EPiC Series in Computing
Volume 85, 2022, Pages 36–47
Proceedings of 43rd Conference of the South African Institute of Computer Scientists and Information Technologists

Memory Requirements for the Detection of Impostor Nodes in Wireless Sensor Networks[*]

Stefan Gruner[†] Dylan Krajnc[‡] Johan Pieter van Rooyen[§]
Department of Computer Science
University of Pretoria
Republic of South Africa

Abstract

This paper shows how it is at least in principle possible to detect impostor nodes in wireless sensor networks with a quite simplistic detection algorithm by purely statistical means and merely from external observation without any knowledge of the impostor's internal composition. This method, however, requires considerable volumes of internal memory for any WSN node on which such detection algorithms are supposed to be implemented.

Keywords: Wireless Sensor Networks · WSN Reliability · WSN Integrity · WSN Safety and Security · WSN Trustworthiness · Node Replication Attacks · Intrusion Detection · Probabilistic Finite Automata · Randomised Detection Algorithm · Detection Accuracy · Memory Requirements

1 Introduction

Because wireless sensor networks (WSN) communicate by means of radio waves amongst their member nodes, it is possible (at least in principle) for any hostile 3rd party to smuggle an impostor node into such a WSN: such an event is called a *node replication attack* [13]. The purpose of such a maliciously infiltrated impostor node could be the theft of valuable data from the WSN, and/or the 'feeding' of the WSN with wrong, misleading, confusing or disturbing pseudo information.

To remain undetected as long as possible, the impostor must act as similarly as possible as all the other nodes of the WSN, though from time the impostor must also clandestinely communicate with some hostile 'kingpin' (outside the WSN) in order to exfiltrate the stolen data. Due to this additional clandestine activity of the impostor, there should be a tiny but nonetheless observable *statistical difference* between the 'behavioural pattern' of the impostor node in comparison against the 'behavioural pattern' of the WSN's genuine nodes. Without

[*]This paper is based on two preliminary (unpublished) academic project reports prepared by the two co-authors under the guidance of the corresponding author. *We dedicate our work to the memory of our late colleague Patricia Lutu, †May 2022.*

[†]Corresponding author: E-mail sg@cs.up.ac.za, ORCID 0000-0001-6008-6123

[‡]E-mail dylan.krajnc@tuks.co.za

[§]E-mail johan.vanrooyen@tuks.co.za

any such difference the impostor would remain theoretically undetectable by the genuine nodes of the WSN. However, because the observable difference between the behaviour of the impostor and the behaviour of the genuine nodes is only tiny, the question arises for how long (how much time) the impostor must be observed before the difference gets statistically significant enough to arouse the 'suspicion' of the WSN's genuine nodes. Because of the close correlation between 'time' and 'space' in matters of computation, the above-mentioned question can be transformed into the question of how much memory (internal storage) a genuine WSN node needs in order to be able to store a long-time observation 'trace' from which a 'suspicion' can arise with high plausibility and reliability. This is the question with which we are dealing in this paper.

However because the externally observable behaviour of WSN nodes is in principle *non*-deterministic any answer to such a question can merely be a statistical answer without any absolute certainty — though hopefully still without too many false-positive α-errors ('friend' node wrongly suspected to be an impostor) as well as without too many false-negative β-errors (impostor node wrongly accepted as 'friend').

The remainder of this paper motivates and explains the *models and methods* by means of which we tentatively tackled the above-mentioned problem, presents and discusses the *results* we obtained from the experiments which we designed on the basis of those considerations, and points to various possibilities of further improving our models, methods, techniques and results in 'future work'. Some 'related work' is briefly recapitulated in the semi-final section before the conclusions.

2 Method

2.1 The Casino Analogon

Due to the well-known *law of the large numbers* in stochastics [25], we know that any growing sequence of random numbers approaches its 'average' expectation with certainty *if* those random numbers belong to a random distribution for which a stochastic expectation value exists at all. In a *fair* gambling casino, for example, in which *dice* games are played with dices of six faces (that are showing the numbers $1 \ldots 6$),

- a fair dice must show each of its six faces with the same probability $p = \frac{1}{6}$,
- and the *theoretical* expectation value of such a setting is $\frac{1+\cdots+6}{6} = \frac{7}{2} = 3,5$.

In such a context, the well-known *law of the large numbers* tells us that for a growing sequence of fair dice rolls the actually observed average value of the sequence *must* approach the theoretical expectation value (which indeed exists for the probability distribution of this example) with certainty — in other words:

$$n \overset{\lim}{\to} \infty \, [\tfrac{f_1+\cdots+f_n}{n}] \overset{!}{=} 3{,}5$$

where each $f_i \in \{1 \ldots 6\}$ is a face of the dice ($\forall i \in \{1 \ldots n\}$). This law thus enables the croupier of the casino to determine *empirically* and with ever growing confidence whether any given *unknown* dice is fair, because for an *unfair* dice (for which $\exists u_x, u_y \in \{1 \ldots 6\}$ with $u_x \neq u_y$ and $p(u_x) \neq p(u_y)$) the croupier will detect *necessarily* that

$$n \overset{\lim}{\to} \infty \, [\tfrac{u_1+\cdots+u_n}{n}] \neq 3{,}5$$

although an epsilon-small ‘residue of uncertainty’ will always remain for any finite $n < \infty$.

To bridge the conceptual gap between this casino scenario with its fair versus unfair dices, we show briefly how such a dice (with its observable faces) can be modelled as a *probabilistic finite automaton* (PFA) or *stochastic finite automaton* (SFA) that has —in this casino example— only one internal state (d, as the dice is a solid object without any internal mechanism) and six transitions from that internal state to itself.[1]

In the subsequent sections of this paper we also model WSN nodes in the form of such SFA — albeit with more than one internal state and different transitions between them. The formal definition of such SFA is well-known [20], and it is also known that such stochastic ‘machines’ are to some extent identifiable or ‘learnable’ (albeit only with considerable computational difficulties) by means of Artificial Intelligence techniques on the basis of nothing more than the external observation of the empirically observable ‘words’ which such SFA ‘emit’ to public visibility while they internally ‘hop’ probabilistically from internal state to internal state [20]. If two different SFA, A', A'', correspond structurally to the same DFA [5], A, then their ‘pure’ languages are all the same: $\mathcal{L}(A') = \mathcal{L}(A'') = \mathcal{L}(A)$, though their individual words have different probabilities in the cases of A' versus A'' (whereas the words of A do not have any probabilities at all). Thus, $\exists w \in \mathcal{L}(A)$: $p_{A'}(w) \neq p_{A''}(w)$, such that (due to the above-mentioned *law of the large numbers*) the difference between A' and A'' will be empirically observable after a sufficiently long duration of observation *even if* the *internal* structures of those machines remain hidden and unknown.

In our motivating casino scenario, the *fair* dice corresponds obviously to the one-state SFA with the following transition table (**1**),

Table 1

$\sigma \in \Sigma$	*current* state	transition *probability*	*successor* state
‘1’	d	p	d
‘2’	d	p	d
‘3’	d	p	d
‘4’	d	p	d
‘5’	d	p	d
‘6’	d	p	d

with all $p = \frac{1}{6}$, whereas the transition table of the SFA of any *unfair* dice must have at least two rows in which $p' \neq p''$. As we are interested in observing visible output ‘streams’ (words) of arbitrary length, it is *irrelevant* for the purposes of this paper whether any state s of a given SFA is an ‘accepting’ state; (alternatively: *every* state is an ‘accepting’ state).

2.2 SFA Models of WSN Nodes

From the concrete WSN node example of [30] we have abstracted the formal structure of a finite automaton the transition table of which is shown below (**2**). Thereby we followed the method of abstraction described in [29]. On the basis of [30] our finite automaton of a ‘normal’

[1] The theory of probabilistic automata (with its many decidability or undecidability results) is well established at least since the early 1960s [27]. Equally well known is the close theoretical relationship between such automata and Markoff chains [14] via the concept of Markoff automata [18], such that already available stochastic and statistical analysis techniques for Markoff chains [9][24](Ch. 2) can be made fruitful for the theoretical analysis of probabilistic automata, too. The discussion of such statistical methods themselves, such as (for example) the χ^2 adaptation test that constructively establishes a notion of statistical ‘significance’ [10], cannot be included into the scope of this paper.

WSN node has the following seven internal (un-observable) states: q_0 start state ‘sleeping’, q_1 ‘announcing’, q_2 ‘listening’, q_3 ‘forwarding foreign message type <A>’, q_4 ‘forwarding foreign message type <C>’, q_5 ‘processing’, and q_6 ‘sending own message’. Thus: at each time-unit, the node does something. This ‘doing’ is represented by a state transition in the SFA which models the node. Moreover, each transition is associated with an observable output symbol. Thus, a period of stochastic operation of a node is represented by a *w*ord produced by the SFA. In other words, each random word is a description of the observable behaviour of a node over a finite period of time.

For our automaton’s alphabet Σ, the symbols of which represent the node’s *externally observable communication* events, we also consulted [6] in which the various types of distinguishable messages (including their header parts, their body parts, their closure parts) are described. In our alphabet we have the following six ‘visible’ symbols: a_h ‘header of a message of type <A>’, a_b ‘body of a message of type <A>’, a_c ‘closure of a message of type <A>’, c_h ‘header of a message of type <C>’, c_b ‘body of a message of type <C>’, and c_c ‘closure of a message of type <C>’, whereby <A> represents some ‘announcement’ message and <C> represents some message about some ‘sensor event’.

Moreover, in our model-design the *special symbol* $\epsilon \in \Sigma$ represents *one time-unit of silence* during which the node is internally busy without emitting any message. Though the node does not emit anything during that period of time, the phenomenon of its silence is externally observable, too, which justifies our model-design of a specific symbol for this phenomenon. For example: the string ‘$\epsilon\epsilon$’ represents the observable passing of *two* time-units of silence. For the conceptual purposes and the research questions of this paper the actual ‘physical’ duration of ‘one unit of time’ is not relevant.

Table 2

$\sigma \in \Sigma$	*current* state	transition *probability*	*successor* state
ϵ	q_0	0.9	q_0
a_h	q_0	0.1	q_1
a_b	q_1	0.9	q_1
a_c	q_1	0.1	q_2
ϵ	q_2	0.6	q_0
ϵ	q_2	0.1	q_1
ϵ	q_2	0.1	q_2
a_h	q_2	0.1	q_3
c_h	q_2	0.05	q_4
ϵ	q_2	0.05	q_5
a_c	q_3	0.1	q_2
a_b	q_3	0.9	q_3
c_c	q_4	0.1	q_2
c_b	q_4	0.9	q_4
ϵ	q_5	0.1	q_2
ϵ	q_5	0.8	q_5
c_h	q_5	0.1	q_6
c_c	q_6	0.1	q_2
c_b	q_6	0.9	q_6

The probability values ($0 \le p_i \le 1$) attached to the various transitions of our automaton could not be retrieved from any of the above-mentioned papers [6][29][30]. At that point we had to apply some 'common sense' intuitively in order to arrive at a reasonable and plausible formal model.[2] As far as the observable behavioural difference between a normal member node and an impostor node in a WSN is concerned, we assumed that the impostor 'wants' to imitate the observable behaviour of regular nodes as perfectly as possible whilst also having to communicate confidentially with some 'kingpin' outside the WSN (on a different radio frequency for which the regular WSN nodes do not have any sensor). Thus, from the perspective of a regular WSN node, the impostor's hidden communication with the 'kingpin' is quasi-'observable' only in the form of some time-units of silence (ϵ). For these reasons, the formal models of the regular node and the impostor node *differ mainly in the transition probabilities* of the ϵ transitions and only *little* in the transition probabilities of the 'visible' communication events which the impostor 'wants' to imitate as best as possible. Structurally, the transition table of the impostor node is the same as the transition table of the normal node; due to shortage of page space in this paper we do not print this similar impostor table here.

This smallness of the differences between the automata models of the regular node and the impostor node (which vary numerically in the various series of our experiments: see below) will considerably challenge our following *decider algorithm*.

2.3 The Decider Algorithm

Because our decider algorithm makes a *statistical* comparison between an actually received *input* word and a (pseudo)-randomly generated *reference* word, which is created from an output stream that represents the series of a specific node's most recent actions, the (pseudo)-random number generator called by the decider algorithm must be of high 'randomness' quality; otherwise the statistical comparison between the input word and the reference word would yield unreliable decisions on the basis of spurious correlations resulting from 'instrumental artefacts'. After several preliminary tests, the Julia implementation of Xoroshiro128+, version 1.5.3,[3] was found good enough and was thus chosen for the subsequent experiments.

Algorithm A

```
PROC DetectImpostor(A: word_received)
     { n := length(A)
       B := randomGenerateInternalWord(n)
       positiveSignals := 0
       FOR(k = 1...n)
          { subStrA := firstSymbols(A,k)
            subStrB := firstSymbols(B,k)
            tabA := generateFreqTable(subStrA)
            tabB := generateFreqTable(subStrB)
            signif := calcChiSquare(tabA,tabB)
            IF(signif > criticalVal)
              { increment positiveSignals } }
       IF(positiveSignals > n/2)
         { return true } // oracle: "impostor"
         ELSE
```

[2]In the end, however, our actually chosen p_i values are not even relevant for the purpose of demonstrating the capability of our technique: it would be easy to re-run all of our experiments (see below) with differently chosen p_i values just as well.

[3]https://github.com/JuliaRandom/RandomNumbers.jl [7]

```
        { return false } // oracle: "friend" }
```

For the sake of rapid decision-making the design of our statistical decider algorithm **A** (see above) is quite simplistic, although there is much reason to assume that more sophisticated (but, hence, also slower) algorithms would lead to more accurate decisions; (see below: 'Related Work' and 'Future Work'). Simply stated, our algorithm is *intended* to:

- output 'friend' *if* there *is* a 'statistically significant' correlation between a random input word that was externally produced from the normal node automaton and a random reference word that was internally produced (by the algorithm itself with help of Julia) from the normal node automaton, too.
  - The oracle is fallible (error type α: false positive)
- output 'crook' *if* there is *no* 'statistically significant' correlation between a random input word that was externally produced from the impostor node automaton and a random reference word that was internally produced (by the algorithm itself with help of Julia) from the normal node automaton.
  - The oracle is fallible (error type β: false negative)

In both oracles, 'statistical significance' is determined by the method of 'χ^2'.

For the sake of speed the algorithm does not consider the (exponentially large) set of all possible sub-strings, but only a 'linear' sub-set thereof. Since the χ^2 test is called repeatedly in the body of the `FOR`-loop, the algorithm even in this simple form is already rather slow.

2.4 Design Considerations for the Experiments

Our randomised experiments are designed along the lines of the following considerations and parameters:

- We do not only want to detect an impostor node, but also 'confirm' a friend node as such. For this reasons we need
  - several experimental series in which random 'words' from the impostor automaton are compared against random 'words' from the normal automaton;
  - several experimental series in which random 'words' from the normal automaton are compared against other random 'words' from the normal automaton, too.
- Due to the *law of the large numbers* we may reasonably hypothesize that the identification *rate* in any of the above-mentioned scenarios gets better when the 'words' that are compared against each other get longer. For this reason we need
  - several experimental series in which, for *fixed* probability values, the 'words' compared against each other are rather short;
  - several experimental series in which, for *fixed* probability values, the 'words' compared against each other are rather long;
- For the detection of the impostor node (with its slightly different finite automaton) it is also 'reasonable' to hypothesise that its detection gets easier when the probability differences (compared against the transition probabilities in the normal automaton) get larger. For this reason we also need

– several experimental series in which, for *fixed* word lengths, the probability differences between the two automata are rather ‘small’;
– several experimental series in which, for *fixed* word lengths, the probability differences between the two automata are rather ‘large’.

For all the above-mentioned combinations we conducted numerous experiments the results of which are shown in the subsequent section.

To be able to measure our algorithm’s detection accuracy in percent (%), each experimental ‘run’ consisted of 100 ‘repetitions’ (each with its own different random input) in which the numbers of α-errors (false-positive) and the numbers of β-errors (false-negative) were recorded. The final *aim* of these experiments —in accordance with the title of this paper— was the *discovery of a positive correlation between memory requirements* (proportionally represented by the lengths of the ‘words’) and *accuracy of detection*, whereby it is (again) reasonable to hypothesise that the highly accurate identification of the impostor or the friend will require considerably amounts of memory.

3 Experimental Results

Because of the page space limitation in this paper, we cannot show the results of all experiments which we have carried out. In this section we can only show a ‘representative selection’, whereby our other results (which are not shown in this paper) are all quite similar to the ones which are shown.

3.1 Identification of Friend Nodes

The following table (**3**) shows that the friend node can be identified as a ‘friend’ with high reliability by our simple algorithm only after approximately 1000 symbols have been received and processed (whereby for each word length ℓ we had 100 experimental ‘runs’ with different words):

Table 3

ℓ	Friend **identified**	β-Error
9	**17**/100	83/100
99	**73**/100	27/100
199	**81**/100	19/100
299	**83**/100	17/100
399	**87**/100	13/100
499	**89**/100	11/100
599	**93**/100	7/100
699	**94**/100	6/100
799	**92**/100	8/100
899	**96**/100	4/100
999	**92**/100	8/100

The table also shows that, with the method implemented by our simple algorithm, the improvement in the accuracy of the identification grows considerably *slower* than ℓ — i.e.: for a ‘slightly’ more reliable identification of a friend node, the observation length (and, hence, the memory requirements) must ‘strongly’ increase.

Table 4

ℓ	Impostor **detected**	α-Error
9	**14**/100	86/100
99	**69**/100	31/100
199	**83**/100	17/100
299	**83**/100	17/100
399	**83**/100	17/100
499	**88**/100	12/100
599	**87**/100	13/100
699	**87**/100	13/100
799	**86**/100	14/100
899	**86**/100	14/100
999	**87**/100	13/100

3.2 Detection of the Impostor Node

3.2.1 Probability Difference 'large' and fixed, Word Length growing

Similar to the previous table, also table (**4**) shows a rather slowly growing improvement of the detection rate for longer and longer observations.

3.2.2 Word Length 'large' and fixed, Probability Difference growing

In the final table of this paper, $\ell = 1000$ for all experimental 'runs'. The differences D, which appear as natural numbers in the following table (**5**), simply refer to the number of *modifications* which were made in the transition table of the regular WSN node's finite automaton (see above) in order to obtain the finite automaton of a similar impostor node. As mentioned above, these differences affect only the transition probabilities of the two automata — not their alphabets nor their graphical structures.

Table 5

D	Impostor **detected**	α-Error
1	**73**/100	27/100
3	**89**/100	11/100
5	**89**/100	11/100
7	**92**/100	8/100
9	**96**/100	4/100
11	**98**/100	2/100
13	**98**/100	2/100
15	**97**/100	3/100
17	**98**/100	2/100
19	**99**/100	1/100

As expected, any 'clumsy' impostor, who cannot imitate the 'statistical average behaviour' of a regular WSN node with a high level of similarity, can be identified by the purely statistical methods of our simple detection algorithm with high reliability.

However, our detection rates did not reach acceptable levels of reliability for $\ell \leq 1000$. According to our experiments, our simplistic algorithm can yield acceptably reliable results only for observations of $\ell > 1000$, which corresponds to a considerable volume of internal memory for any WSN node on which such detection algorithms are supposed to be implemented.

4 Related Work

Surveys such as [2][3][6] about wireless sensor networks *in general* can be found in large quantities. The literature about particular case-specific WSN designs, deployments and applications is already so vast that it cannot be concisely summarised any more. Recent *application*-oriented surveys can be found in [1][21][28]. All in all, the formal modelling of WSN nodes and communication algorithms by means of stochastic automata is well known and widely accepted [23][31]: for comparison see also [17]. More specific surveys about WSN *protection and security* (which is the theme of our paper) are also known [8][11][12][15][16][33].

Somewhat related to our work is [26] in which the nodes of a WSN must make decisions about whether a data measurement result communicated by some node to its adjacent nodes is reliable or perhaps a negligible 'outlier' due to some technical 'glitch'. Such a decision problem would occur, for example, when *most* nodes in a region would measure a spring time temperature of $+22$ Celsius whilst *one* node reports to have measured a winter temperature of -17 Celsius. Whereas in [26] such a node is simply presumed to have 'erred', in the context of our work such a node might possibly be a hostile 'crook' node attempting to confuse the WSN deliberately.[4] Also in [26] some type of finite automaton (NFA) is used to model a WSN node formally.

In [13] the presence of a hostile impostor node in a WSN is explicitly presumed. In that paper it was also noted that the detection of such a node by the other nodes is *"memory demanding"* [13]. Also in that paper (such as in ours) we found a random-number-based impostor identification algorithm [13](Fig. 1) together with some experimentally obtained detection likelihood statistics [13](Fig. 7); however little was stated about the opposite dilemma of falsely suspecting a friend node to be an impostor.

Also [32] was concerned specifically with the detection of node replication attacks, for the purpose of which several highly sophisticated (i.e.: quite complicated —even including cryptography) distributed algorithms (protocols) were defined and discussed. With such distributed methods many WSN nodes are 'helping together' to identify an impostor, whereas with our technique any WSN node can make such a detection attempt individually. According to [32] the detection probability was on average very high (often up to 100%), whereas the memory consumption of the detection protocol remained somewhat problematic, and little was stated about the run-time efficiency (i.e.: time needed from the start of a distributed protocol to the decision about the detection of an impostor) of those methods. By contrast, we have attempted to tackle the given problem quite simplistically, by purely statistical means and without any sophisticated cryptographic techniques, albeit with somewhat lower detection rates.

Moreover, our paper was also inspired by [22] in which, too, for some given probabilistic finite automaton (PFA), a detection problem was 'asymptotically' solved *"with increasing certainty as more information is acquired from observing the behaviour of the given PFA"*. However, the detection problem in that paper differs slightly from our detection problem in the sense that the observer in [22] tries to guess in which particular internal state the observed PFA currently is (which requires 'white box' knowledge of its entire internal state space), whereas our work aims merely at guessing whether the observed PFA as a whole belongs to a specific class ('friend' or

[4]As the algorithm presented in this paper cannot discern between malicious behavior and node's behavior which has simply 'erred', α-errors were taken into account during experimental runs.

'impostor') whereby the observed PFA's internal state space remains hidden in a 'black box'.[5] In [22] the problem was solved by way of reduction to the comparability problem for convergent Markoff chains.

Last but not least it is worth noting that, from a *software engineering* perspective, the given problem (in its formal PFA or SFA representation) is also amenable to the well-known software engineering techniques of *mutation testing* [19]. The problem in this context is the ingenious 'discovery' or creative 'invention' of suitable mutation operators [4] by the software engineer in such a manner that the resulting test sequences can reliably distinguish an automaton A from a mutant A'. However, after some suitable mutation operators have been found and defined, the distinction (or equivalence) decision about A and A' can be reached in *polynomial time* [19].

5 Conclusion

In this paper we have demonstrated that it is at least in principle possible to detect impostor nodes in wireless sensor networks with quite simplistic detection algorithms by purely statistical means and from mere observation without any knowledge of the impostor's internal composition.

It is important to note that throughout this paper we have worked under the the *assumption* of a so-called *homogeneous* WSN in which *all* member nodes are *supposed to be of identical types*, whereby the 'type' of a node is defined by its abstract PFA (or SFA). By contrast, our method is *not* directly applicable in so-called *heterogeneous* WSN which are deliberately and purposefully composed of nodes of various different types: in such cases, *multiple* detector algorithms would be needed for the multiple types of nodes that can be found in a heterogeneous WSN.

Anyway —and in accordance with other already published literature— our detection rates did not reach acceptable levels of reliability for observations of length $\ell < 1000$. This result implies that considerable volumes of internal memory are needed for any WSN nodes on which such detection algorithms are supposed to be implemented.

Last but not least we may remark that the 'implantation' of our algorithm into a 'normal' node n changes the observable run-time-behaviour of n itself as soon as the detector algorithm gets invoked. For this reason it might seem recommendable to deploy our detection algorithm only in special 'passive' (or 'always-silent') listener-nodes which do not normally communicate with the 'normal' other nodes of the WSN into which they are placed.

5.1 Outlook to Possible Future Work

As both our formal models and our detection algorithm had been designed quite simplistically for the exploratory purposes of this paper, more precise results might be obtained in 'future work' by refining the formal models with more details as well as by augmenting the detection algorithm with more sophisticated techniques of reasoning (Artificial Intelligence) — albeit at the price of a longer time-to-detection. Moreover, as mentioned above, further adaptations and improvements of our technique would also be needed for heterogeneous WSN of multiple types.

5.2 Acknowledgments

We first became aware of this paper's topic of node replication attacks in July 2017 during a seminar at LaBRI, Université Bordeaux 1, hosted by our colleague *Mohamed Mosbah* and his

[5]In fact the semantic identity of two regular languages, $\mathcal{L}(A') = \mathcal{L}(A'')$, does *not* require any structural (syntactic) identity (not even isomorphy) between the two finite automata, A' and A'', by which those languages are generated.

research group. Many thanks to the anonymous reviewers for their valuable remarks about the initially submitted draft.

References

[1] Ian F. Akyildiz, Tommaso Melodia, and Kaushik R. Chowdhury. A Survey on Wireless Multimedia Sensor Networks. Computer Networks 51/4, 921–960, 2007.

[2] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A Survey on Sensor Networks. IEEE Communications Magazine 40/8, 102–114, 2002.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: a Survey. Computer Networks 38/4, 293–422, 2002.

[4] Paul Ammann and Jeff Offutt, Introduction to Software Testing. Cambridge University Press, 2008.

[5] James Anderson. Automata Theory with Modern Applications, Cambridge University Press, 2006.

[6] Paolo Baronti, Prashant Pillai, Vince Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless Sensor Networks: a Survey on the State of the Art, and the 802.15.4 and ZigBee Standards. Computer Communications 30/7, 1655–1695, 2007.

[7] Jeff Bezanson, Alan Edelmann, Stefan Karpinski, and Viral B. Shah. Julia: a Fresh Approach to Numerical Computing. SIAM Review 59/1, 65–98, 2017.

[8] Tapolina Bhattasali and Rituparna Chaki. A Survey of Recent Intrusion Detection Systems for Wireless Sensor Networks. Proceedings International Conference on Network Security and Applications, 268–280, 2011.

[9] Patrick Billingsley, Statistical Methods in Markov Chains. The Annals of Mathematical Statistics 32/1, 12–40, 1961.

[10] Sorana D. Bolboacá, Lorentz Jäntschi, Adriana F. Sestras, Radu E. Sestra, and Doru C. Pamfil. Pearson-Fisher Chi-Square Statistic revisited. Information 2/3, 528–545, 2011.

[11] Ismail Butun, Salvatore D. Morgera, and Ravi Shankar. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. IEEE Communications Surveys and Tutorials 16/1, 266–282, 2013.

[12] Okan Can and Ozgur Koray Sahingoz. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. Proceedings 6th International Conference on Modeling, Simulation, and Applied Optimization, 1–6, IEEE 2015.

[13] Mauro Conti, Roberto di Pietro, Luigi V. Mancini, and Alessandro Mei. A Randomized, Efficient, and Distributed Protocol for the Detection of Node Replication Attacks in Wireless Sensor Networks. Proceedings 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 80–89, 2007.

[14] Randal Douc, Eric Moulines, Pierre Priouret, and Philippe Soulier. Markov Chains. Springer, 2018.

[15] Ashfaq Hussain Farooqi and Farrukh Aslam Khan. A Survey of Intrusion Detection Systems for Wireless Sensor Networks. International Journal of Ad Hoc and Ubiquituous Computing 9/2, 69–83, 2012.

[16] Ashfaq Hussain Farooqi and Farrukh Aslam Khan. Intrusion Detection Systems for Wireless Sensor Networks: a Survey. Proceedings International Conference on Future Generation Communication and Networking, 234–241, 2009.

[17] Stefan Gruner, Towards a Generic Design for General-Purpose Sensor Network Nodes: Position Paper. Proceedings 5th International Conference on Evaluation of Novel Approaches to Software Engineering, 259–264, 2010.

[18] Hassan Hatefi and Holger Hermanns. Model Checking Algorithms for Markov Automata. Electronic Communications of the EASST 53 (Automated Verification of Critical Systems), paper #8, 2012.

[19] Robert M. Hierons and Mercedes G. Merayo. Mutation Testing from Probabilistic and Stochastic Finite State Machines. Journal of Systems and Software 82/11, 1804–1818, 2009.

[20] Colin de la Higuera and Jose Oncina. Learning Stochastic Finite Automata, LNAI 3264, 175–186, 2004.

[21] Dionisis Kandris, Christos Nakas, Dimitrios Vomvas, and Grigorios Koulouras. Applications of Wireless Sensor Networks: an Up-to-date Survey. Applied System Innovation 3/1, paper #14, 2020.

[22] Christoforos Keroglou and Christoforos N. Hadjicostis, Verification of Detectability in Probabilistic Finite Automata. Automatica 86, 192–198, 2017.

[23] Sajjad A. Madani, Jawad Kazmi, and Stefan Mahlknecht. Wireless Sensor Networks: Modeling and Simulation, ch. 1 in Aitor Goti (ed.), Discrete Event Simulations. InTech, 2010.

[24] Rudolf Mathar, Informationstheorie: Diskrete Modelle und Verfahren. Teubner, 1996.

[25] Rudolf Mathar and Dietmar Pfeifer, Stochastik für Informatiker. Teubner, 1990.

[26] Dylan McDonald, Stewart Sanchez, Sanjay Madria, and Fikret Ercal. A Communication-Efficient Framework for Finding Outliers in Wireless Sensor Networks: Extended Abstract. Proceedings 11th International Conference on Mobile Data Management, 301–302, 2010.

[27] Michael O. Rabin, Probabilistic Automata. Information and Control 6/3, 230–245, 1963.

[28] S.R. Jino Ramson and D.J. Moni. Applications of Wireless Sensor Networks: a Survey. Proceedings International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology, 325–329, IEEE 2017.

[29] Animesh R. Tayal, N.V. Choudhary, and Madhuri A. Tayal. Simulation of Sensor Nodes for Energy Conception in Wireless Sensor Networks using Finite Automata. Proceedings International Conference on Advances in Computing, Communication and Control, 685–688, 2009.

[30] Ikjune Yoon, Dong Kun Nooh, Dongeun Lee, Rony Teguh, Toshihisa Honma, and Heonshik Shin. Reliable Wildfire Monitoring with Sparsely deployed Wireless Sensor Networks. Proceedings 26th International Conference on Advanced Information Networking and Applications, 460–466, 2012.

[31] Fengling Zhang, Lei Bu, Linzhang Wang, Jianhua Zhao, Xin Chen, Tian Zhang, and Xuangdong Li. Modeling and Evaluation of Wireless Sensor Network Protocols by Stochastic Timed Automata. ENTCS 296, 261–277, 2013.

[32] Ming Zhang, Vishal Khanapure, Shigang Chen, and Xuelian Xiao. Memory-Efficient Protocols for detecting Node Replication Attacks in Wireless Sensor Networks. Proceedings 17th IEEE International Conference on Network Protocols, 284–293, 2009.

[33] Yun Zhou, Yuguang Fang, and Yanchao Zhang. Securing Wireless Sensor Networks: a Survey. IEEE Communications Surveys and Tutorials 10/3, 6–28, 2008.