

# Formally Verified Authenticated Query Dissemination in Sensor Networks

Frank Werner  
Computer Science Department  
University of Karlsruhe (TH)  
werner@kit.edu

Zinaida Benenson  
Computer Science Department  
University of Mannheim  
zina@uni-mannheim.de

**Abstract**—We consider the problem of authenticated query dissemination in sensor networks, where each sensor node should be able to decide whether the query was injected into the network by a legitimate entity (e.g., the base station), or by an adversary that tries to gain unauthorized access to the sensor data. We present *sAQF*, an improved variant of the *AQF* protocol by Benenson et al. [2] and verify its correctness by means of a model based on probabilistic action systems.

**Index Terms**—authentication, query, dissemination, formal quantitative analysis, probabilistic, verification, probabilistic model checking

## I. INTRODUCTION

### A. Motivation

Wireless sensor networks consist of small wireless devices (nodes) that measure and process environmental characteristics such as temperature, humidity, light, vibration. We assume that in order to gain information from the sensor network, a query should be sent to the nodes. The query is a message containing the information request, e.g., “What is the average temperature in the observed area?”.

Query dissemination in sensor networks needs to be secure, reliable and energy efficient. Reliability and energy efficiency issues are usually considered in the context of transport protocols [7], [16]. In this work we consider an aspect of secure query dissemination: restricting access to a sensor network only to authorized entities, e.g., to the base station or to legitimate users. In particular, only authorized entities should be able to communicate with the sensor nodes and ask them to reveal the observed data.

Authenticity of the query requires some additional information, called the *authenticator*, to be appended to the query. Only authorized entities can generate the authenticator that usually depends on the query and some additional secret information. By checking the correctness of the authenticator sensor nodes can ascertain the query’s legitimacy.

Most existing solutions to query authentication are deterministic, or more precisely, they allow any sensor node to ascertain the authenticity of the query with probability exponentially close to 1. However, this high assurance level comes at the cost of large authenticators and energy-consuming authenticator checking.

On the other hand, this work pursues a *probabilistic* approach. We allow each single sensor node to make mistakes in authenticator checking. That is, in the most general case, a node can fail to recognize a faked query with non-negligible probability, or it may falsely classify a legitimate query as fake with non-negligible probability. It turns out that relaxing the requirements in these ways allows for shorter authenticators and cheaper authenticator checking, which results in energy savings. In the sequel we make the following contributions:

- We present a more efficient variant of our previous probabilistic algorithm *AQF* (Authenticated Query Flooding) [2] for authenticated query dissemination. We call the new algorithm *sAQF* (simplified *AQF*).
- We present a theoretical analysis of *sAQF*.
- We develop a formal model based on *probabilistic action systems (PAS)* [11] and use it to

verify correctness claims made by the previous theoretical analysis.

## B. Related Work

1) *Query Authentication*: Query authentication is often considered in the literature under the name “broadcast authentication”. One of the first broadcast authentication protocols for sensor networks is  $\mu$ TESLA [12]. It uses efficient symmetric cryptography and has an authenticator size of 160 bits.  $\mu$ TESLA is the most efficient query authentication protocol to date. However, it requires global time synchronization and periodic re-initialization of protocol parameters at every sensor node, thus reducing the overall efficiency.

Digital signatures can also be used for broadcast authentication [13], [18]. Unfortunately, public key cryptography is several orders of magnitude slower than symmetric key cryptography and therefore, introduces significant delay.

Numerous papers on secure code update [9], [14], [6], [8] appeared in the last years. Secure code update is a variant of broadcast authentication where the data to be authenticated is a new program to be installed on the sensor nodes. This means that the data is known beforehand, and therefore, the full authentication information can be computed before the broadcast. Moreover, as code update is not a frequent operation, its efficiency is not as critical as the efficiency of query dissemination techniques. These properties make code update less relevant for our work. All proposals apply either hash chains or hash trees, or a combination of both techniques.

The protocol sAQF presented in this work reduces the authenticator generation and checking time more than twice compared to our previous protocol AQF [2] since e.g., the intermediate computation of key ids is no longer required. Therefore, whereas AQF had to trade off its communication efficiency against computation time, sAQF favorably compares to existing solutions in both communication and computation efficiency. Moreover, here we verify the correctness of the algorithm analysis using formal methods.

2) *Formal Methods*: Formal Methods are investigated in the context of probabilistic flooding proto-

cols in the work of [4] with respect of their general application. The authors focus on quantitative performance measures and provide background about PAS. Their modeled network system consisting of five nodes using a modeled channel behavior that accounts for collisions and delays.

In previous work [17] we analyzed the AQF protocol [2] for query authentication with respect to energy consumption of the sensor nodes. Under various parameter settings topology dependent energy draws were computed that – in contrast to simulation – reflect the true performance of the protocol by precise and accurate values. In this work, however, we do not consider energy consumption, but give a qualitative statement about the correctness of algorithm’s analysis.

## II. SIMPLIFIED AUTHENTICATED QUERY FLOODING (sAQF)

### A. System Model

We consider a sensor network consisting of some number of nodes and a base station. In order to gain information from the nodes, the base station injects queries into the network. Thus, all legitimate queries originate from the base station. We assume the existence of mechanisms for query dissemination, for example the query may be flooded (each sensor which receives the query for the first time forwards it to all its neighbors), or reach the relevant nodes using some previously established structure, e.g., a routing tree. The most important assumption is that the query cannot reach all nodes directly and follows a multi-hop route. This assumption is crucial for probabilistic query flooding algorithms.

### B. Adversary

The goal of the adversary is to send arbitrary queries into the sensor network. To do this, the adversary will try to fake the authenticator of the query, such that sensor nodes accept the query as originated from the base station.

The adversary can compromise  $\tilde{n}$  sensor nodes out of  $n$ . This means that the adversary knows data and program of the compromised nodes, including all their secrets, e.g., the cryptographic keys. It can coordinate all compromised nodes such that they

possess all knowledge of the adversary. Moreover, the adversary can also arbitrarily reprogram the compromised nodes.

However, if the node is not compromised, the adversary has no means to find out its cryptographic keys, that is, the adversary cannot break cryptographic mechanisms if they are considered secure according to the current knowledge.

### C. Problem Definition

We firstly formally define the problem of authenticated query dissemination. We call it *Authenticated Query Flooding* in order to emphasize the multi-hop nature of considered sensor networks.

*Definition 1 (Authenticated Query Flooding):*

Let *WSN* be a sensor network. After receiving a query, each sensor node *decides* whether the query comes from the base station. If its decision is positive, we say that the node *accepts* the query, otherwise the node *rejects* the query. Consider an arbitrary query  $q$ . The *WSN* design satisfies *authenticated query flooding (AQF)* if it satisfies the following properties:

- (Safety) If a node  $s$  in *WSN* accepts the query  $q$  as a legitimate query, then  $q$  is a legitimate query, e.g.,  $q$  was originated by the base station.
- (Liveness) Any legitimate query  $q$  will be received by all nodes in *WSN*.

As we consider probabilistic algorithms for AQF, we actually allow that safety and liveness properties are satisfied with sufficiently high probabilities.

### D. The sAQF Algorithm

In our protocol, we use message authentication codes (MACs) with a single bit output. The idea of using 1-bit MACs for broadcast authentication originates from [3]. We view a 1-bit MAC under a given key as a random function, i.e., we require the following:

- A single 1-bit MAC (under an unknown random key) cannot be feasibly guessed with any probability significantly exceeding  $\frac{1}{2}$ .
- Similarly, an  $m$ -bit string of  $m$  1-bit MACs under  $m$  independent random keys cannot be

guessed with probability significantly more than  $\frac{1}{2^m}$ .

1-bit MACs can be constructed from the usual MACs by taking their first bit of output. We also assume that all nodes and the base station can compute a hash function  $h(\cdot)$ . In the following, we describe the protocol.

*a) Base station:* The base station knows  $\ell$  symmetric keys  $key_1, \dots, key_\ell$ . This collection of keys is called *key pool*.

The base station first computes the query  $q$  and a hash  $x = h(q)$  of the query using a hash function  $h(\cdot)$ . Then it generates  $\ell$  1-bit MACs on  $h(q)$ . We call these  $\ell$  1-bit MACs *authenticator* for  $q$ , denoted as  $macs(q)$ .

Finally, the base station floods the query  $q$  into the sensor network, accompanied by the authenticator for the query.

*b) Sensor nodes:* Each node knows  $k$  keys from the key pool that are selected randomly and independently for each node according to a uniform probability distribution. The nodes know the IDs of their keys. A node's collection of keys is called *key ring*.

Upon receiving a new query  $q$  with the authenticator  $macs(q)$ , each node  $s$  checks the  $k$  1-bit MACs from  $macs(q)$  using its  $k$  keys. If all checked MACs are correct, the node forwards the query to its neighbors according to the underlying flooding mechanism or transport protocol. Otherwise, the query is dropped.

Pseudo-code descriptions of the algorithms for the base station and the sensor nodes are given in Algorithms II.1 and II.2.

---

**Algorithm II.1:** Base Station: sAQF-generate (Query  $q$ , KeyPool ( $key_1, \dots, key_\ell$ ))

---

```

 $x = h(q)$  /* compute the hash value */
 $macs(q) = (mac_1, \dots, mac_\ell) =$ 
(1-bit-MAC( $key_1, x$ ),  $\dots$ , 1-bit-MAC( $key_\ell, x$ ))
return  $macs(q)$ 

```

---

### E. Evaluation of sAQF

If the adversary knows some keys of the key pool through the node compromise, and tries to

---

**Algorithm II.2:** Sensor Nodes: sAQF-check ( $q$ ,  $macs(q)$ , KeyRing ( $key_{r_1}, \dots, key_{r_k}$ ))

---

```

 $x = h(q)$  /* compute the hash value */
for  $i = 1$  to  $k$  do
  if 1-bit-MAC( $key_{r_i}, x$ )  $\neq mac_{r_i}$  then
    return false /* reject the query */
  end if
end for
return true /* forward the query */

```

---

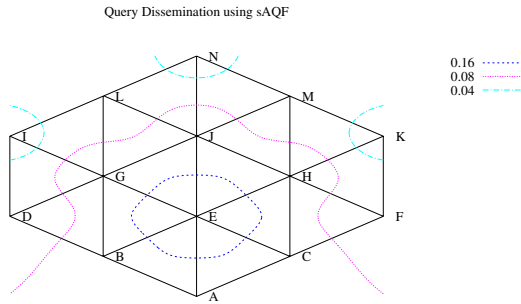


Fig. 1: Probability for a node to be reached by a fake query using parameter  $p_f = 0.3$ .

guess the unknown 1-bit MACs, then there is some probability  $p_f$  for each node to accept a fake query.

A sample topology for the proposed query dissemination using probabilistic authenticator checking is displayed in Figure 1. The lines interconnecting nodes represent the routing information. Node  $E$  represents the node that initially receives the fake query. Each node accepts the fake query with probability  $p_f = 0.3$ . While the likelihood of accepting the fake query at node  $E$  is relatively high, it drops rapidly for surrounding nodes.

This paper focuses on theoretical analysis of sAQF using formal methods. However, we also implemented sAQF for TmoteSky sensor nodes [15] running TinyOS [10], and evaluated it using the simulator TOSSIM provided by TinyOS. Moreover, we theoretically analyzed optimal parameter sizes for different network settings [1]. Due to space limit we cannot present the full results here. As a showcase, consider the authenticator size of 200 bits, and each node having 4 keys. The query is

disseminated in a network where each node has 6 neighbors on average, and the adversary captured 30 nodes out of 600. In this case, the propagation of the fake query is restricted to less than 10 nodes after the fake query is introduced into the network at some node. Authenticator checking takes around 30 ms in this case.

We conclude that sAQF is currently the most efficient query authentication solution that is not based on global time synchronization, such as in case of the currently most efficient query authentication protocol  $\mu$ TESLA [12]. In comparison to [2] it is more efficient since no key ids are generated.

sAQF provides immediate authentication of an arbitrary number of messages (without the need in protocol re-initialization), and the impact of adversarial attacks is restricted to a small part of the network. The price that one has to pay for all these good properties is the probabilistic nature of authentication. In the following, we give an analysis of sAQF and verify it using formal methods.

### III. ANALYSIS OF PROBABILITY TO ACCEPT A FAKE QUERY

#### A. Analysis by Random Set Theory

Assume that the adversary captured  $\tilde{n}$  nodes and, therefore, knows their key rings. The adversary computes its query  $Q$  and tries to compute the corresponding authenticator.

Consider the  $i$ -th authenticator bit. If the adversary knows  $key_i$ , it can compute the corresponding 1-bit-MAC. If the adversary does not know  $key_i$ , its best strategy is to guess the corresponding 1-bit-MAC with probability  $\frac{1}{2}$ .

Assume that node  $s$  which is not compromised by the adversary receives the faked query and starts verification of some authenticator bit  $b_i$ . We firstly compute the probability  $p_{bit\_known}$  that the adversary was able to compute  $b_i$  due to its partial knowledge of the key pool. The probability that one of the captured nodes does not have the corresponding key  $key_i$  in its key ring is

$$\frac{\binom{\ell-1}{k}}{\binom{\ell}{k}} = \frac{\ell - k}{\ell} = 1 - \frac{k}{\ell} \quad (1)$$

as there are  $\binom{\ell}{k}$  possibilities to choose a key ring, and  $\binom{\ell-1}{k}$  possibilities to choose a key ring which does not contain a particular key. As the nodes are captured independently, the probability that none of the captured nodes knows  $key_i$  is

$$\left(1 - \frac{k}{\ell}\right)^{\tilde{n}} \quad (2)$$

Thus, the probability that at least one captured node knows  $key_i$  is

$$p_{bit\_known} = 1 - \left(1 - \frac{k}{\ell}\right)^{\tilde{n}} \quad (3)$$

On the other hand, the adversary does not know  $b_i$  with probability  $1 - p_{bit\_known}$ . In this case, the adversary has to guess the bit. Thus, it guesses  $b_i$  with probability

$$p_{guessed} = \frac{1}{2} \left(1 - \frac{k}{\ell}\right)^{\tilde{n}} \quad (4)$$

Putting together Equations (3) and (4), the adversary computes  $b_i$  correctly with probability

$$1 - \frac{1}{2} \left(1 - \frac{k}{\ell}\right)^{\tilde{n}} \quad (5)$$

Therefore, the adversary correctly computes all  $k$  bits which a particular node can verify with probability can be well approximated by

$$p_f = \left(1 - \frac{1}{2} \left(1 - \frac{k}{\ell}\right)^{\tilde{n}}\right)^k. \quad (6)$$

### B. Analysis via Evolution of the Hyper-Geometric Distribution

The above probability computation is not fully correct, since the authenticator bits  $b_i$  from Formula (5) are not independent from each other. This matter can be clearly seen with the following illustration. If each node has one key ( $k = 1$ ) and one node is captured ( $\tilde{n} = 1$ ) then by Formula (6) there is a non-zero probability that the attacker does not know any key. Otherwise  $k = \ell$  would hold.

Due to this reason, we provide a more accurate and consequently more complex analysis.

We start by calculating with which probability the adversary acquires how many keys of the key-pool  $\ell$ . Nodes are assumed to be captured one by one. Afterwards it reads the node's data and adds the newly gained keys to its keypool.

Let

- $X_j$  := number of keys known by an adversary after capturing  $j$  nodes
- $X'_j := X_j - X_{j-1}$  = number of keys gained by an adversary on capturing the  $j$ -th node.

It is obvious that

$$P(X_0 = b) = \begin{cases} 1 & b = 0 \\ 0 & b \neq 0 \end{cases}$$

and

$$P(X_1 = b) = \begin{cases} 1 & b = k \\ 0 & b \neq k \end{cases}$$

Keys deployed on one node are independent of the keys that another node has. Therefore keys possessed by some node are independent of the keys the adversary gained by capturing other nodes. That is why the number of keys gained through capturing the next node  $X'_j$  only depends on the number of keys the adversary already had before  $X_{j-1}$ .

$X'_j$  can be described by the *hyper-geometric distribution*, with the total number of keys  $\ell$  as the whole set, the number of keys *marked* as unknown to the adversary  $\ell - X_{j-1}$  and the number of keys *drawn* with one node  $k$ :

$$\begin{aligned} P(X'_j = b | X_{j-1} = a) &= h(b|l, l - a, k) \\ &= \frac{\binom{l-a}{b} \binom{a}{k-b}}{\binom{l}{k}} \end{aligned} \quad (7)$$

Since  $X_j = X'_j + X_{j-1}$  and  $X'_j \in \{0, \dots, k\}$  we get:

$$\begin{aligned} P(X_j = b) &= \\ & \sum_{a=b-k}^b P(X'_j = b - a, X_{j-1} = a) = \\ & \sum_{a=b-k}^b P(X'_j = b - a | X_{j-1} = a) P(X_{j-1} = a) \end{aligned} \quad (8)$$

In the next section, we verify the above formulas by means of formal methods.

#### IV. FORMAL ANALYSIS BY MEANS OF PAS

The formalism of PAS allows state-based modeling of reactive systems, where upon some external stimuli, probabilistic transition taking is triggered. Such a system's model is build in the following.

##### A. Abstraction and Refinement of the Model

Two processes are used to imitate the scenario in which an adversary tries to construct a query. This fake query is then send to a proper node to test the acceptance of the authenticator. For this purpose two processes are modeled and combined using PAS [11] and evaluated with Prism [5]. After the model is completed, the verification is started by defining a PCTL query. The outcome of the verification results is afterwards matched with results of Formula (6) and (8).

1) *Global System Definition:* The following global variables are used in addition to the probabilistic system description to analyze the model. We use for the assignment of an array index  $n_i = 1$  as an abbreviation for setting the value of array  $n$  at position  $i$  to value 1, i.e.,  $n[i] = 1$ . We write  $n_{0,\dots,\ell-1} = 0$  to stress that all indices of array  $n$  are set to 0, i.e.,  $n[0] = 0, n[1] = 0, \dots, n[\ell - 1] = 0$ . Arrays  $n$ ,  $u$ , and  $a$  are read- and writable by all processes to establish a global inter-process communication. The parameters externally set by the user originate from the protocol. We use  $k$  as the number of keys per sensor node and  $\tilde{n}$  to denote the number of nodes captured by an adversary. As the length of the authenticator  $b$  denotes the number of bits to be checked. The final system is instantiated using the parallel composition without renaming as

$$System \hat{=} Node \parallel Adversary$$

2) *Node Process:* The *Node* process represents the sensor network device (see Figure 2a) which receives a faked query from the adversary. Since the key distribution is randomly and independent for all sensor nodes this process is modeled as follows: The *Node* process has variable  $\tilde{n}_{0,\dots,\ell-1}$  representing the key ring of one particular node. After the

initialization phase is completed, the counter  $i$  is set to 0 and all  $k$  array values are assigned to 2. This value represents an undefined or unassigned value in the model. For the remainder, the node process chooses non-deterministically  $k$  bits out of its  $\ell$  unassigned array locations. When doing so a chosen array entry is set to either 0 or 1 with probability  $\frac{1}{2}$ . Subsequently the node holds  $k$  keys which are assigned at random chosen locations in the array as required by the protocol description. After this step the node processes terminates. In terms of a probabilistic system the description of the node model is shown in Figure 2a.

3) *Adversary Process:* The process imitating the adversary behavior has two arrays (see Figure 2b). That is array  $a_{0,\dots,\ell-1}$  containing the keys already known and correctly probed and array  $u_{0,\dots,\ell-1}$  used to keep track of the captured bits in the current round. The values assignable to array  $a_{0,\dots,\ell-1}$  are 2 for *unknown bit*, 3 for *correct bit*, and 0 or 1 if the *bit is guessed* with the corresponding value.

Similar to the process from above, randomly chosen locations are assigned with value 3 in the adversary process to denote correct bits captured from a node. In the first round  $k$  bits are guessed. By the use of array  $u$  (*unique bits per node*) we keep track that no bits are double assigned within one round. Not doing so would mean that a particular bit is read twice and in effect the overall results would be distorted since each node can only contribute with each bit only once. Whenever within one round a bit is used on a sensor it is indicated by setting the corresponding bit at array  $u$ . In addition we check by the guard  $a_i \neq 2$  in line 5 of Figure 2b whether the bit of the compromised node is already known to the adversary. In this case the bit does not need to be updated.

After all possible transitions are taken and consequently all  $k$  bits of the first compromised node are read, we proceed with the second captured node by incrementing the round counter  $h$ . In this round the same task is performed, namely filling up the bits an adversary knows from a legitimate sensor node. After  $k$  consecutive runs there is a high chance that some of the locations of array  $a$  are still empty ( $a_i = 2$ ). For all of these locations bits are

randomly guessed, i.e., with probability 0.5 value 1 is assigned and otherwise value 0. If all unknown bits are guessed we end the Adversary Process and check whether the generated authenticator would be accepted by a random node key  $k$ .

4) *PCTL Property Checking*: The PCTL Formula in Figure 2c denotes the probability of a faked query being accepted by a legitimate sensor node as a single numerical value.

Essentially it computes the probability to reach a state in which the properties expressed by the conjunction over all bits that contribute to the authenticator. The conjunction term reads as follows: if  $a_i = 3$  so the adversary knows the right bit *true* is returned. Otherwise if  $n_i = 2$  then the node does not know bit  $i$  and consequently it cannot tell whether the authenticator bit is correct or not. Last, if  $a_i = n_i$  then the adversary guessed the same bit that the node has. So the node will falsely accept the query send by the adversary and the probability to reach that state is returned. In addition the *deadlock* statement is added to stress that the desired state is final i.e., no further execution steps are possible. If all of the above failed, then the node discovers a faked authenticator.

### B. Correctness of the sAQF Analysis

The results of the formal model are compared against Formulas (6) and (8) from section III. W.l.o.g. the number of bits used is restricted to 8 while varying  $\tilde{n}$  between 0 and 8. For parameter  $0 \leq k \leq 8$  series are computed using the Prism tool [5], a probabilistic model checker. The plot is displayed in Figure 3 showing that the results obtained by the hyper-geometric Formula (8) and the formal verification model perfectly equal. On the other hand, results using the *random-set theory* based Formula (6) differ heavily for small  $\tilde{n}$ .

Considering the figure in more detail, for  $\tilde{n} \leq 1$  there is a discrepancy between the correct value and the formula. At the beginning ( $\tilde{n} = 1$ ) this fault is relatively high but is vanishing for higher values of  $\tilde{n}$ . Also notably is that for values of  $k = 1$  the failure does not show up. This error stems from the neglected dependence relation during the derivation of the random-set formula, as stated before.

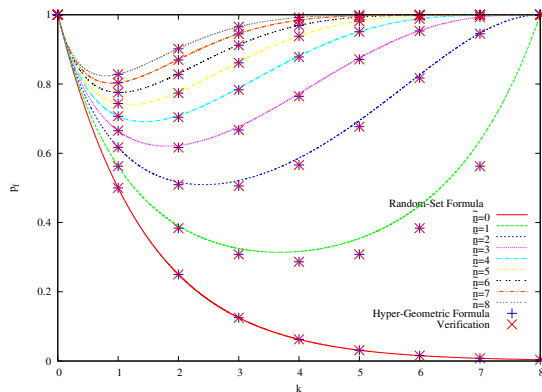


Fig. 3: Results obtained by *Random-Set theory*, *Hyper-Geometric distribution*, and *verification*.

### C. Differences between two formulas

We approached the analysis of sAQF from two distinct directions, each with its advantage and respective drawbacks. Using *random set theory* the results are overestimated using Formula (6) due to a violated dependability constrain. On the other hand this approach provides a compact formula that can easily be used to compute the probabilities. Dependability becomes negligible for large  $\ell$  and small  $k$  sizes, such as  $\ell = 200$  and  $k = 4$  that should be used in practice, as presented in Section II-E.

The *hyper-geometric* Formula (8) delivers the precise data at cost of complexity. Due to the recursive nature, it is complex to compute probabilities by hand.

## V. CONCLUSION

We proposed an improved algorithm for the authenticated query flooding and analyzed it by means of random set theory. By the use of the formal approach via PAS model checking, errors in the analysis were found, which resulted in the development of more precise analysis using the hyper-geometric distribution.

For the future work, we are developing a model that integrates a higher level of detail. This refinement step is in particular useful to make quantitative statements about the probabilistic aspects of liveness and safety that were not analyzed with the actual model.

$$\left( \begin{array}{l} \text{var } i : \{-1, \dots, k\}, n_j : \{1, \dots, 2\} \\ \text{initially } i := 0, n_j := 2 \\ \square_{i: A} \text{action}_i : (i < k \wedge n_i = 2) \rightarrow (n_i := 1 \wedge i := i + 1) \oplus_{0.5} (n_i := 0 \wedge i := i + 1) \end{array} \right)$$

(a) Node Process

$$\left( \begin{array}{l} \text{var } j : \{-1, \dots, k\}, h : \{-1, \dots, n\}, u_j : \{1, \dots, 2\} \\ \text{initially } i := 0, n_j := 2 \\ \square_{i: A} \text{action}_i : (j = k \wedge h < n) \rightarrow (u_i := 0) \\ \square_{i: A} \text{action}_i : (i < k \wedge n_i = 2) \rightarrow (n_i := 1 \wedge i := i + 1) \oplus_{0.5} (n_i := 0 \wedge i := i + 1) \\ \square_{i: A} \text{action}_i : (j < k \wedge u_i = 0 \wedge a_i \neq 2 \wedge h < n) \rightarrow (u_i := 1 \wedge j := j + 1) \\ \square_{i: A} \text{action}_i : (h = n \wedge j = 0 \wedge a_i = 2) \rightarrow (a_i := 0 \oplus_{0.5} a_i := 1) \end{array} \right)$$

(b) Adversary Process

$$P = ? [ \text{true} U \wedge (\text{"deadlock"}) \wedge_{i=1}^{\text{bits}} (\text{"} a_i = 3 ? \text{true} : (n_i = 2 ? \text{true} : (a_i = n_i ? \text{true} : \text{false})) \text{"}) ]$$

(c) PCTL Formula

Fig. 2: PAS Models for adversary process, node process, and the PCTL formula solving the PAS.

## REFERENCES

- [1] Z. Benenson. *Access Control in Wireless Sensor Networks*. PhD thesis, University of Mannheim, 2008.
- [2] Z. Benenson, F. C. Freiling, E. Hammerschmidt, S. Lucks, and L. Pimenidis. Authenticated query flooding in sensor networks. In *Security and Privacy in Dynamic Environments, Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006)*, pages 38–49, 2006.
- [3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. IEEE INFOCOM'99*, volume 2, pages 708–716, New York, NY, Mar. 1999. IEEE.
- [4] A. Fehnker and P. Gao. Formal verification and simulation for performance analysis for probabilistic broadcast protocols. In *Proceedings of the 5th International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW'06)*, volume 4104 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2006.
- [5] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer, 2006.
- [6] S. Hyun, P. Ning, A. Liu, and W. Du. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. *International Conference on Information Processing in Sensor Networks*, pages 445–456, 2008.
- [7] H. Karl and A. Willig. Data transport reliability in wireless sensor networks – a survey of issues and solutions. *Praxis der Informationsverarbeitung und Kommunikation*, 28:86–92, April 2005.
- [8] D. H. Kim, R. Gandhi, and P. Narasimhan. Castor: Secure code updates using symmetric cryptosystems. *Real-Time Systems Symposium, IEEE International*, 0:479–488, 2007.
- [9] I. Krontiris and T. Dimitriou. Authenticated in-network programming for wireless sensor networks. In *In Proceedings of the 5th International Conference on AD-HOC Networks and Wireless (ADHOC-NOW 2006)*, 2006.
- [10] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The emergence of networking abstractions and techniques in tinyos. In *First USENIX/ACM Symposium on Networked Systems Design and Implementation*, 2004.
- [11] A. McIver. Quantitative refinement and model checking for the analysis of probabilistic systems. In *In FM 2006, volume 4085 of LNCS*, pages 131–146. Springer Verlag, 2006.
- [12] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.
- [13] K. Ren, K. Ren, W. Lou, W. Lou, Y. Zhang, and Y. Zhang. Multi-user broadcast authentication in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 223–232, 2007.
- [14] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. SCUBA: Secure Code Update By Attestation in sensor networks. In *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security*, pages 85–94, New York, NY, USA, 2006. ACM.
- [15] Tmote Sky datasheet. Available at <http://www.moteiv.com>.
- [16] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, 2006.
- [17] F. Werner and P. H. Schmitt. Analysis of authenticated query flooding by probabilistic means. In *The Fifth Annual Conference on Wireless on Demand Network Systems and Services (WONS 2008)*, pages 101–104, Jan. 2008.
- [18] S. Yoon, T. Asano, M. Kusakawa, H. Lee, and K. Kim. Hybrid multi-user broadcast authentication for wireless sensor networks. In *The 2008 Symposium on Cryptography and Information Security (SCIS 2008)*, Japan, January 2008.