Efficient Flooding in Wireless Sensor Networks Secured with Neighborhood Keys

Amin Hassanzadeh, Radu Stoleru, Jianer Chen
Department of Computer Science and Engineering, Texas A&M University
{hassanzadeh, stoleru, chen}@cse.tamu.edu

***Abstract*—Network flooding is a fundamental communication primitive for Wireless Sensor Networks (WSN). Flooding is used for disseminating code updates and parameter changes, affecting the operation of all nodes in the network. When flooding occurs each node, typically, broadcasts the flooding packet once. The costs for flooding, however, can become significant if neighborhood keys are used for communication (as proposed in recent research on secure localization and key distribution [1]), since, instead of a single broadcast, a node is required to perform several unicast transmissions. In this paper we address the problem of minimizing the number of unicast transmissions required for ensuring 100% network coverage for flooding in WSN secured with neighborhood keys. We show that the problem is NP-hard and propose an approximation algorithm for solving it. Through simulations, we demonstrate that our algorithm ensures 100% network coverage for flooding, while requiring, surprisingly, as low as 0.75 packet transmissions per node.**

I. Introduction

Wireless Sensor Networks (WSN), consisting of large numbers of resource constrained (e.g., energy, storage, computation, communication) sensor nodes, have recently been developed for several applications [2]–[4]. Securing WSN, especially those targeting military, industrial and civil applications, has long been a top priority, evidenced by research in several areas, including key distribution [5], secure communication [6], and secure localization [7], to name a few.

As a solution for both secure node localization *and* key distribution in WSN, Secure Walking GPS (S-WGPS) was recently proposed [1]. S-WGPS, an extension of Walking GPS (WGPS) [8], is a practical localization scheme and secure against *Dolev-Yao* and *GPS-denial* attacks. In both WGPS and S-WGPS, a GPS-enabled master node obtains its current location from the onboard GPS or Inertial Management Unit (IMU) and sends it to each newly deployed node, which must be in its proximity. For secure neighborhood communication S-WGPS distributes pairwise, neighborhood keys to the nodes being deployed. The neighborhood keys ensure network resilience against *Wormhole* attacks.

The decision regarding what keys to distribute to a sensor node, during deployment, resides with the GPS-enabled master node. The master node estimates the location of the node being deployed (based on master's location) and compares it with locations of the already deployed nodes. It then decides what keys to distribute to the newly deployed sensor node, such that the node shares a key with at least one of its neighbors. It is paramount to remark that: i) S-WGPS, to ensure network resilience against wormhole attacks, forbids the sharing of a key between a node and all its neighbors, unless all node' neighbors are also neighbors (i.e., a clique); and ii) for limiting the number of keys deployed on each node (because of storage constraints) S-WGPS does not guarantee that a key is shared by any two sensor nodes within radio range. Consequently, it is possible to have true sensor node neighbors unable to communicate because they do not share a key.

Since the keys distributed by S-WGPS can only be used for unicast communication, supporting broadcast can become very costly in terms of energy. In order to broadcast a packet, a node provisioned with neighborhood keys, would need to send multiple unicast packets. Motivated by the high cost of flooding packets from a basestation in a network secured with neighborhood keys, we seek to investigate how to reduce the number of unicast transmissions in order to support network flooding. The first research question we address is whether or not there exists a solution for the network flooding problem, in which each node uses only one key for re-broadcasting a packet[1], such that, eventually, all nodes in the network receive the packet. It is critical to understand that a node might not be able to directly communicate with all its neighbors (because they have different keys), but they may be able to communicate, indirectly, through some shared neighbors, or neighbors of neighbors, etc. The second research question we plan to investigate is: if it is not possible to support networking flooding with one packet transmission per node, then what is the optimal number of unicast transmissions that ensure 100% coverage for the flooding packet?

To address the aforementioned research questions, we formally define the problem of efficient network flooding (i.e., that uses a single unicast transmission by each node) in WSN secured with neighborhood keys. We prove that the problem is NP-complete, propose an optimization version of it, and develop an approximation algorithm for solving it. In our proposed solution, the master node employed by S-WGPS decides the keys that each node in the network needs to use for flooding packets from a given basestation (we call this *Node/Keys* mapping file). Interestingly, after this decision is made, it is flooded, from the basestation to the entire network efficiently. Each node receiving the Node/Keys packet, uses the information from the packet to decide which of its keys to use for re-broadcasting. This solution will guarantee that all

[1]This means the node sends only one flooding packet, as it would have been the case for broadcast-based flooding

978-1-4577-2014-7/11/$26.00

nodes in the network receive the Node/Keys mapping file for efficiently supporting network flooding. More precisely, this paper makes the following contributions:

- We formulate the efficient flooding in WSN secured with neighborhood keys as an optimization problem and show that it is NP-hard.
- We propose an approximation algorithm which selects an optimal subset of keys that each node needs to use for flooding a packet, ensuring a 100% network coverage.
- Through extensive simulations we demonstrate that our algorithm achieves 100% network coverage for flooding, while requiring, surprisingly, on average less than 1 packet transmitted per node.

This paper is organized as follows. We present background material and review the state of art in Section II. We formulate the problem of efficient flooding in WSN secured with neighborhood keys, and prove that it is NP-hard in Section III. We show how to formulate our problem as a satisfiability problem in Section IV, and present an approximation algorithm for solving it, in Section V. Performance evaluation results for our proposed algorithm are presented in Section VI. We conclude in Section VII summarizing the main ideas and contributions.

II. State of Art and Background

Public key cryptography has long been considered very difficult for resource constrained sensor nodes. Despite successful implementations on more capable sensor node hardware, e.g., iMote [9], [10], most of the proposed encryption algorithms for typical sensor networks are still based on symmetric cryptography. Related to secure flooding/broadcasting, two approaches have been proposed for broadcast authentication in wireless sensor networks: i) digital signatures; and ii) μTESLA-based algorithms [11].

In solutions based on digital signatures, receiver nodes authenticate received broadcast messages by verifying the senders' signature attached to the message. The challenging aspect of these types of solutions remains the high number of signature operations needed, despite attempts at reducing them, as in [12] where an authentication expander graph with constant degree reduces the number of hashes to be included in the message. Similarly, other ideas to reduce the number of unnecessary signature verifications have been proposed [13].

The μTESLA-based algorithms, based on TESLA [14] and its variations [15]–[17], use symmetric cryptography mechanisms. A major limitation for μTESLA is the fact that it requires the basestation and sensor nodes to be time synchronized. Additionally, most of the algorithms in this class lack the capability of immediate authentication, making them vulnerable to DoS attacks against broadcast authentication. Ning et. al [11] address some of these limitations through a technique that mitigates DoS attacks against both signature-based and μTESLA-based algorithms.

Compared with state of art, our proposed solution requires neither time synchronization nor additional information attached to the broadcast messages (eliminating the need for expensive signature-related operations on each packet). The only cost for our solution is the one time flooding that distributes to all sensor nodes in the network information about the keys they need to use (the nodes already have the keys) for network flooding.

A. Support for Broadcast in S-WGPS

For secure neighborhood communication, S-WGPS distributes pairwise, neighborhood keys to the nodes being deployed. The neighborhood keys ensure network resilience against wormhole attacks. For assigning keys to nodes, S-WGPS follows two rules [1]:

Definition 1: **Distance Bounding Rule:** *Only physical neighbors are allowed to share communication keys; this helps nodes resist the Wormhole attack since no two nodes far apart will share a key.*

Definition 2: **Connectivity Rule:** *Each node must share a key with at least one physical neighbor which is already deployed; this provides neighbor connectivity for all nodes.*

Despite S-WGPS attempting to allow nodes to securely communicate with as many neighbors as possible, it is important to note that S-WGPS may cause a true physical link between two neighbor nodes to become invalid/unusable, if the nodes do not posses a common key.

Due to very limited storage available to sensor nodes, key distribution algorithms need to limit the number of keys they store and use, while continuing to fulfill performance and security requirements. Certainly there are tradeoffs. On one hand, if we distribute a small number of keys to each node, it might be difficult to ensure that aforementioned rules are satisfied. On the other hand, a large number of keys distributed to each node would waste precious storage resources. Interestingly, Mi et al. proved that the minimum number of communication keys (m_{min}) that need to be assigned to a sensor node has a close relation to the maximum number of neighbors (N) of a node, as follows [1]:

$$m_{min} = \begin{cases} 5 & if \quad N \geq 6 \\ N & otherwise \end{cases}$$

Despite providing a simple and practical key distribution mechanism for sensor networks, S-WGPS lacks an inexpensive secure message broadcasting mechanism. Considering Definition 1, a node never shares the same key with two neighbors that are not physically connected to each other. Hence, S-WGPS, does not allow a node to broadcast an encrypted message to all its neighbors with only one transmission, unless all the neighbors are physically connected to each others (i.e., set of neighbors plus source node make a clique and every two nodes are physically connected).

An example of how keys are distributed by S-WGPS is depicted in Figure 1(a), where nodes 1 through 6 are equipped with different sets of keys (e.g., node 3 has keys $\{k_1, k_2, ...\}$). Depending on what key a node uses, an encrypted broadcast message is received[2] by: i) all neighbors (BRDCST); ii)

[2]From here on the term "receive" implies that a node can decrypt the received message.

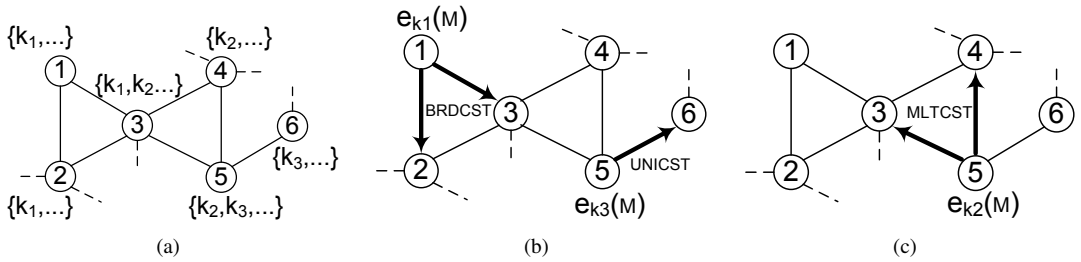

Fig. 1. Example of keys distributed by S-WGPS. a) Network connectivity and key distribution. b) The broadcast message sent by node 1 can be decrypted by all its neighbors (BRDCST) while only node 6 can decrypt the message sent by node 5 (UNICST). c) A subset of neighbors can decrypt the message sent by node 5 (MLTCST).

some of neighbors (MLTCST); or iii) one of the neighbors (UNICST) (Definition 2 guarantees that at least one of the neighbors will receive the broadcast message). As shown in Figure 1(b), a broadcast message, encrypted and sent by node 1 using k_1 is received by all its neighbors. A broadcast message sent by node 5, however, will be received either by node 6, using key k_3 (as shown in Figure 1(b)) or nodes 3 and 4, using key k_2 (as shown in Figure 1(c)). Therefore, for broadcasts by node 5, not all its neighbors can receive the message, unless it is sent twice, encrypted with two different keys. The same inefficient broadcast occurs for node 3.

III. Preliminaries and Problem Formulation

We model a WSN as a graph $G = (V, E)$ with vertices $V = \{v_1, v_2, ..., v_n\}$ representing nodes, and edges $E = \{e_1, e_2, ..., e_h\}$ representing communication links. In the WSN there is a basestation b. During node deployment, the S-WGPS scheme is used for localizing nodes and for distributing keys from a set of keys $K = \{k_1, k_2, ..., k_m\}$, to nodes. We denote the set of keys assigned to a node v_i by K^i, where $K^i \subseteq K$ and $\cup_{i=1}^{n} K^i = K$.

We remark here again, that it is possible for two nodes to be true neighbors (i.e., the distance between them is smaller than the communication range), and still not be able to communicate with each other, because they do not share a key. To ensure a connected network, however, each node v_i has at least one neighbor it can communicate with (i.e., $|K^i| \geq 1, \forall v_i$) using communication key k_l assigned to both. Moreover, nodes v_i and v_j, where edge $(v_i, v_j) \in E$, cannot securely communicate unless there is at least one k_l such that $k_l \in K^i \cap K^j$. Therefore, we define E_s, called the *Set of Secure Edges*, as the subset of all edges ($E_s \subseteq E$) where for any $e_i \in E_s$ there is at least one k_l assigned to both end points of that edge. The edges of the graph shown in Figure 1(a) represent precisely E_s. While in Figure 1 there may be nodes that are within radio range of each other (say nodes 3 and 6), if they do not share the same key, they will not be able to communicate directly, and the link between them is excluded from E_s.

A. Problem Formulation

Definition 3: **Secure Flooding in WSN with Neighborhood Keys Problem (SFNK):** *Given a graph G and a set of keys K, find a set $K' \subseteq K$ and* mapping $L = \{(v_i, k_j) | (k_j \in K') \wedge (k_j$ selected by node $v_i)\}$*, such that: i) each node v_i has at most one key in set K' for rebroadcasting a flooding packet; and ii) the set $E'_s \subseteq E$, obtained by the selection of set K', creates a connected graph that spans all the vertices in $V \cup \{b\}$ (*i.e., ensure complete network coverage*).*

The above problem formulation means that we need to obtain the mapping L of keys to nodes, such that each node has at most one key for rebroadcasting a message. The basestation initiates a flood in the network. A node receives a message, it encrypts it with its key, and rebroadcasts it. E'_s must create a *connected* graph to ensure complete *Network Coverage* for the flooding message. From here on, we will use the term "*node covered*" when a node can successfully receive the flooding message. *A network is covered when all its nodes are covered.*

A solution for the SNFK problem shown in Figure 1, with node 1 as basestation, is the mapping $L = \{(1, k_1), (2, \phi), (3, k_2), (4, \phi), (5, k_3), (6, \phi)\}$ (we use ϕ to indicate that a node does not necessarily need to rebroadcast the flooding message). The mapping $L = \{(1, k_1), (2, \phi), (3, k_1), (4, k_2), (5, k_3), (6, \phi)\}$, however, produces a disconnected graph, an unacceptable solution. In this case, node 4 never receives the broadcast messages originated from the basestation since the key k_1 selected by node 3 *does not cover* nodes 4 and 5.

B. SFNK Hardness

In this section, we first argue that SFNK is different than other similar problems in graph theory, then prove SFNK hardness. The Hamiltonian Path Problem (HPP), a known NP-complete problem, seeks to find a path in an undirected graph such that each vertex is visited exactly once [18]. One might remark the similarity of our SFNK problem wit HPP. As described before, SFNK allows a node to send the broadcast message to more than one neighbor at the same time (if they all share the same key). For HPP, however, this is not a valid case since each node has to choose only one neighbor as the next hop in the path.

A graph that has a Hamiltonian cycle is called Hamiltonian graph, e.g., complete graphs and cycle graphs are Hamiltonian graphs. If a graph has Hamiltonian path, then it has a solution to the SFNK problem; however, there are graphs that do not have a Hamiltonian path, but do have an SFNK solution.

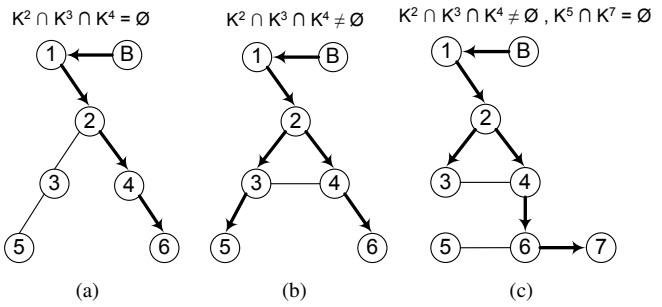

Fig. 2. Comparison of SFNK and HPP. a) There is no solution for SFNK or HPP, since node 2 has to choose either node 3 or node 4 as the next hop. b) SFNK has a solution because of the availability of a key shared by nodes 2, 3, and 4 that makes a triangle. c) There is no solution for SFNK and HPP, although the graph has a triangle.

Obviously, it is possible that a graph has no Hamiltonian path and no SFNK solution. Figure 2(a), depicts an example of a graph that has neither Hamiltonian path nor SFNK solution because node 2 splits the graph in two subgraphs that can never be visited by a single path. Considering Definition 1, nodes 3 and 4 cannot share the same key with node 2, since they are not physically connected.

Figure 2(b) depicts the same graph as in Figure 2(a) with a connection between vertices 3 and 4. We mentioned that any physical connection between two nodes does not necessarily mean that they share a key. We assume, however, that the graph edges in Figure 2 belong to E_s. In this example, node 2 can choose the key shared by nodes 3 and 4, so that they can decrypt the transmitted message sent by node 2 exactly once. This is because any triangle in graph $G = (V, E_s)$ guarantees that only one transmission is necessary to send the message to both neighbors. Nevertheless, having a triangle in a graph does not necessarily mean that our problem has a solution, as shown in Figure 2(c).

Theorem 1: SFNK in NP-complete.

Proof 1: The input to SFNK is a graph that can be triangle-free or non triangle-free. For a triangle-free input graph, SFNK is the same as HPP since none of the nodes shares a key with more than one neighbor. Hence, HPP is a special case of SFNK. Consequently, regardless the input graph, SFNK is NP-complete unless $P \neq NP$. □

Similar to other NP-complete problems, it is possible to have inputs to SFNK for which there is no solution. In other words, any key selection K' and corresponding mapping L for such an input is a *no-instance* in the decision version of the SFNK problem. Consequently, we propose to reduce our SFNK problem, in polynomial time, to an optimization problem (MAX-SFNK) as follows:

Definition 4: **MAX-SFNK:** *Given a graph G and a set of keys K, find a set $K' \subseteq K$ and* mapping $L = \{(v_i, k_j) | (k_j \in K') \wedge (k_j$ selected by node $v_i)\}$*, such that each node v_i has the minimum number of keys in set K' for rebroadcasting a flooding message, and the corresponding set E'_s creates a connected graph spanning all vertices in $V \cup \{b\}$.*

This means there may be nodes that have to rebroadcast the flooding message more than once. We propose to solve MAX-SFNK using approximation algorithms, as will be described in Section V. Before proceedings with approximation algorithms, we formulate the SFNK problem as a Satisfiability (SAT) problem because SAT, due to the attention it has received, has good approximation algorithms.

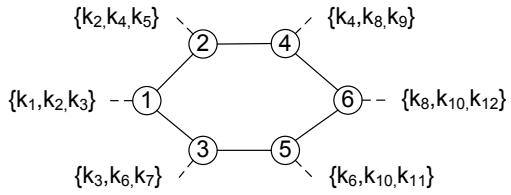

Fig. 3. Graph example for posing SFNK as a 3-SAT problem.

IV. SFNK IN SAT FORMAT

For SAT, the input is a set $\mathcal{F} = \{C_1, C_2, ..., C_n\}$ of *clauses*, where each clause contains boolean variables $X = \{x_1, x_2, ..., x_m\}$ as either x_i or its negation $\overline{x_i}$ *literal*. A truth assignment to boolean variables satisfies a clause if the assignment makes the disjunction of the literals of the clause TRUE.

For expressing SFNK in SAT format, let clause C_i represent the fact that node v_i is covered and that it uses a single rebroadcast for flooding a message, from a given basestation, in the network.

$$C_i = \begin{cases} TRUE & \text{if } v_i \text{ is covered, and rebroadcasts once} \\ FALSE & \text{otherwise} \end{cases}$$

Let boolean variable x_j correspond to key k_j such that x_j is TRUE if k_j is selected by a node, otherwise x_j is FALSE. For a node v_i and its corresponding key set $K^i = \{x_1, x_2, ..., x_q\}$, C_i is a clause of boolean variables x_j.

Flooding aims to transmit a message to all nodes in the network, while each node selects at most one key for re-broadcasting the message. If we take into account that before transmitting a message a key was needed for decrypting it, then each node needs two keys for supporting network flooding. If a node uses anything different than two keys, then the node either has not sent the received message or it has sent it more than once. Hence, *network coverage requires that each node uses exactly two keys*, one for receiving and one for transmitting a flooding packet.

Therefore, C_i is satisfied if and only if exactly two boolean variables are TRUE and all other variables are FALSE. This results in C_i being a disjunction of smaller clauses, which themselves are conjunctions of literals. Consequently, SFNK will have a 3-level format (note: the SAT problem has a standard 2-level format since it is a conjunction of clauses (first level), where each clause is a disjunction of literals (second level)). To represent the SFNK problem in a standard SAT format, we use De Morgan's law and rewrite each clause in a conjunctive normal form (CNF) so that function F becomes a CNF as well.

Example: For clarity of presentation we show how SFNK is formulated as a 3-SAT problem on the input graph given in Figure 3. This formulation, without loss of generality, is valid for any k-SAT problem where k is the maximum number of keys assigned to each node in SFNK. Figure 3 shows a typical example of a graph with at least 3 neighbors (and also 3 keys) for each node. The clause for node 1 is $C_1 = (x_1 \wedge x_2 \wedge \overline{x_3}) \vee (x_1 \wedge \overline{x_2} \wedge x_3) \vee (\overline{x_1} \wedge x_2 \wedge x_3)$. This means C_1 is satisfied if and only if exactly two keys are selected (the same applies to all C_is).

By applying the same boolean algebra to other clauses, the counterpart of the satisfiability function is $F = C_1 \wedge C_2 \wedge ... \wedge C_n$, i.e., F is satisfied if and only if all clauses are satisfied:

$$
\begin{aligned}
F &= C_1 \wedge C_2 \wedge ... \wedge C_n \\
&= \left(\underbrace{(x_1 \wedge x_2 \wedge \overline{x_3}) \vee (x_1 \wedge \overline{x_2} \wedge x_3) \vee (\overline{x_1} \wedge x_2 \wedge x_3)}_{C_1} \right) \\
&\wedge \left(\underbrace{(x_2 \wedge x_4 \wedge \overline{x_5}) \vee (x_2 \wedge \overline{x_4} \wedge x_5) \vee (\overline{x_2} \wedge x_4 \wedge x_5)}_{C_2} \right) \\
&\vdots \\
&\wedge \left(\underbrace{(x_{m-2} \wedge x_{m-1} \wedge \overline{x_m}) \vee (x_{m-2} \wedge \overline{x_{m-1}} \wedge x_m)}_{C_n} \right. \\
&\qquad \left. \underbrace{\vee (\overline{x_{m-2}} \wedge x_{m-1} \wedge x_m)}_{C_n} \right)
\end{aligned}
$$

Let H be the complement of F, i.e., H is TRUE if and only if F is FALSE and vice versa. Therefore, $H = \overline{F}$ represents any key selection that does not cover all nodes. The boolean function H is then: $H = \overline{F} = \overline{C_1 \wedge C_2 \wedge ... \wedge C_n} = \bigvee_{i=1}^{n} \overline{C_i}$. Now, let $U_i = \overline{C_i}$ denote any selection that does not cover node v_i. Considering: i) our definition of C_i, for a 3-key node (i.e., $C_i = \Sigma m(3,5,6)$ - summation of all minterms that have exactly two variables which appeared as x_i, while others as negations); and ii) the De Morgan's law, U_i for any node with 3-keys is: $U_i = \Sigma m(0,1,2,4,7)$.

$$
\begin{aligned}
U_1 = \overline{C_1} &= \Sigma m(0,1,2,4,7) \\
&= \underbrace{(\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3})}_{m_0} \vee \underbrace{(\overline{x_1} \wedge \overline{x_2} \wedge x_3)}_{m_1} \\
&\vee \underbrace{(\overline{x_1} \wedge x_2 \wedge \overline{x_3})}_{m_2} \vee \underbrace{(x_1 \wedge \overline{x_2} \wedge \overline{x_3})}_{m_4} \vee \underbrace{(x_1 \wedge x_2 \wedge x_3)}_{m_7}
\end{aligned}
$$

Therefore, $H = \bigvee_{i=1}^{n} U_i$. Now we convert F into a CNF using De Morgan's law. Therefore, the produced expression is in standard SAT format, i.e., conjunction of clauses where each clause is a disjunction of boolean variables.

$$
\begin{aligned}
F = \overline{H} &= \overline{\bigvee_{i=1}^{n} U_i} = \bigwedge_{i=1}^{n} \overline{U_i} = \overline{U_1} \wedge \overline{U_2} \wedge ... \wedge \overline{U_n} \\
&= \underbrace{(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)}_{\overline{U_1}} \\
&\wedge \underbrace{(\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})}_{\overline{U_1}} \\
&\wedge \underbrace{(x_2 \vee x_4 \vee x_5) \wedge (x_2 \vee x_4 \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_4} \vee x_5)}_{\overline{U_2}} \\
&\wedge \underbrace{(\overline{x_2} \vee x_4 \vee x_5) \wedge (\overline{x_2} \vee \overline{x_4} \vee \overline{x_5})}_{\overline{U_2}} \qquad (1) \\
&\vdots \\
&\wedge \underbrace{(x_{m-2} \vee x_{m-1} \vee x_m) \wedge (x_{m-2} \vee x_{m-1} \vee \overline{x_m})}_{\overline{U_n}} \\
&\wedge \underbrace{(x_{m-2} \vee \overline{x_{m-1}} \vee x_m) \wedge (\overline{x_{m-2}} \vee x_{m-1} \vee x_m)}_{\overline{U_n}} \\
&\wedge \underbrace{(\overline{x_{m-2}} \vee \overline{x_{m-1}} \vee \overline{x_m})}_{\overline{U_n}}
\end{aligned}
$$

The expression shown in Equation 1 is in SAT format, i.e., conjunction of clauses, where each clause is a disjunction of boolean variables. One can observe that the number of clauses in SFNK significantly increases with the number of keys a node has. A higher number of keys, however, is because of a higher number of neighbors (i.e., higher node density). While the larger number of clauses might make one think that the produced expression becomes more sophisticated, it is paramount to observe that higher node densities result in higher node coverage (as proven in the theorem below), which, in turn, result in a shorter execution time for the algorithm that solves the satisfiability problem, as will be described in Section V.

Theorem 2: The likelihood of SFNK covering all nodes is higher for networks with higher node density.

Proof 2: Let N_a be the average number of neighbors of a node, and N_M the maximum number of neighbors of a node in a given *connected* network of size n. We consider the range for N_a: i) if $N_a = 1$ then SFNK has a trivial solution; ii) if $N_a < 2$ and $N_M \leq 2$ (for linear and circular topologies there always exists a solution). If $N_a < 2$ and $N_M > 2$ there is no solution because there is no loop and no triangle in the network (e.g., star topology). This means that there is at least one node that has to send the flooding message more than once; iii) if $N_a > 2$, then the graph has at least one loop, which increases the number of nodes covered with SFNK. The larger N_a is, the higher probability of having loops and triangles in the network is; iv) if $N_a = n$, then the complete graph requires only one transmission for covering all the nodes. Consequently, for larger N_a, more nodes will be covered. □

V. MAX-SFNK Approximation Algorithm

Our proposed approximation algorithm for SFNK is based on Johnson's [19], an approximation algorithm for MAX-SAT that runs in polynomial time.

First, we introduce notations used in our proposed approximation algorithm, depicted in Algorithm 1. Let φ_j be the number of nodes that share corresponding key k_j. We denote by N_i^j the set of neighbors of node v_i that share key k_j, and by K^{BS} the set of keys assigned to the basestation for communicating with sensor nodes within radio range. Let s_i denote the number of times a node has received the flooding packet (i.e., number of keys used for receiving). Our proposed approximation algorithm, has input sets V, E_s, K and outputs the mapping L of nodes and keys used. For clarity of our presentation, we will also use an example graph, depicted in Figure 4.

Given an input graph and a set of distributed keys, the algorithm first creates all clauses for corresponding keys (Line 1). The mapping L is initially empty as shown in Line 2. Line 3 selects the most frequent key from the basestation and sets the corresponding boolean variable to TRUE. For the example depicted in Figure 4, the assignment in Line 3 is represented as $(0 : x_2)$ which means that the truth assignment at time 0 for k_2 is $\tau(x_2) = TRUE$. Then, Line 4 adds the first pair (b, k_2) to the mapping L. Since this key is used for transmission from the basestation, we set $s_i = 1$ for the corresponding receivers (Line 5), i.e., they can participate in the rebroadcasting process. Line 6 makes all other keys shared between basestation and the other nodes FALSE, so that its neighbors have to select other keys for transmitting the message, upon receiving it. This intuitively increases the chance of sending message towards farther nodes.

Next, similar to Johnson's algorithm, our algorithm assigns a weight to each clause based on the number of variables in each clause (Line 7). This weighting mechanism helps the algorithm to assign a truth value to a variable that satisfies more clauses. Then, the truth assignment for any unassigned variable will be executed iteratively. Our approach in assigning truth values to the boolean variables is different from Johnson's algorithm since we go over all boolean variables $\{x_1, x_2, ..., x_m\}$ not by their index, but by their sorted sequence. We sort all nodes in non-decreasing s_i, for $s_i \geq 1$ (if they are equal, the recently updated is first). Then, we update the sequence of the corresponding boolean variables that have unassigned truth values. This update is performed in the same order with that of non-decreasing s_i order. We update this list at each iteration of our algorithm before we pick the next variable for truth assignment. This order avoids selecting k_j (i.e., $\tau(x_j) = TRUE$) from node v_i unless it has received the message before. Moreover, it guarantees that E_s' is always *connected*. In the first iteration, line 9 sorts the list of covered nodes based on s_i, that is $\{2, 3\}$ in Figure 4. Thus, the next variable to be assigned is a variable corresponding to the keys in K^2. Then, for any variable x_j, our algorithm compares the weight of all clauses that have x_i with those that have $\overline{x_j}$ and

Algorithm 1 MAX-SFNK (**Input**: V, E_s, K, **Output**: L)

1: create set $\mathcal{F} = \{C_1, ..., C_p\}$ on $\{x_1, ..., x_m\}$ from sets V, E_s, K using Equation 1
2: $L = \emptyset$
3: find Max$\{\varphi_j$ for $\forall k_j \in K^{BS}\}$ and set $\tau(x_j) = TRUE$
4: $L = L \cup \{(b, k_j)\}$
5: $s_i = 1$ for $\forall v_i \in N_{BS}^j$
6: set $\tau(x_l) = FALSE$ for $\forall x_l \in K^{BS}$ where $j \neq l$
7: set $w(C_u) = 1/2^{|C_u|}$ for each clause C_u
8: **while** there is any unassigned x_j **do**
9: sort list of node in a nondecreasing order based on s_i for $\forall v_i$ that $s_i \geq 1$
10: update the sequence of X for unassigned x_j
11: pick x_j with Max$\{\varphi_j\}$ corresponding to node i from the head of the list
12: find all clauses $C_1^T, ..., C_a^T$ in $\mathcal{F}$ that contain x_j
13: find all clauses $C_1^F, ..., C_b^F$ in $\mathcal{F}$ that contain $\overline{x_j}$
14: **if** $\Sigma_{u=1}^{a} w(C_u^T) \geq \Sigma_{u=1}^{b} w(C_u^F)$ **then**
15: $\tau(x_j) = TRUE$ and delete $C_1^T, ..., C_a^T$ from $\mathcal{F}$
16: $L = L \cup \{(i, k_j)\}$
17: update s_r for $\forall v_r \in N_i^j$
18: **for** $u = 1$ **to** b **do** $w(C_u^F) = 2w(C_u^F)$
19: **else**
20: $\tau(x_j) = FALSE$ and delete $C_1^F, ..., C_b^F$ from $\mathcal{F}$
21: **for** $u = 1$ **to** a **do** $w(C_u^T) = 2w(C_u^T)$
22: **end if**
23: **end while**

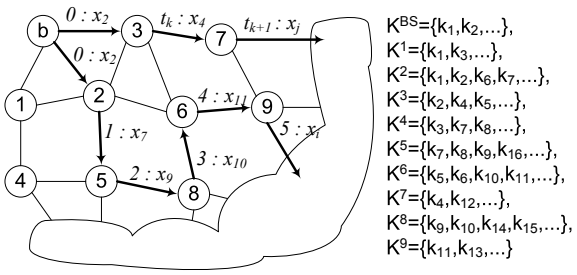

Fig. 4. An example of truth assignment by MAX-SFNK

assigns a truth value that satisfies a heavier set of clauses in which such a variable appears in. It is shown in Figure 4 that x_7 becomes TRUE. Similar to Lines 4 and 5, the mapping L and all variables s_i will be updated in Lines 16 and 17 if a new key is selected. After any truth assignment (i.e., Lines 15 and 20), satisfied clauses are removed from set $\mathcal{F}$.

Lines 18 and 21, similar to Johnson's algorithm, double the weight of clauses that contain x_j which were not satisfied by the decision made in Line 14. The double weight increases the chances of those clauses to be satisfied in the following iterations. Another important difference between our algorithm and Johnson's is that a TRUE assignment in our algorithm means that a message rebroadcasting with key k_j will occur. Thus, all nodes using this key are considered *covered*, and the algorithm increments their corresponding s_i. Our algorithm extends the list of covered nodes in the next iteration which affects Line 9. Considering the example in Figure 4, when node 6 is added to the list of covered nodes and a corresponding variable is taken to be assigned, most of its clauses that contain x_6 (since k_6 is shared by nodes 2 and 6) are already satisfied when we

set x_2 and x_7 as TRUE (based on Equation 1) and surely the weight of those containing $\overline{x_6}$ is much higher. This causes $\tau(x_6) =$ FALSE and increases the chance of another key of node 6 (say k_{11}) to be TRUE. One can observe that MAX-SFNK covers nodes in a depth first manner. Considering the example in Figure 4, our algorithm covers nodes 2, 3, 5, 8, 6, 9, ..., in order, until no new keys can be assigned to the last node. At that time, the algorithm continues from node 3, since $s_3 = 1$.

A. Algorithm Analysis

The approximation ratio for Johnson's algorithm is based on the number of clauses it can satisfy; however, in MAX-SFNK we aim to cover as many nodes as possible. We denote the upper bound for the number of keys per node by q (q is analogous with network density - higher node density will result in more neighbor nodes and keys to communicate with them), and the number of nodes in the graph by n. The number of clauses in the corresponding MAX-SAT is $n[2^q - \binom{q}{2}]$. Our MAX-SFNK algorithm, presented in Algorithm 1, has the time complexity as MAX-SAT. The number of clauses in MAX-SFNK, however, increases with the number of keys of a node. This means that MAX-SFNK deals with more clauses than the regular q-MAX SAT problem, where q is the maximum number of keys assigned to a node (note that S-WGPS aims to keep q low).

Johnson's algorithm constructs a truth assignment that satisfies at least $p(1-1/2^q)$ clauses, where p is number of clauses equal to $n[2^q - \binom{q}{2}]$. Consequently, the minimum number of covered nodes is $n - n/2^q$. Since a node with an unsatisfied clause is not necessarily uncovered, we can say that at most $n/2^q$ nodes may be left uncovered and additional rebroadcasts are needed for them to be covered. Thus, the upper bound for the number of additional rebroadcasts is $OPT+(n/2^q)$, where OPT is the optimal value. This implies that the success rate of covering the entire network, such that each node uses exactly two keys, increases when the network density increases.

It is interesting to identify the worst case scenario for our algorithm, which is a star graph. In a star topology, the central node cannot cover all its neighbors with a single rebroadcast. Our MAX-SFNK allows the central node to perform multiple rebroadcasts (Lines 8 - 11 in Algorithm 1 continue execution as long as there exists an unassigned boolean variable (key) corresponding to the covered nodes.)

VI. Performance Evaluation

We implemented our MAX-SFNK algorithm in Matlab and performed simulations on a desktop PC with a 3GHz Intel Core 2 Duo CPU, 4GB RAM. We evaluated the performance of MAX-SFNK by measuring the number of rebroadcasts needed (Rebroadcast Ratio) for 100% network coverage of the flooding message. Additionally, we measured the execution time of the algorithm for different network sizes and different node densities (i.e., 7, 12, 19, and 28 nodes per radio range for all of the evaluations). Since the performance of S-WGPS is affected by localization success rate and GPS error, we also evaluated network coverage and execution time of the algorithm for different localization success rates and GPS errors. Localization success rate refers to the percentage of nodes that are localized by GPS (the remaining nodes are less precisely localized by the IMU).

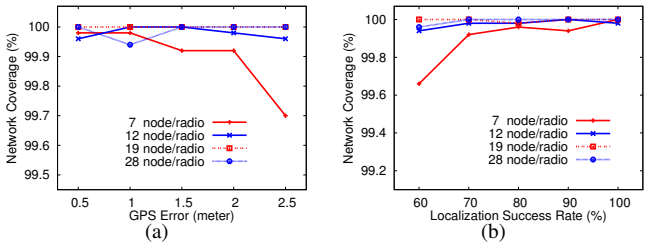

Fig. 5. The impact of GPS error and localization success rate on network coverage.

A. Network Coverage

First, we evaluated the impact of two parameters, localization success rate and GPS error, on network coverage. These two parameters affect the accuracy of localization and consequently the key assignment in S-WGPS. Since only two physical neighbors may share a key, a key shared by two far apart nodes is set to FALSE in MAX-SFNK. Moreover, if there is a key assigned to only one node, MAX-SFNK ignores it. By avoiding such invalid keys a node may become disconnected from the entire network, affecting the network coverage. It is important to note that coverage less than 100% is caused by S-WGPS (its key distribution scheme). Figures 5(a) and 5(b) show the impact of GPS error and localization success rate on network coverage, respectively, for a 500-node network of different densities. As depicted, a higher GPS error causes a lower network coverage; however, it is very close to 100%. Also, the higher success rate in localization means less keys are invalid and the network coverage is higher.

B. Rebroadcast Ratio

Next, we ran simulations for different network sizes (e.g., 200, 400, 600, and 800 nodes), and different network densities, to measure the rebroadcast ratio of our algorithm. The average value for rebroadcast ratio is for all network configurations (i.e., GPS error and localization success rate) for which S-WGPS ensured 100% coverage. As shown in Figure 6(a), networks with higher density have lower rebroadcast ratio for covering the entire network because the higher node density results in a larger number of triangles in the network. As discussed before, triangles in the network help nodes to have MLTCST-type transmissions, which require only one key. Thus, more neighbor nodes are covered, while using less rebroadcasts. Figure 6(a) also depicts a small improvement in the number of required broadcasts when the network size increases, while keeping a constant node density. This can be explained by the fact that larger networks have more possible paths to cover nodes. It is very interesting to remark that the average rebroadcast ratio of our MAX-SNFK algorithm can be below 1 (more precisely 0.75, as shown in Figure 6(a)),

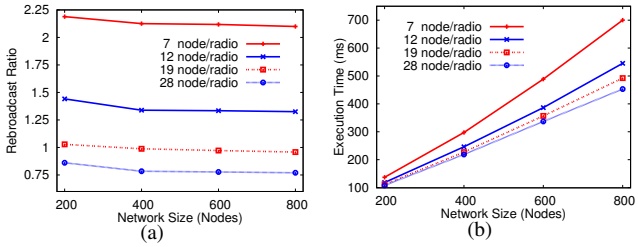

Fig. 6. a) The rebroadcast ratio for different network sizes and node densities. b) The execution time for different network sizes and densities.

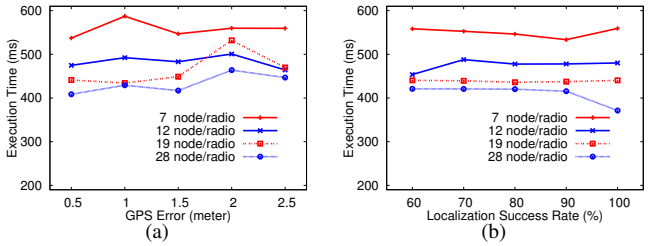

Fig. 7. The impact of GPS error and localization success rate on execution time.

while a typical flooding mechanism has a rebroadcast ratio of 1, and, worse, S-WGPS has an average rebroadcast ratio equal to the average number of keys a node has.

C. Execution Time

Finally, we investigated the effects of network size, network density, GPS error, and localization success rate on execution time. The execution time of MAX-SFNK, for different network sizes and densities, is depicted in Figure 6(b). As shown, it takes less than 1 second to select the set of keys for optimal flooding, even for the largest network size. The results also show that in networks with higher node densities the execution time of MAX-SFNK decreases. This is because each key is shared by more nodes in a network with high node density, thus more clauses are satisfied by a single boolean variable set to TRUE by our algorithm. Figures 7(a) and 7(b) also show the impact of GPS error and localization success rate on execution time for 500-node network. As depicted, these two parameters, have a small impact on execution time. This can be explained as follows: when some of the nodes become disconnected from the network, possibly because removing invalid keys, there will still be some unassigned keys belonging to uncovered nodes. This results in MAX-SFNK algorithm to perform additional iterations in its main loop. We note here that our MAX-SFNK algorithm executes until all nodes are covered or their corresponding keys are assigned a FALSE value (Lines 8 and 9). In this scenario, we will not be able to cover all nodes, hence the condition for terminating the algorithm is FALSE, which will require additional iterations.

VII. Conclusions

In this paper we investigate efficient flooding in WSN secured with neighborhood keys. When neighborhood keys are used, broadcasting packets incurs high costs, since multiple unicast packets need to be sent. For flooding code updates (a common operation in WSN, since they are physically inaccessible) the naive support of broadcasting through multiple unicast transmission can be very costly. We formulate the problem of deciding if it is possible to achieve 100% network coverage by a flooding packet, when each node cleverly chooses one of its keys to unicast the broadcast message. We show that the problem in NP-hard and propose an optimization version of it, and an approximation algorithm. Through simulations, we demonstrate the 100% network coverage by the flooding packets can be achieved, at a cost, surprisingly, that can be less than that of traditional flooding (where each node broadcasts each message exactly once).

References

[1] Q. Mi, J. A. Stankovic, and R. Stoleru, "Secure walking GPS: a secure localization and key distribution scheme for wireless sensor networks," in *WiSec*, 2010, pp. 163–168.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comp. Netw.*, pp. 393–422, 2002.

[3] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: An integrated sensor network system for energy-efficient surveillance," *ACM Trans. on Sen. Netw.*, pp. 1–38, 2006.

[4] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, pp. 18–25, 2006.

[5] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," RPI Computer Science Department, Technical Report 04-10, Tech. Rep., 2004.

[6] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Commun. ACM*, vol. 47, pp. 53–57, June 2004.

[7] L. Lazos and R. Poovendran, "SeRLoc: secure range-independent localization for wireless sensor networks," in *WiSe*, 2004, pp. 21–30.

[8] R. Stoleru, T. He, and J. Stankovic, "Walking GPS: a practical solution for localization in manually deployed wireless sensor networks," in *the 29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 480 – 489.

[9] D. Malan, M. Welsh, and M. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *SECON*, 2004.

[10] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ser. SASN '04, 2004.

[11] P. Ning, A. Liu, and W. Du, "Mitigating DoS attacks against broadcast authentication in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, pp. 1:1–1:35, February 2008.

[12] D. Song, D. Zuckerman, and J. D. Tygar, "Expander graphs for digital stream authentication and robust overlay networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[13] C. A. Gunter, S. Khanna, K. Tan, and S. Venkatesh, "DoS protection for reliably authenticated broadcast," in *NDSS*, 2004.

[14] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," in *IEEE Symp. on Security and Privacy*, 2002.

[15] D. Liu and P. Ning, "Multilevel uTESLA: Broadcast authentication for distributed sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, pp. 800–836, November 2004.

[16] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, pp. 521–534, September 2002.

[17] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," in *NDSS*, 2003.

[18] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[19] J. Chen, D. K. Friesen, and H. Zheng, "Tight bound on Johnson's algorithm for maximum satisfiability," *Journal of Computer and System Sciences*, vol. 58, pp. 622–640, June 1999.