Computers **2015**, *4*, 215-233; doi:10.3390/computers4030215

OPEN ACCESS

computers

ISSN 2073-431X

www.mdpi.com/journal/computers

Article

Security Property Validation of the Sensor Network Encryption Protocol (SNEP)

Salekul Islam

Department of Computer Science and Engineering, United International University, Dhaka 1209, Bangladesh; E-Mail: salekul@cse.uiu.ac.bd; Tel.: +88-2-9125912; Fax: +88-2-9118170 .

Academic Editor: Pedro Alonso Jordá

Received: 30 May 2015 / Accepted: 17 July 2015 / Published: 24 July 2015

Abstract: Since wireless sensor networks (WSNs) have been designed to be deployed in an unsecured, public environment, secured communication is really vital for their wide-spread use. Among all of the communication protocols developed for WSN, the Security Protocols for Sensor Networks (SPINS) is exceptional, as it has been designed with security as a goal. SPINS is composed of two building blocks: Secure Network Encryption Protocol (SNEP) and the "micro" version of the Timed Efficient Streaming Loss-tolerant Authentication (TESLA), named μTESLA. From the inception of SPINS, a number of efforts have been made to validate its security properties. In this paper, we have validated the security properties of SNEP by using an automated security protocol validation tool, named AVISPA. Using the protocol specification language, `HLPSL`, we model two combined scenarios—node to node key agreement and counter exchange protocols—followed by data transmission. Next, we validate the security properties of these combined protocols, using different AVISPA back-ends. AVISPA reports the models we have developed free from attacks. However, by analyzing the key distribution sub-protocol, we find one threat of a potential DoS attack that we have demonstrated by modeling in AVISPA. Finally, we propose a modification, and AVISPA reports this modified version free from the potential DoS attack.

Keywords: wireless sensor networks (WSN); Secure Network Encryption Protocol (SNEP); key distribution, strong freshness, validating security protocols; AVISPA; DoS attack

---

1. Introduction

Wireless sensor networks (WSNs) with proliferation in micro-electromechanical systems (MEMS) technology and low-cost smart sensors have emerged as a widely-deployed technology for gathering data from the environment. In recent years, many limitations of WSNs, including cost, limited processing and computing resources, and shorter battery life have been improved significantly [1]. As a result, a wide range of tiny sensors—mechanical, thermal, biological, chemical, optical and magnetic sensors—is being designed to perform various tasks [2]. Due to the versatile uses of sensor nodes, the applications of WSNs are found in military surveillance, home medical care and environmental science [3].

Typically, a node is comprised of one or more sensors that can read data from its surroundings. The read data are then transmitted to a base station, which has more computing and storage resources than a node. In general, node to base station and node to node connections are wireless connections and, thus, are publicly accessible. Since any wireless link is susceptible to attack, a WSN is inherently vulnerable to attacks from outsiders. Sensor networks face a number of security challenges, including secure routing, key establishment, secrecy, authentication, privacy, robustness and denial-of-service (DoS) attacks [4,5]. Conventional end to end security mechanisms, such as SSH or SSL, are not possible to deploy in WSNs, due to in-network aggregation of data that requires intermediate routers to have access to the message content [6]. Moreover, sensor nodes with their low computing and storage resources are not capable of processing asymmetric cryptographic algorithms (e.g., typical RSA keys are 1024 to 2048 bits long [7]).

Most of the sensor networks' routing protocols are vulnerable to some kind of attack, as these protocols have not been deigned with security as a goal [6]. One exception is Security Protocols for Sensor Networks (SPINS) [8], which has been designed to meet a set of security requirements. SPINS has been designed to provide secured communication in WSNs, which is feasible for limited resources, as well. SPINS, a set of security protocols, has two building blocks: the Secure Network Encryption Protocol (SNEP) and a "micro" version of the Timed Efficient Streaming Loss-tolerant Authentication (TESLA) [9], named μTESLA. The security requirements of SPINS protocols are summarized as follows:

- Data confidentiality confirms that a sensor network's information, especially the readings of the sensors, is kept secret from other networks. Encryption is the most commonly-used technique for providing data confidentiality.
- Data authentication confirms that data were really sent by the claimed sender. A symmetric key (shared by the sender and the receiver)-based message authentication code (MAC) ensures data authenticity.
- Data integrity verifies that data have not been altered *en route* by any adversary. MAC-based data authentication includes data integrity.
- Data freshness ensures that data are recent and not replayed by an adversary. Nonces and counters are two ways to provide freshness.
- Replay protection prevents an adversary from reusing an old message.
- Low communication overhead is a basic security property that should be ensured by any WSN security protocols.
- Semantic security confirms that no two messages are identical. This could be confirmed by sending a counter or timestamp that is modified in every message.

In this paper, we have validated the security properties of SNEP by modeling the protocol using Automated Validation of Internet Security Protocols and Applications (AVISPA) [10]. AVISPA is a push-button tool with industrial-strength technology for the analysis of different Internet security protocols and applications. It is being used by the developers of different security protocols and by academic researchers, as well [11]. Previously, we used AVISPA to validate the security properties of the Lightweight Mutual Authentication Protocol (LMAP), an RFID tag authentication protocol [12]. In our validation, although AVISPA reports the SNEP protocol free from attacks, we finds the threat of a DoS attack that we have demonstrated by modeling in AVISPA.

The rest of the paper is organized as follows: Section 2 presents different SNEP sub-protocols. Section 3 briefly explains the tool, AVISPA: how to develop models of security protocols and how to validate security goals. Section 4 illustrates how we develop the AVISPA model of SNEP: the assumptions and limitations of the model, the security goals and how we model those goals and the validation results. Section 5 explains the potential DoS attack: simulation of the attack and how to prevent this attack. The next section summarizes the related work, and finally, Section 7 concludes the paper, explaining the future work that could be accomplished based on the contributions of the paper.

2. Secure Network Encryption Protocol

SPINS has been designed to provide security for two types of communications among the base station and the nodes:

- Unicast communication: Node to base station (e.g., sensor readings) and base station to node (e.g., specific requests) are kinds of unicast communication between the base station and a specific node. SNEP has been designed to secure all of these unicast communications.
- Multicast communication: Base station to all nodes (e.g., routing beacons, queries or reprogramming of the entire network) needs one to many multicast communication. Note that the TESLA protocol [9] was originally developed to secure multicast data transmission. However, TESLA is not designed for the limited computing environments commonly found in WSNs. To secure multicast communications in WSNs, a tiny and stripped version of the TESLA protocol has been proposed, which is referred to as µTESLA.

Note that in this paper, we model and validate the security properties of SNEP only. Table 1 presents the notations that are being used in SNEP specification. To secure the communication from the direction A to B, two different keys are used for encryption (*i.e.*, symmetric key K_{AB}) and message authentication code (MAC) generation (*i.e.*, shared key K'_{AB}). Similarly, two different keys are used to secure the communication from the direction B to A. Hence, A and B need two pairs of keys (four keys in total) to secure bidirectional communication between them. These keys are either derived from the master secret, χ_{AB}, shared between A and B, or distributed by a trusted server. SNEP is composed of a number of sub-protocols, which are explained in the following.

Table 1. Notations being used in SNEP.

Notation	Meaning
A, B	Principals, such as communicating nodes.
N_A	Nonce generated by A.
C_A	Counter generated and updated by A.
χ_{AB}	Master secret key shared between A and B.
K_{AB}	Symmetric key between A and B used for encryption of messages. This is derived from χ_{AB}.
K'_{AB}	MAC key between A and B used for MAC calculation. This is derived from χ_{AB}.
$\{M\}_{K_{AB}}$	Encryption of message M with the encryption key K_{AB}.
$\{M\}_{\langle K_{AB},IV\rangle}$	Encryption of message M with the encryption key K_{AB} and the initialization vector IV.
$MAC(K'_{AB}, M)$	Computation of the message authentication code (MAC) of message M, with the key K'_{AB}.

2.1. Data Transmission with Weak Freshness

Data transmission protocol with weak freshness has been presented in Figure 1. Using the only message of this protocol, any node A can send data D to another node B, where D is encrypted using the shared secret K_{AB} and the counter C_A. Moreover, the MAC of the encrypted data and the counter is calculated using the MAC key K'_{AB}. This protocol provides all of the security properties mentioned in the earlier section, except it provides weak freshness. By checking the increased counter value, the receiver can verify that the message has been sent after the previous one.

$$A \rightarrow B : \{D\}_{\langle K_{AB},C_A\rangle}, MAC(K'_{AB}, C_A||\{D\}_{\langle K_{AB},C_A\rangle})$$

Figure 1. Data transmission protocol with weak freshness.

2.2. Data Transmission with Strong Freshness

For strong freshness, two messages are being exchanged, which are shown in Figure 2. In this protocol, A first sends a fresh nonce N_A to B with a request message R_A. When B returns its response (*i.e.*, the actual data are being transmitted here), it includes N_A while calculating the MAC. The inclusion of the nonce ensures that B has sent this message in response to A's request. This ensures a total order through the request to response pair and is referred to as strong freshness in SNEP. Moreover, this completes the challenge to response exchanges and authenticates the identity of B to A.

$$A \rightarrow B : N_A, R_A$$
$$B \rightarrow A : \{R_B\}_{\langle K_{BA}, C_B \rangle}, MAC(K'_{BA}, N_A||C_B||\{R_B\}_{\langle K_{BA}, C_B \rangle})$$

Figure 2. Data transmission protocol with strong freshness.

2.3. Counter Exchange Sub-Protocol

The counter exchange sub-protocol shown in Figure 3 bootstraps the initial counter values C_A and C_B. In the first two messages, A sends its counter C_A and B sends its counter C_B to each other. Note that counters are sent without encryption, since they are not secret values. In these messages, counters are used as nonces to ensure strong freshness. In the second and third messages, B and A are being authenticated by A and B, respectively.

$$A \rightarrow B : C_A$$
$$B \rightarrow A : C_B, MAC(K'_{BA}, C_A||C_B)$$
$$A \rightarrow B : MAC(K'_{AB}, C_A||C_B)$$

Figure 3. Counter exchange sub-protocol.

2.4. Node to Node Key Agreement Sub-Protocol

The node to node key agreement sub-protocol, shown in Figure 4, is being used when a node wants to communicate with another node and they do not have any shared secret key. With the help of the base station (here, the trusted third party S), two nodes can establish a shard secret key, SK_{AB}. Note that with S, A and B share secret χ_{AS} and χ_{BS}, respectively.

$$A \rightarrow B : N_A, A$$
$$B \rightarrow S : N_A, N_B, A, B, MAC(K'_{BS}, N_A||N_B||A||B)$$
$$S \rightarrow A : \{SK_{AB}\}_{K_{SA}}, MAC(K'_{SA}, N_A||B||\{SK_{AB}\}_{K_{SA}})$$
$$S \rightarrow B : \{SK_{AB}\}_{K_{SB}}, MAC(K'_{SB}, N_B||A||\{SK_{AB}\}_{K_{SB}})$$

Figure 4. Node to node key agreement sub-protocol.

The following security properties are expected to be fulfilled by this protocol:

- Due to the use of N_A and N_B, two properties are guaranteed: (i) strong key freshness to both A and B; and (ii) authentication of S to A (at the end of the third message) and S to B (at the end of the fourth message).
- The confidentiality of SK_{AB} through encryption.
- The data authenticity and integrity of SK_{AB} through keyed MAC, which confirms that it was generated by the base station.
- Defending the base station against the denial-of-service attack through the MAC in the second message; thus, the base station confirmed that the request has come from a legitimate node.

- Saving energy of the nodes by engaging base station to perform the major tasks, e.g., key generation and distribution.

3. AVISPA

AVISPA is being used by the developers of different security protocols and by academic researchers, as well [13]. The AVISPA community has modeled a number of Internet Engineering Task Force (IETF) security protocols, and several protocols have been found flawed. A modular and expressive formal language is being used to specify a protocol and its security properties. AVISPA depends on its back-ends, which use an automatic analytic technique to detect any flaws if they exist.

3.1. Architecture

The architecture of AVISPA is shown in Figure 5 [10]. In AVISPA, protocol roles are modeled in the High-Level Protocol Specification Language (`HLPSL`) as state transition systems. The `HLPSL2IF` tool translates a `HLPSL` specification into an Intermediate Format (`IF`), a transition system of infinite state. This `IF` specification is being analyzed by any of the four back-end tools: On-the-Fly Model-Checker (`OFMC`), CL-based Attack Searcher (`CL-AtSe`), SAT-based Model Checker (`SATMC`) and Tree Automata-based Protocol Analyzer (`TA4SP`).

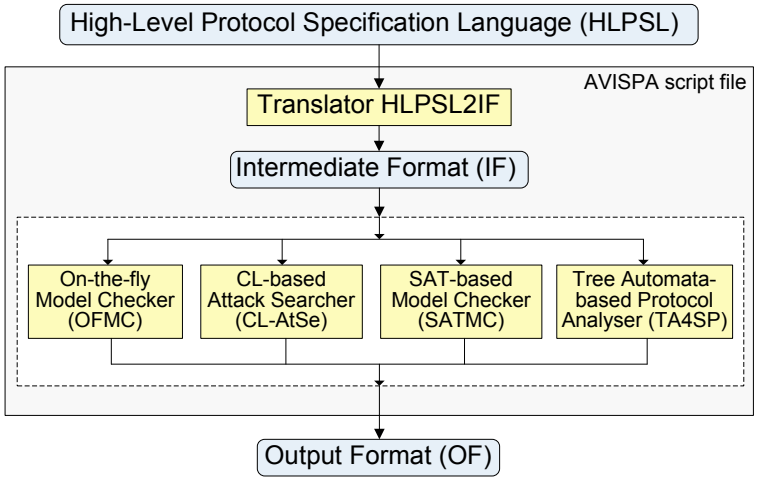

Figure 5. Architecture of the AVISPA tool.

3.2. Security Goal Specification

Using the back-end tools of AVISPA, secrecy and different forms of authentication goals could be validated. AVISPA supports four types of goal predicates: `witness`, `wrequest` (for weak authentication), `request` (for strong authentication) and `secret`. In the following, we explain how the predicates are used to specify security goals: secrecy and authentication.

Secrecy is verified with the help of goal predicate `secret(E, id, S)`, which confirms that the secret information `E` should be known only by the agents of set `S`. The label `id` (of type `protocol_id`) is used to identify the goal. In the `HLPSL` goal section, the statement `secrecy_of id` should be given to refer to it. An intruder attempting to break a secret mounts different attacks. On a successful attack,

the intruder learns a value that is considered as secret and that he is not allowed to know, and hence, the secrecy property is violated.

Authentication is verified with the help of different goal predicates: `witness(A, B, id, E)`, `request(B, A, id, E)` (for strong authentication) and `wrequest(B, A, id, E)` (for weak authentication). The `witness` predicate is used for (weak) authentication of `A` by `B` on `E`. This is a confirmation that `A` is a witness for the information `E`. The `request` predicate is used for the strong authentication property (similarly, `wrequest` is for weak authentication) of `A` by `B` on `E`, which declares that `B` requests a check of the value `E`. The authentication property is expressed in the `HLPSL` goal section, which is written as `authentication_on id` (similarly, `weak_authentication_on id` for weak authentication). In this expression, `id` is a label (of type protocol_id) to tag the goal. If any of the back-end tools finds a trace in which the `request` event is preceded by a `witness` event originated by an agent other than `A`, an attack will be reported. Moreover, an attack trace will be reported if no valid `witness` is found for a `request`.

4. Developing SNEP Model in AVISPA

Before explaining how we develop the models, we would like to explain the simplifications we have introduced to the SNEP protocols.

4.1. Simplification of SNEP Protocols

4.1.1. Number of Keys

In original SNEP protocols, when A and B communicate, they use separate keys for encryption and MAC calculation and also in opposite directions. Thus, a set of four keys (*i.e.*, K_{AB}, K_{BA}, K'_{AB} and K'_{BA}) is derived from the master secret, χ_{AB}. This provides the separation of cryptographic primitives. However, AVISPA does not differentiate among symmetric keys, and thus, to keep our model simple, yet not deviating from the original specification, we reduce the number of keys between A and B to one. Since, AVISPA does not allow key derivation, we simply pass the same symmetric keys to A and B.

4.1.2. Replacing MAC with a Hash Function

In SNEP, the keyed MAC function is used for message authentication. AVISPA provides the hash function, which might be also used as the keyed hash function by adding the symmetric key in the hash calculation. Thus, instead of using $MAC(K'_{AB}, M)$, we use `h(K'ab.M)` in AVISPA models where `h()` is a hash function and `K'ab.M` means the concatenation of `K'ab` and `M`.

4.1.3. Modeling $\{M\}_{\langle K_{AB}, IV\rangle}$ with Regular Encryption

AVISPA does not support encryption with an initialization vector. Therefore, we use `{M}_(Kab.IV)`, where the initial vector (`IV`) is concatenated with the key `Kab`, and this concatenated message is used to encrypt the message `M`. The reason for introducing `IV` with the symmetric key in encryption was to ensure that even if the same `M` is being sent, the encrypted message will be changed

due to the change in IV, and thus, semantic security is provided. Hence, this modification ensures that the security property remains the same as the original SNEP protocol.

4.2. Node to Node Key Agreement with Strong Freshness

While developing the AVISPA model, we develop a combined model of two sub-protocols: node to node key agreement and data transmission with strong freshness. The protocol messages we have modeled are shown in Figure 6.

```
1: A -> B : Na.A.Ra
2: B -> S : Na.Nb.A.B.h(Kbs.Na.Nb.A.B)
3: S -> A : {SKab}_Kas.h(Kas.Na.B.{SKab}_Kas)
4: S -> B : {SKab}_Kbs.h(Kbs.Nb.A.{SKab}_Kbs)
5: B -> A : {Rb}_SKab.h(SKab.Na.{Rb}_SKab)
```

Figure 6. Messages for node to node key agreement with strong freshness.

In the following, we explain each message briefly:

- We assume the scenario where node A sends a request Ra with a nonce Na to node B, but they do not have any shared key. However, A and B share master secrets χ_{AS} and χ_{BS} with the trusted third party S, from which symmetric keys Kas and Kbs are derived.
- B requests S to establish a symmetric key between A and B. While sending this request, B sends a fresh nonce, Nb with Na. Note that the keyed hash of the first part, *i.e.*, h(Kbs.Na.Nb.A.B), is included for message authentication.
- Next, S generates a new symmetric key, SKab, and sends this to A and B through two separate messages. While sending A (B), this key is encrypted with Kas (Kbs). Moreover, the keyed hash is calculated using Kas (Kbs), which also includes the nonce Na (Nb).
- Now, B sends Rb encrypted by SKab to A in response to the request it received in the first message. The keyed hash sent with this message is calculated using SKab, which includes Na.

In the AVISPA model we develop, there are three agent roles: server for the trusted third party (S), alice for node A and bob for the other node (B). Figure 7 shows the partial HLPSL specification that we develop to model these three roles. The security goals of the node to node key agreement protocol and the data transmission protocol with strong freshness were mentioned earlier. However, AVISPA is not able to validate all of those properties. The following security properties can be validated using AVISPA:

(1) Secrecy of the symmetric key SKab, which is validated using the HLPSL goal secrecy_of and the corresponding secret fact secret(SKab, skab, {A, S, B}). Here, the label skab identifies the goal, and {A, S, B} is the set of agents that are allowed to learn the value SKab.
(2) Upon receiving the third message, A authenticates S through the nonce Na. We use the label alice_server_na to identify the goal. To validate this security goal, we add witness(S, A, alice_server_na, Na') and request(A, S, alice_server_na, Na) statements inside the roles of S and A, respectively.

(3) Upon receiving the fourth message, `B` authenticates `S` through the nonce `Nb`. We use the label `bob_server_nb` to identify the goal. To validate this security goal, we add `witness(S, B, bob_server_nb, Nb')` and `request(B, S, bob_server_nb, Nb)` statements inside the roles of `S` and `B`, respectively.

(4) Upon receiving the fifth message, `A` authenticates `B` through the nonce `Na`. We use the label `alice_bob_na` to identify the goal. To validate this security goal, we add `witness(B, A, alice_bob_na, Na)` and `request(A, B, alice_bob_na, Na)` statements inside the roles of `B` and `A`, respectively.

(5) The secrecy of the response of `B` to `A` (*i.e.*, `Rb`) is validated using the fact `secret(Rb, rb, {A, B, S})`. Here, the label `rb` identifies the goal, and `{A, B, S}` is the set of agents that are allowed to learn the value `Rb`.

```
role alice (A,S,B:agent, Kas:symmetric_key, Hash:hash_func,
     SND_BA,RCV_AB,RBA,RCV_SA   :channel(dy)) played_by A def=

local State:nat, Na,Ra,Rb:text, SKab:symmetric_key

init State :=0

transition
0. State  =  0 /\ RCV_SA(start) =|>
   State' :=  2  /\  Na' := new() /\ SND_BA (Na'.A.Ra)
2. State   = 2 /\ RCV_SA ({SKab'}_Kas.Hash(Kas.Na.B.{SKab'}_Kas)) =|>
   State' := 4 /\ request (A,S,alice_server_na,Na)
4. State = 4 /\ RCV_AB({Rb}_SKab.Hash(SKab.Na.{Rb}_SKab)) =|>
   State' :=6 /\ request(A,B,alice_bob_na,Na) /\ secret(SKab,skab,{A,S,B})
end role

role server (A,S,B:agent, Kas,Kbs:symmetric_key, Hash:hash_func,
     SND_AS,SND_BS, RCV_BS:channel (dy)) played_by S def=

local State:nat Na,Nb:text, SKab:symmetric_key

init State :=1

transition
1. State =1  /\  RCV_BS(Na'.Nb'.A.B.Hash(Kbs.Na'.Nb'.A.B)) =|>
   State' :=3 /\ SKab' := new() /\ SND_AS ({SKab'}_Kas.Hash(Kas.Na'.B.{SKab'}_Kas))
           /\ SND_BS({SKab'}_Kbs.Hash(Kbs.Nb'.A.{SKab'}_Kbs))
           /\ witness(S,A,alice_server_na,Na') /\ witness(S,B,bob_server_nb,Nb')
end role

role bob(A,S,B:agent, Kbs:symmetric_key, Hash:hash_func,
     SND_SB,SND_AB,RCV_AB, RCV_SB:channel(dy)) played_by B def=

local State:nat, Na,Nb,Ra,Rb:text, SKab:symmetric_key

init State :=6

transition
6. State = 6 /\ RCV_AB(Na'.A.Ra) =|>
   State' := 8 /\ Nb' :=new() /\ SND_SB(Na'.Nb'.A.B.Hash(Kbs.Na'.Nb'.A.B))
8. State  =8 /\  RCV_SB ({SKab'}_Kbs.Hash(Kbs.Nb.A.{SKab'}_Kbs))=|>
   State' :=10/\ SND_AB({Rb}_SKab'.Hash(SKab'.Na.{Rb}_SKab'))
   /\ request (B,S,bob_server_nb,Nb)/\ witness(B,A,alice_bob_na,Na)
   /\ secret(SKab,skab,{A,S,B})
end role
```

Figure 7. Agent roles modeled in the node to node key agreement with strong freshness.

4.3. Counter Exchange Protocol with Strong Freshness

In case A and B do not have the updated counter values of each other, the counter exchange protocol is executed between A and B. This is a bootstrap protocol. While we model in AVISPA, we develop a counter exchange protocol followed by data transmission that uses the updated counter value. Figure 8 shows the messages that we have modeled in AVISPA. As we have mentioned before, instead of using keyed MAC, we use the keyed hash function. Similarly, we use {D}_(Kab.Ca) to replace encryption with the initialization vector (*i.e.*, $\{D\}_{\langle K_{AB}, C_A \rangle}$), where Ca is the initialization vector. Note that we use two different keys, Kab and Kba in two directions, *i.e.*, from A to B and from B to A. Here, the counters Ca and Cb are being used as nonces, which could be considered as challenges for authentication.

```
1: A -> B : Ca
2: B -> A : Cb.h(Kba.Ca.Cb)
3: A -> B : h(Kab.Ca.Cb)
4: A -> B : {D}_(Kab.Ca).h(Kab.Ca.{D}_(Kab.Ca))
```

Figure 8. Messages for the counter exchange protocol with data transmission.

In the AVISPA model we develop, there are two agent roles: alice for node A and bob for the other node, B. Figure 9 shows the partial HLPSL specification that we develop to model these two roles. The following security properties can be validated using AVISPA.

(1) Upon receiving the second message, A authenticates B through the counter (nonce) Ca. We use the label alice_bob_ca to identify the goal. To validate this security goal, we add witness(B, A, alice_bob_ca, CA') and request(A, B, alice_bob_ca, CA) statements inside the roles of B and A, respectively.

(2) Upon receiving the third message, B authenticates A through the counter (nonce) Cb. We use the label bob_alice_cb to identify the goal. To validate this security goal, we add witness(A, B, bob_alice_cb, CB') and request(B, A, bob_alice_cb, CB) statements inside the roles of A and B, respectively.

(3) The secrecy of data, D sent to B from A. This goal is validated using the fact secret(D', d, {A, B})). Here, the label d identifies the goal, and {A, B} is the set of agents that are allowed to learn the value D.

```
role alice (A,B:agent, Kab,Kba:symmetric_key,
     Hash:hash_func, SND,RCV:channel(dy)) played_by A def=

local
  State : nat,
  CA,CB : text,
  D     : message

init
  State :=0

transition
0. State = 0 /\ RCV(start) =|>
   State' := 2 /\  CA' :=new() /\ SND(CA')
2. State = 2  /\ RCV(CB'.Hash(Kba.CA.CB')) =|>
   State' := 4 /\ SND(Hash(Kab.CA.CB')) /\ D' := new()
          /\ SND({D'}_(Kab.CA).Hash(Kab.CA.{D'}_(Kab.CA)))
          /\witness(A,B,bob_alice_cb,CB')
          /\ request (A,B,alice_bob_ca,CA)
          /\ CA' := xor (CA,1)
end role

role bob (A,B:agent, Kab,Kba:symmetric_key,
     Hash:hash_func, SND,RCV:channel(dy)) played_by B def=

local
  State : nat,
  CA,CB : text,
  D     : message

init
 State :=1

transition
1. State  =1 /\ RCV(CA') =|>
   State' := 3 /\ CB' := new() /\SND(CB.Hash(Kba.CA'.CB'))
               /\witness(B,A,alice_bob_ca,CA')
3. State  =3 /\ RCV(Hash(Kab.CA.CB)) =|> State' := 5
             /\ request (B,A, bob_alice_cb,CB)
5. State =5/\RCV({D'}_(Kab.CA).Hash(Kab.CA.{D'}_(Kab.CA)))
                =|> State' :=7 /\ secret(D',d,{A,B})
                               /\ CA' := xor (CA,1)
end role
```

Figure 9. Agent roles modeled in the counter exchange protocol with strong freshness.

4.4. Freshness, Replay and MitM Attacks

AVISPA supports a `new()` function that generates a fresh value at runtime. For ensuring the freshness of the nonces (denoted by `Na` and `Nb` in Figure 7) and the counters (denoted by `Ca` and `Cb` in Figure 9), we use the `new()` function inside the roles that generate the nonce/counter.

Figure 10 shows the HLPSL definition of the roles `session()` and `environment()` that we develop in the node to node key agreement with strong freshness. Similar roles are also being developed for the counter exchange model. In these models, to detect the existence of any replay attack, we invoke two identical `session()`'s of the model in parallel to the `environment()` role.

```
role session(A,S,B:a ent, Kas,Kbs:symmetric_key, Hash:hash_func) def
local SBA,SBS, SAS,SSB, SAB,RBA, RSA, RSB, RAB, RBS : channel(dy)

composition
     alice(A,S,B,Kas,Hash,SBA,RAB,RBA,RSA)   invokes a ent A
     server(A,S,B,Kas,Kbs,Hash,SAS,SBS,RBS)  invokes a ent S
     bob(A,S,B,Kbs,Hash,SSB,SAB,RAB,RSB)     invokes a ent B
end role


role environment() def
const
  a,s,b,i : a ent,
  kas,kbs,kis : symmetric_key,
  alice_server_na, bob_server_nb, alice_bob_na,skab : protocol_id,
  h : hash_func

intruder_knowled e {a,s,b,kis,h,i}

composition
     session(a,s,b,kas,kbs,h)
     session(a,s,b,kas,kbs,h)
     session(a,s,i,kas,kis,h)  intruder playin  role of B
     session(i,s,b,kis,kbs,h)  intruder playin  role of A
end role
```

Figure 10. `session` and `environment` roles modeled in the node to node key agreement protocol with strong freshness.

Moreover, an active intruder, who can play simultaneously the role of any communicating node, has been introduced. Since the server has been considered as a trusted entity, the intruder is not allowed to play the role of the server. This will also detect Man-in-the-Middle (MitM) or connection hijack attacks if any such attack exists. Figure 10 shows that in the `environment()` role, we introduce an active intruder playing the roles of the agents `A` and `B`.

4.5. Model Validation and Analysis

Upon completion of the AVISPA model development using `HLPSL`, we use SPAN (Security Protocol Animator) [14], which symbolically executes an `HLPSL` protocol specification. SPAN is very useful while writing `HLPSL` specifications, since it gives better understanding and confirms if the specification is executable. We use SPAN's protocol simulation that builds message sequence charts (MSC) corresponding to our `HLPSL` specifications. During protocol simulation, no intruder's role has been introduced. Figures 11 and 12 show the MSC of the AVISPA models that we have built to validate node to node key agreement and counter exchange protocols with strong freshness. The figures confirm that the number of messages exchanged while the models are executed are equal to the messages we have modeled.

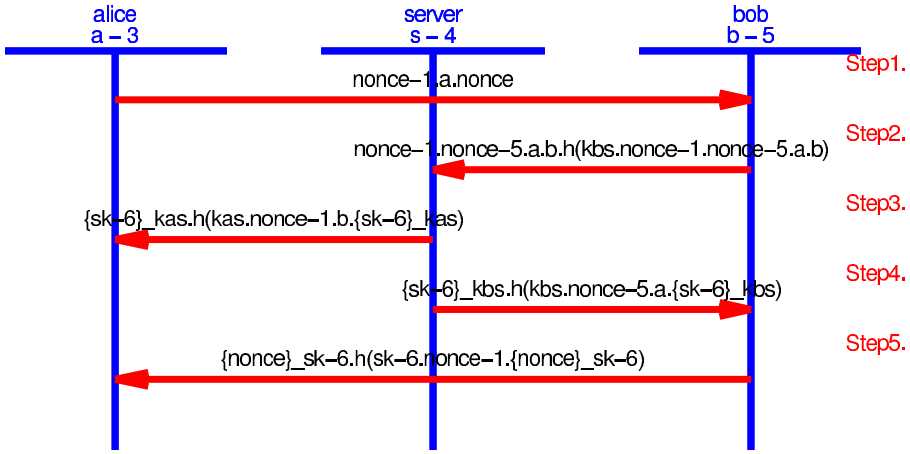

Figure 11. Protocol simulation for node to node key agreement with strong freshness.

Figure 12. Protocol simulation for the counter exchange protocol with strong freshness.

AVISPA supports Dolev–Yao channels [15] (denoted by `channel (dy)` in the roles) for message transfer. In this model, the adversary or intruder has full control over message transfer. She can overhear, intercept and synthesize any message, and her actions are only limited by the constraints of the cryptographic methods used. Next, we introduce an active intruder and verify the model using all four back-ends of AVISPA to find an attack if any exists. AVISPA cannot simultaneously detect multiple attacks during a single run. Therefore, we verify the security goals one-by-one. The first two back-ends, On-the-Fly Model-Checker `OFMC` and Constraint Logic-based Attack Searcher `CL-AtSe` have reported `SAFE` for `BOUNDED_NUMBER_OF_SESSIONS`. The other two, SAT-based Model-Checker `SATMC` and Tree Automata based Automatic Approximations for the Analysis of Security Protocols `TA4SP` , have reported `NOT_SUPPORTED` and produced `INCONCLUSIVE` results. Figure 13 shows the validation output of `OFMC` for node to node key agreement with strong freshness. Due to the complexity of the model, we have to run `OFMC` with a bounded depth. Finally, we can conclude that the AVISPA models that we have developed are free from the attacks that AVISPA is able to find. Hence, the security goals of the two models that we have mentioned earlier have been validated.

```
  OFMC
  Version of 2     2 13
SUMMAR
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C: SPAN testsuite results n2n-freshness.if
 OAL
  as_specified
BACKEND
  OFMC
COMMENTS STATISTICS
  parseTime:  . 1s
  searchTime: 1.3 s
  visitedNodes: 454 nodes
  depth: 13 plies
```

Figure 13. Validation output of OFMC for node to node key agreement with strong freshness.

5. Denial-of-Service Attack on Node to Node Key Agreement Protocol

Although AVISPA could not find any attack on the models we have developed, by analyzing the node to node key agreement protocol, a potential denial-of-service (DoS) attack might be found. A DoS attack is always considered a threat for any kind of wireless network where jamming a network with a strong signal is always possible. Especially, this might be critical in sensor networks due to its use in life-critical applications [16]. Figure 14 shows a possible DoS attack on the node to node key agreement protocol, where the intruder is masquerading as the role of the node A. Pretending to be in the role of A, the intruder triggers the key agreement protocol and, thus, forces node B and server S to generate and exchange the rest of the three messages. Hence, the intruder may engage the server S and other nodes in the network to generate and exchange numerous messages. This is clearly a threat for a potential DoS attack, where the server is being engaged in useless message generation and exchange. Thus, the server denies legitimate nodes from receiving services that they have requested, *i.e.*, key generation and distribution.

$$
\begin{aligned}
&I(A) \rightarrow B : N_A, A \\
&B \rightarrow S : N_A, N_B, A, B, MAC(K'_{BS}, N_A||N_B||A||B) \\
&S \rightarrow A : \{SK_{AB}\}_{K_{SA}}, MAC(K'_{SA}, N_A||B||\{SK_{AB}\}_{K_{SA}}) \\
&S \rightarrow B : \{SK_{AB}\}_{K_{SB}}, MAC(K'_{SB}, N_B||B||\{SK_{AB}\}_{K_{SB}})
\end{aligned}
$$

Figure 14. DoS attack on the node to node key agreement protocol.

5.1. Simulating a DoS Attack in AVISPA

To simulate the DoS attack, we develop an AVISPA model of the node to node key agreement protocol (without any data transmission). We introduce an attacker by adding an agent `darth`, who sends the first message N_A, A to B by masquerading as the role of the node A. Figure 15 shows the protocol simulation generated by SPAN executing the AVISPA model we developed to simulate the node to node key agreement protocol. This MSC shows that the intruder (`darth`) can easily convince the server (`S`) and the other node (`B`) to generate and transfer the next three messages.

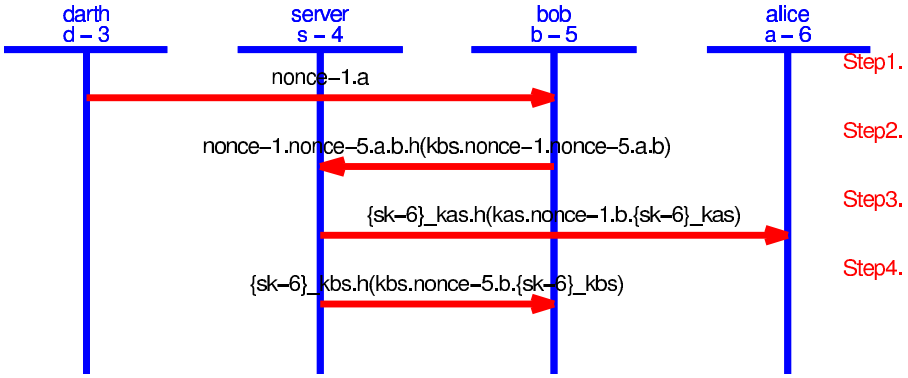

Figure 15. Protocol simulation for DoS attack on the node to node key agreement protocol.

5.2. Preventing the DoS Attack

The intruder is successful at generating the attack because the identity of the node (*i.e.*, node `A`'s identity) that commenced the message exchanges is not authenticated. Therefore, we propose a modified version of the protocol, which is presented in Figure 16. In this modified version, in the first message, `A` appends the MAC of the message (*i.e.*, N_A, A) at the end of the message. While generating this MAC, `A` uses the MAC key, K'_{AS}, shared between A and S. The second message, which is sent to S from B, has the following three components:

- the first message it receives from A, *i.e.*, $N_A, A, MAC(K'_{AS}, N_A||A)$;
- B adds its nonce and identity, *i.e.*, N_B, B;
- finally, B appends the MAC of the first two components using the MAC key, K'_{BS}, shared between B and S.

When S receives this message, S authenticates the messages generated by A and B. Since the MAC key, K'_{AS}, is shared between A and S, only A can generate the first message. As a result, no intruder is able to masquerade as the identity of A. In this modified version, any attack generated by the intruder will be detected in the second message, and thus, S will not generate any new symmetric key (SK_{AB}) for A and B. Thus, the third and the fourth messages will not be forwarded by S. This will definitely prevent the potential DoS attack on the key distribution protocol.

Note that there is a replay attack when the adversary intercepts the first message, keeps a copy of it and then replays it to B. Here, the adversary masquerades in the role of A. This attack is only possible if the nonce N_A in the first message is not fresh, *i.e.*, not generated from a random or pseudo-random

number. If the replay attack is not able to be prevented through the nonce, an increasing counter might be used instead of the nonce (note the similar technique being used in the counter exchange protocol in Section 4.3).

$$A \rightarrow B : N_A, A, MAC(K'_{AS}, N_A||A)$$
$$B \rightarrow S : N_A, A, , MAC(K'_{AS}, N_A||A), N_B, B, MAC(K'_{BS}, N_A||A||MAC(K'_{AS}, N_A||A)||N_B||B)$$
$$S \rightarrow A : \{SK_{AB}\}_{K_{SA}}, MAC(K'_{SA}, N_A||B||\{SK_{AB}\}_{K_{SA}})$$
$$S \rightarrow B : \{SK_{AB}\}_{K_{SB}}, MAC(K'_{SB}, N_B||B||\{SK_{AB}\}_{K_{SB}})$$

Figure 16. Proposed modification to prevent a DoS attack.

6. Related Work

Since the development of the SNEP protocol, a number of security analyses has been carried out. There are two popular methods of formal modeling and verification of security protocols: through an automated tool and algebraic specification techniques. Both methods have been applied in validating SNEP's security properties.

In [17], an AVISPA model has been developed to verify two security properties: authenticity and confidentiality of SNEP message components. Two separate models have been developed to verify the scenarios: requesting information from the base station to a normal node and key distribution between two nodes. In this analysis, an attack has been found, where an intruder masquerading in the role of the base station requests information from a node. It has been claimed that exploiting this attack, the intruder may receive confidential data from a node. However, this claim is not justified, since a node sends data to the base station in encrypted format (note that the node to base station data transmission protocol is explained in Section 2.2), and the intruder is not able to divulge the encryption key. The models we are presenting in this paper are different from the models explained in [17] in many ways that are explained here:

(1) We develop the model of the counter exchange protocol with strong freshness, which they did not.
(2) They develop a node to node key distribution followed by data transmission (in total, six messages are being exchanged), while we develop an integrated model (in total, five messages are being exchanged).
(3) Finally, the attack we discovered was not reported by the previous model.

A popular, open-source software verification tool, Spin [18], has been used to verify the confidentiality and authentication of SNEP [19]. In this study, a fine-state machine-based model for SNEP has been developed first. Next, the security properties (*i.e.*, authenticity and confidentiality) have been expressed using the linear temporal logic (LTL). Finally, these properties have been verified using Spin. Note that the model checking tool (*i.e.*, Spin) they have used is not particularly developed for security property validation. Thus, the focus of this paper is to show how Spin could be used in security property validation. Here, only the key distribution protocol SNEP has been modeled.

As we mentioned earlier, modal logic and algebraic specification techniques have been used to validate the security properties of a protocol. Here, we present two such efforts ([20,21]), where instead

of automated tools, the security properties of SNEP have been validated using a mathematical proof. In [20], the Coffey–Saidha–Newe (CSN) modal logic has been used to verify the SNEP key agreement sub-protocol. CSN [22] combines the logic of belief and knowledge during execution of a protocol and, thus, is capable of verifying both security and trust. The analysis has not found any weakness of the key agreement protocol and also has confirmed the property's strong security due to the use of nonces. Another algebraic specification technique, based on the observational transition systems (OTS)/CafeOBJ method, has been used to formally analyze SNEP's data transmission and node to node key agreement protocols [23]. In [21], a generalized Gorrieri and Martinelli's timed generalized non-deducibility on compositions (tGNDC) schema has been utilized for formal verification of well-known key management protocols for WSN, including μTESLA, Localized Encryption and Authentication Protocol (LEAP)+ and Lightweight Security Protocol (LiSP) .

Unlike SPINS, which is a network layer protocol, TinySec, a link layer security architecture, is fully implemented for the first time for WSNs [24]. TinySec is a lightweight protocol with a generic security package designed to impact minimally on performance. The security goals of TinySec protocols are access control, message integrity and confidentiality. In [25], these security properties have been verified by developing an AVISPA model. The verification process confirms that TinySec satisfies all of these security goals. However, AVISPA reports a replay attack where an intruder plays the role of the base station. This is expectable, since replay protection is intentionally excluded in TinySec and is left for the higher layer protocol stack.

By taking the virtue of TinySec, MiniSec has been designed to provide higher security and low energy consumption, as well [26]. Lower energy consumption has been achieved by increasing the memory size slightly. It has two operating modes: single-source communication and multi-source broadcast communication. The security goals it fulfills are confidentiality, data authentication, replay protection and weak message freshness. In [27], an AVISPA model has been developed to validate the authenticity and confidentiality of the MiniSec protocol. The validation has not reported any possibility of attacks.

7. Conclusions

Due to sensor nodes' resource constraint, widely-deployed cryptographic primitives (e.g., SSL, SSH), especially public key-based authentication protocols, are not feasible to use in sensor networks. Therefore, designing a secured communication protocol in a sensor network is a challenging task. An industry-scale automated tool, such as AVISPA, alleviates the daunting task of rigorous analysis of security protocols. The model we develop definitely establishes the security properties of SNEP. Moreover, the DoS attack we discovered could be minimized by the modification we propose. In the future, we will validate the security properties of μTESLA and other prominent sensor network protocols, including TinySec, MiniSec, *etc*.

Conflicts of Interest

The author declares no conflict of interest.

References

1. Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422.
2. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330.
3. Stankovic, J.; Wood, A.; He, T. Realistic Applications for Wireless Sensor Networks. In *Theoretical Aspects of Distributed Computing in Sensor Networks*; Nikoletseas, S., Rolim, J.D., Eds.; Monographs in Theoretical Computer Science. An EATCS Series; Springer: Berlin, Germany, 2011; pp. 835–863.
4. Sen, J. Security in Wireless Sensor Networks. In *Wireless Sensor Networks: Current Status and Future Trends*; Khan, S., Pathan, A.S.K., Alrajeh, N.A., Eds.; CRC Press: Boca Raton, FL, USA, 2012; pp. 407–460.
5. Perrig, A.; Stankovic, J.; Wagner, D. Security in Wireless Sensor Networks. *Commun. ACM* **2004**, *47*, 53–57.
6. Karlof, C.; Wagner, D. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Netw.* **2003**, *1*, 293–315.
7. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126.
8. Perrig, A.; Szewczyk, R.; Tygar, J.D.; Wen, V.; Culler, D.E. SPINS: Security Protocols for Sensor Networks. *Wirel. Netw.* **2002**, *8*, 521–534.
9. Perrig, A.; Canetti, R.; Tygar, J.; Song, D. Efficient authentication and signing of multicast streams over lossy channels. In Proceedings of IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 14–17 May 2000; pp. 56–73.
10. Armando, A.; Basin, D.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuellar, J.; Drielsma, P.H.; Heám, P.; Kouchnarenko, O.; Mantovani, J.; *et al.* The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Computer Aided Verification*; *Lecture Notes in Computer Science*; Etessami, K., Rajamani, S., Eds.; Springer-Verlag: Berlin, Germany, 2005; Volume 3576, pp. 281–285.
11. Vigan, L. Automated Security Protocol Analysis With the AVISPA Tool. *Electron. Notes Theor. Comput. Sci.* **2006**, *155*, 61–86.
12. Islam, S. Security Analysis of LMAP Using AVISPA. *Int. J. Secur. Netw.* **2014**, *9*, 30–39.
13. Heen, O.; Genet, T.; Geller, S.; Prigent, N. An Industrial and Academic Joint Experiment on Automated Verification of a Security Protocol. In Proceedings of the IFIP Networking Workshop on Mobile and Networks Security, Singapore, Singapore, 5–9 May 2008; pp. 39–53.
14. Glouche, Y.; Genet, T.; Houssay, E. *SPAN: A Security Protocol ANimator for AVISPA*; User Manual; IRISA/Université de Rennes 1: Rennes, France, 2006.
15. Dolev, D.; Yao, A. On the Security of Public-Key Protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208.
16. Wood, A.; Stankovic, J. Denial of service in sensor networks. *Computer* **2002**, *35*, 54–62.

17. Tobarra, L.; Cazorla, D.; Cuartero, F. Formal Analysis of Sensor Network Encryption Protocol (SNEP). In Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'07), Pisa, Italy, 8–11 October 2007; pp. 1–6.
18. Holzmann, G.J. *The SPIN Model Checker: Primer and Reference Manual*; Addison-Wesley: Boston, MA, USA, 2004.
19. Chen, W.; Xiao, W. Model checking and analyzing the security protocol for wireless sensor networks. In Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), Harbin, Heilongjiang, China, 12–14 August 2011; Volume 8, pp. 4093–4096.
20. Li, Y.; Newe, T. On the Formal Verification of the SNEP Key Agreement Protocol for Wireless Sensor Networks. In Proceedings of the International Conference on Sensor Technologies and Applications (SensorComm'07), Valencia, Spain, 14–20 October 2007; pp. 186–191.
21. Macedonio, D.; Merro, M. A Semantic Analysis of Key Management Protocols for Wireless Sensor Networks. *Sci. Comput. Program.* **2014**, *81*, 53–78.
22. Newe, T.; Coffey, T. Formal verification logic for hybrid security protocols. *Int. J. Comput. Syst. Sci. Eng.* **2003**, *18*, 17–25.
23. Ouranos, I.; Stefaneas, P.; Ogata, K. Formal Modeling and Verification of Sensor Network Encryption Protocol in the OTS/CafeOBJ Method. In *Leveraging Applications of Formal Methods, Verification, and Validation*; *Lecture Notes in Computer Science*; Margaria, T., Steffen, B., Eds.; Springer-Verlag: Berlin, Germany, 2010; Volume 6415, pp. 75–89.
24. Karlof, C.; Sastry, N.; Wagner, D. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD, USA, 3–5 November 2004; pp. 162–175.
25. Tobarra, L.; Cazorla, D.; Cuartero, F.; Díaz, G.; Cambronero, E. Model checking wireless sensor network security protocols: TinySec+ LEAP+ TinyPK. *Telecommun. Syst.* **2009**, *40*, 91–99.
26. Luk, M.; Mezzour, G.; Perrig, A.; Gligor, V. MiniSec: A Secure Sensor Network Communication Architecture. In Proceedings of the 6th International Conference on Information Processing in Sensor Networks, Cambridge, MA, USA, 25–27 April 2007; pp. 479–488.
27. Tobarra, L.; Cazorla, D.; Cuartero, F.; Díaz, G. Analysis of security protocol minisec for wireless sensor networks. In Proceedings of the IV Congreso Iberoamericano de Seguridad Informatica (CIBSI'07), Mar del Plata, Argentina, 26–28 November 2007; pp. 1–13.