows node d and e to verify if c has correctly incremented the hop count value received by its parent p. Note that each node, during the relay, (i) forwards the received proof to verify its correct relay of the received RREQ and also (ii) generates its own proof to ensure that the next forwarder node correctly relays its RREQ message.

In the next section, we give a detail description of how nodes in the network share a secret keys with their one-hop and two-hop neighbors, allowing then to check the correct relay of the RREQ messages.

2.3 Notations

The following notations in Table 1 are used throughout the paper.

Table 1. Notations

Notation	Description
$E(K, m)$	Encryption of m using key K
$A \rightarrow B : m$	A sends m to B
$A \rightarrow * : m$	A broadcasts m
$K_{A,B}$	Secret pairwise key between A and B
BK_A	Broadcast key of A
$\|\|$	Concatenation operator

3 ETKE description

In this section we describe how to establish the required key materials between communicating nodes to secure the tree-based routing construction in SMART. ETKE extends EPKE [7] by adding the two-hop pairwise key and the two-hop broadcast key. Mainly, the required keys are the two-hop and one-hop broadcast keys, shared between each node and its one-hop and two neighbors, respectively.

To achieve key agreement between communicated nodes, the pre-distribution method is more suitable for WSN [8]. In this method, nodes are preloaded, prior to deployment, with secret information that will be used to establish secure links between neighboring nodes. In this section, we propose a solution to establish the one hop pair-wise key, one hop broadcast key, two hops pair-wise key and two hops broadcast key, based on the Transitory Initial Key setup scheme of LEAP [9] and OTMK [10]. The transitory initial key K_{IN} is used to establish keys between neighboring nodes during a key setup phase (trust period). To secure nodes against capture attacks, the K_{IN} is erased from the node's memory at the end of the trust period. The trust period represents the minimum time (T_{min}) needed by an adversary to compromise a legitimate node.

3.1 Key setup phase

Each node u is preloaded with a transitory initial key K_{IN} and a random number N_u. Node u compute its master key MK_u as follows: $MKu = G(K_{IN}, ID_u)$, where G is a pseudo random function. After T_{min}, each node u erases K_{IN} and N_u from its memory. Node u discovers its neighbors by broadcasting the following message

$$u \rightarrow * : Join1, u, E(K_{IN}, u||N_u) \tag{1}$$

When u's neighbors receive this message, they relay it to u's two-hop neighborhood as follows

$$Relay_node \rightarrow * : Join2, u, E(K_{IN}, u||N_u) \tag{2}$$

Depending on the state of node u's neighbors, i.e., still keeping the initial key K_{IN} or already erased it, we distinguish two cases to establish one-hop pair-wise key, two-hop pair-wise key, one-hop broadcast key and two-hop broadcast key.

3.2 Case 1: K_{IN} is available

Creation of one-hop pair-wise key: To create the symmetric one-hop key $K_{u,v}$ between two direct neighbor nodes u and v, after they receive message (1) from each other, the following formula is used:

$K_{u,v} = G(MK_{min(u,v)}, ID_{max(u,v)}||N_{max(u,v)})$

Note that u and v can compute $MK_{min(u,v)}$ because each one knows K_{IN} and can generate the master key of any other node.

Creation of two-hop pair-wise key: When two-hop neighbor nodes u and v receive message (2) from their relay nodes, the following formula is used to create the symmetric key between them:

$K_{u,v} = G(MK_{min(u,v)}, ID_{max(u,v)}||N_{max(u,v)})$

Creation of one-hop broadcast key: Each node u creates one-hop broadcast key, which it shares with its direct neighbors. This key is not used to authenticate the node but to encrypt the message content. When node u's neighbor receives message (1), it uses K_{IN} to generate u's master key MK_u, and computes u's one-hop broadcast key as the following:

$BK_u = G(MK_u, N_u)$

Creation of two-hop broadcast key: After the reception of message (2), each u's two-hop neighbor node uses K_{IN} to compute u's master key MK_u, and computes u's two-hop broadcast key as follow:

$BK2_u = G(MK_u, N_u||N_u)$

3.3 Case 2: K_{IN} is not available

After T_{min}, each node v erases the transitory initial key K_{IN}, and will not be able to generate other master keys. It can only use its master key to calculate a symmetric key with a new deployed node u (since u can generate any master key). So, the following messages must be generated in order to establish a par-wise key between them: If node v is a direct neighbor of node u, then, the following message, $Reply1$, is sent to u:

$$v \rightarrow u : Reply1, v, E(MK_v, v||N_v) \tag{3}$$

Otherwise, the following two messages $Reply2$ and $Reply3$ are sent to relay nodes and to node u respectively.

$$v \rightarrow Relay_node : Reply2, v, E(MK_v, v||N_v) \tag{4}$$

$$Relay_node \rightarrow u : Reply3, v, E(MK_v, v||Nv) \tag{5}$$

Creation of one-hop pair-wise key: The one-hop pair-wise key $K_{u,v}$ between two direct neighbor nodes u and v is computed using the following formula:

$K_{u,v} = G(MK_v, ID_u||N_v)$

Creation of two-hop pair-wise key: To calculate a two-hop pair-wise key $K_{u,v}$ between two-hop neighbor nodes u and v, the following formula is used:

$K_{u,v} = G(MK_v, ID_u||N_v)$

Creation of one-hop broadcast key: Because u's neighbor (i.e., node v) has erased the transitory initial key K_{IN}, it cannot calculate the one-hop broadcast key BK_u of node u using the previous formula. The only way to do so is that u sends BK_u via a unicast encrypted packet using the shared one-hop pair-wise key with node v.

$u \rightarrow v : u, E(K_{u,v}, u||BK_u)$. This message is sent by node u when it receives $Reply1$ from node v.

Creation of two-hop broadcast key: The only way for v to get the two-hop broadcast key $BK2u$ of node u, is that u sends the $BK2u$ via a unicast encrypted packet using the shared two-hop pair-wise key with node v.

$u \rightarrow v : u, E(K_{u,v}, u||BK2_u)$. This message is sent by node u when it receives $Reply3$ from node v.

3.4 Special case

When node v is newly deployed in the network, it can be inserted between two nodes i and j that was not neighbors and become two-hop neighbors through node v. In this case, node i and j should exchange the two-hop broadcast keys of each other. The only way to do so is to transmit the two-hop broadcast through node v during the trust period, using their master key, as follow.

$$i \rightarrow v : Join4, i, E(MK_i, i||BK2_i) \tag{6}$$

$$j \rightarrow v : Join4, j, E(MK_j, j||BK2_j) \tag{7}$$

Node v then forwards $Join4$ to i and j using their master keys to ensure that node v is a trust node as only node that has not yet erased its initial key, can generate the master keys of other nodes. The following $Join5$ message is then sent to j and j.

$$v \rightarrow j : Join5, v, E(MK_j, v, i||BK2_i) \tag{8}$$

$$v \rightarrow i : Join5, v, E(MK_i, v, j||BK2_j) \tag{9}$$

4 SMART description

4.1 Initialization

Global One-way Hash Chain (GOHC): Prior the deployment, a GOHC is generated $(S_0, S_1, S_2, \cdots, S_n)$ and stored in the sink node. Each sensor node is preloaded with the last value (S_n) of the GOHC. After the deployment, the sink node, at each round d, includes the last unused GOHC Value (S_{n-d}) in the RREQ message. Each node that receives RREQ of round d $(d = 1, \cdots, n-1)$ can check if RREQ is generated by the sink or not. This is achieved by applying the one-way hash function on the received value S_{n-d} and verifying whether the result is equal to the pre-loaded GHOC value $S_{n-(d-1)}$; $F(S_{n-d}) = S_{n-(d-1)}$.

Local One-way Hash chain (LOHC): To allow a one hop authentication of the broadcasted RREQ message, each node i generates a LOHC $(L_i^0, L_i^1, L_i^2, \cdots, L_i^n)$ and reveals the last LOHC value (L_i^n) to its reachable neighbors using its one-hop broadcast key to encrypt it L_i^n. At round d $(d = 1, \cdots, n-1)$, each node can check if RREQ is generated by its one-hop neighbor by applying the function F on the received value L_i^{n-d} and verifying whether the result is equal to the pre-loaded LOHC value $L_i^{n-(d-1)}$; $F(L_i^{n-d}) = L_i^{n-(d-1)}$.

4.2 Route construction process

Route construction initialisation: To start the route construction process, the sink node broadcasts the below RREQ packet, which includes the following fields as shown in Table 2

$$Sink \rightarrow * : ID_{sink}, Seq = S_{n-d}, OWC = L_{sink}^{d}, \\ h = 0, MAC2_{sink}, \emptyset, \emptyset \tag{10}$$

Table 2. RREQ message Fields

Field	Description
ID_{Src}	Packet sender identifier
Seq	The first unused GOHC value
OWC	First unused One Way Chain value of a LOHC to authenticate the source
h	Hop count value
$MAC2_{src}$	$MAC(BK2_{src}, h)$; the MAC of the hop count value, generated with the two-hop broadcast key
ID_{Parent}	Primary (or main) parent identifier
$MAC2_{parent}$	The MAC received from the primary parent, generated with the parent's two-hop broadcast key

Primary parent selection: When a node i receives the first RREQ message, which indicates a new round d from a node j, it waits for a random time then it selects the primary parent node with the lowest hop count h' and relays the RREQ message as follows:

$$i \rightarrow * : ID_i, Seq = S_d, OWC = L_i^d, h = h' + 1, \\ MAC2_i, ID_{parent}, MAC2_{parent} \tag{11}$$

After relaying the RREQ message, node i might keep receiving RREQ messages from other neighbors.

Alternative parent selection: When node i with a hop count h receives a RREQ message from node j with a hop count h', the following conditions must hold true so that j is accepted as an alternative parent:

- $h' \leq h$, in order to avoid routing loops.
- The grand parent of the received route must not exist as a grand parent or a parent of an already selected accepted route, in the routing table, except the case when the grand parent is the Sink. The accepted routes from different grand parent nodes guarantee that all paths are two-hop disjoint.

For each accepted RREQ message, node i adds a new entry in the routing table, which contains the following information: <*Parent-id*, *Hop-count*, *Grand_Parent-id*>.

5 Analysis of SMART protocol

5.1 Security mechanisms

In SMART, the RREQ message encapsulates three security mechanisms that ensure sink authentication, source authentication and hop count integrity.

- The *Seq* field, containing the first unused GOHC, allows sensor nodes to verify that the RREQ message is initiated by the sink node and is not a re-injection of an old RREQ message of a previous round.
- To forbid an intruder from spoofing source identities, each node i, when receiving a RREQ from node j, can authenticates the message source by checking if the stored $L_j^{n-(d-1)}$ is equal to the hash value of the received OWC, $F(L_j^{n-d})$.
- The integrity control of the hop count is ensured through the $MAC2_{Grand_parent}$ field. This MAC ensures that the received hop count value is a successor of the hop count of the grand parent. By this way, an adversary which tries to send a RREQ with lower hop count to perform a Sinkhole attack, is then prevented.

5.2 Forged Hop count detection

For an intruder to inject forged routing information, he should first get the cryptographic key materials by compromising some nodes in the network. However, even by compromising some set of nodes, our scheme can detect, in most of cases, the injection attempt of faked RREQ messages. In the following we give three possible cases of attack scenarios by an intruder and show how SMART can protect the routing construction by detecting these attacks.

1. When a compromised node c wants to inject forged routing information, it relay a RREQ of its parent p with forged hop count h' instead of h (see Fig. 1(b)). In this case, any node receiving or overhearing this RREQ can check that h' is not a successor of $h-1$, by simply comparing if $MAC2(BK2_p, h') = MAC2_p$.

2. A more intelligent attack consists in compromising two or more consecutive nodes along the path simultaneously. The first possible scenario is that nodes p, and c are compromised and node c try to inject a RREQ with a forged hop count h'. In this case, node a and b (i.e., any common neighbor of both c and p) using the watch-dog mechanism (i.e., the watchdog consists in monitoring the neighboring nodes to check whether the latter are correctly forwarding the RREQ messages or not), detect that the injected RREQ is not consistent with the previous received RREQ from node p, and they will report a detection message to forbid the concerned nodes d and e to relay this RREQ. The adversary can succeeded at launching such an attack only if all the common neighbors of p and c are compromised. We will present in the next section the probability of success of such an attack under different network densities.
3. In the third possible scenario, both nodes p and c inject a RREQ with a fake hop count. Node a and b, in this case, cannot detect the inconsistency of the received RREQ message. However the parent node of p and any common neighbor between them can detect the forged hop count and then send a detection message to the concerned nodes to detect and reject this RREQ.

5.3 Resilience probability against forged hop count

The fake RREQ can propagate to the lower levels of the network without being detected if the following conditions hold true:

1. Two neighboring nodes in the network, the parent and the child in the tree, are compromised.
2. The child node injects a fake RREQ with false hop count.
3. There are no common legitimate neighbors between the parent and the child node.

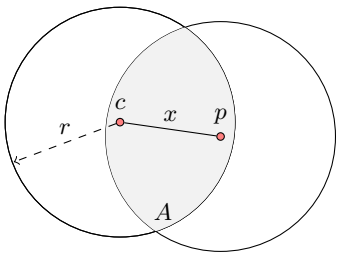


Fig. 2. The intersection region A that contains the common neighbors of p and c.

The region where the common neigbors of the parent p and child c can reside, represents the intersection of two circular communication areas with radius r and centered at p and c . As shown in Fig. 2, The distance x, between the two nodes

p and c, ranges from 0 to r. The area A of the common region is calculated as follows:

$$A(x) = 2r^2 cos^{-1}(\frac{x}{2r}) - x\sqrt{r^2 - \frac{x^2}{4}} \tag{1}$$

The expected area of A, as shown in [11], is calculated as follow:

$$E[A] = \int_{x=0}^{r} A(x)f(x)dx \tag{2}$$

The probability distribution function of x, when node p and c are uniformly distributed in the deployment area, is as follow:

$$F(x) = P(distance < x) = \frac{\pi x^2}{\pi r^2} = \frac{x^2}{r^2} \tag{3}$$

Then, the probability density function of x is given by:

$$f(x) = F'(x) = \frac{2x}{r^2} \tag{4}$$

Then:

$$E[A] = \int_{x=0}^{r} \left(2r^2 cos^{-1}(\frac{x}{2r}) - x\sqrt{r^2 - \frac{x^2}{4}} \right) \frac{2x}{r^2}\ dx \tag{5}$$

$$E[A] = \left(\pi - \frac{3\sqrt{3}}{4} \right) r^2 = 1.84255\ r^2 \tag{6}$$

The average number of common neighbors C_N, within the region A, is given by $C_N = E[A] \times d$, where d is the network density.

Let P_{comp} be the probability of compromising a node, and we define P_{attack} as the probability that an attacker forges a fake RREQ without being detected. The adversary needs to compromise all the common neighbors in A tu succeed at launching its attack, then:

$$P_{attack} = (P_{comp})^{C_N} \tag{7}$$

As depicted in Fig. 3, our protocol is effective when the network density increases. The probability for an attacker to succeed at launching a forged hop count is low, especially when the compromising probability is low. For example, for an average network density of 5, the probability of successful attack does not exceed 0.15, even if the compromising probability is hight (0.8).

5.4 Wormhole immunity

A wormhole attack occurs when a malicious node forwards incoming packets to a distant point of the network by means of a fast link longer than a normal node's range. Wormhole attacks can cause severe damage to hop-count-based routing protocols, since the attacker can present to distant nodes an attractive

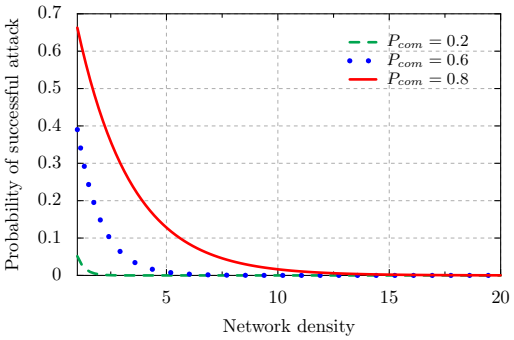


Fig. 3. The probability of successful hop count forgery attack.

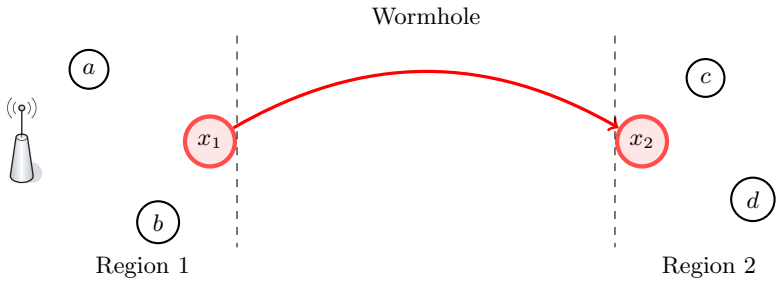


Fig. 4. Defense against a wormhole attack.

path, which other uninfected paths cannot compete. If such a scenario occurs, the attacker will be advantageous compared to other nodes as a large amount of packets will pass through him.

These attacks are challenging because attackers may relay the packets without any malicious modification. Existing secure routing protocols for WSN, such as SeRINS and SEIF, are vulnerable to wormholes. In contrast, SMART can in some cases defeat this attack thanks to its two-hop key distribution scheme. Basically, this is due to the fact that these keys represent *locality proofs* that can be used to detect false neighborhood links introduced by a wormhole.

To clarify how these locality proofs can be used to detect wormholes, let us consider the example in Fig. 4. Attacker x_1 tunnels its RREQ messages to x_2. To propagate this message, the attackers must ensure that nodes at different endpoints of the tunnel share two-hops broadcast keys to make the wormhole looks like a normal single hop. Here, we can distinguish two cases:

1. If the nodes were deployed before the compromise of x_1 and x_2, we can show that such keys cannot be established. Indeed, only a node with K_{IN} can

create an authentic two-hop link between already deployed nodes. Therefore, when x_1 and x_2 become compromised, they are necessary unaware of K_{IN} and thus the keys between nodes in Regions 1 and 2 cannot be established through the tunnel. Consequently, exisitng keys will play the role of locality proofs to detect the far distance between Region 1 and 2, and hence preventing the tunnel to be viewed as a normal single hop.

2. If a node is newly deployed in the vicinity of the attackers, keys between nodes in different tunnel regions can be maliciously created. For example, suppose that node b is deployed after wormhole creation. Attacker x_1 can tunnel the key exchange messages of b to Region 2. Therefore, nodes at this endpoint will consider b as a two-hop neighbor through the wormhole link. The attackers can therefore relay consistent RREQ to connect the two regions and without being detected.

5.5 Energy consumption

In this section, we evaluate the performance of SMART and compare it to SeRINS as both aim to secure the hop count information. Also, in a survey about secure multi-path routing in WSNs [6], SeRINS is the only one so far, which ensures the following security properties: authentication, integrity, confidentiality, freshness, and accountability. It is easy to show that these security properties are also ensured by SMART. Both protocols SMART and SeRINs have been implemented in TinyOS. To evaluate energy consumption, we have used Avrora [12] that emulates and analyzes programs written for AVR microcontroller, which is produced by Atmel and used in MICAz sensor mote.

The energy consumption determines the network lifetime and must be considered when designing protocols for WSNs. For this reason, we have measured the average energy consumption of SeRINS and SMART during one round while varying the number of intruders in the network, as shown in Fig. 5. In our simulation scenario, the total number of nodes in the network is set to 300 nodes. We can notice that SMART does not consume an additional energy when increasing the number of intruders in the network. However, the energy consumption in SeRINS increases as the number of intruders increases. This is due to the number of alerts sent to the sink for each intruder and from each intruder's neighbor.

6 Conclusion

In this paper we have proposed a security framework composed of multi-path routing protocol (SMART) and two-hop key management scheme (ETKE). The proposed framework keeps consistent routing topology by protecting the hop count information from being forged. The two-hop verification and the watch-dog mechanisms ensure a fast detection of inconsistent routing information without referring to the Sink node. The security analysis have shown that the probability of successful attack is very low under medium and high network densities. In addition, simulation experiments have shown that SMART is more energy-efficient compared to SeRINS.

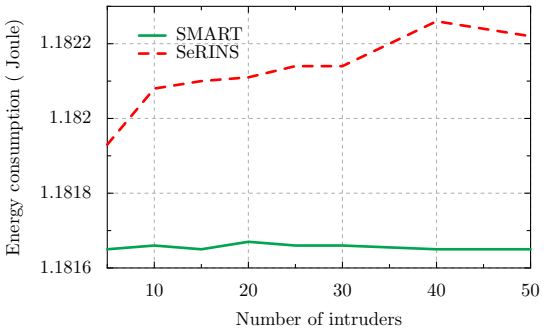


Fig. 5. Energy consumption vs. number of intruders in the network.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks **38**(4) (2002) 393 – 422
2. Lee, S., Choi, Y.: A secure alternate path routing in sensor networks. Computer Communications **30**(1) (2006) 153–165
3. Challal, Y., Ouadjaout, A., Lasla, N., Bagaa, M., Hadjidj, A.: Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks. Journal of Network and Computer Applications **34**(4) (2011) 1380–1397
4. Ghosal, A., Halder, S.: Intrusion detection in wireless sensor networks: Issues, challenges and approaches. In: Wireless Networks and Security. Signals and Communication Technology. (2013) 329–367
5. Dimitriou, T.: Securing communication trees in sensor networks. In: ALGOSENSORS. (2006) 47–58
6. Stavrou, E., Pitsillides, A.: A survey on secure multipath routing protocols in wsns. Computer Networks **54**(13) (2010) 2215 – 2238
7. Bagaa, M., Challal, Y., Ouadjaout, A., Lasla, N., Badache, N.: Efficient data aggregation with in-network integrity control for wsn. J. Parallel Distrib. Comput. **72**(10) (2012) 1157–1170
8. Chen, C.Y., Chao, H.C.: A survey of key distribution in wireless sensor networks. Security and Communication Networks (2011)
9. Cheng, Y., Agrawal, D.P.: An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. Ad Hoc Networks **5**(1) (2007) 35–48
10. Deng, J., Hartung, C., Han, R., Mishra, S.: A practical study of transitory master key establishment for wireless sensor networks. In: Proc First IEEE Int'l Conf Security and Privacy for Emerging Areas in Comm. Networks (SecureComm '05). (2005)
11. Hai, T.H., nam Huh, E., Jo, M.: A lightweight intrusion detection framework for wireless sensor networks. Wireless Communications and Mobile Computing **10**(4) (2010) 559–572
12. Titzer, B., Lee, D.K., Palsberg, J.: Avrora: scalable sensor network simulation with precise timing. In: Proceedings of the 4th International Symposium on Information Processing in sensor Networks (IPSN). (2005) 477–482