Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2011, Article ID 575171, 13 pages
doi:10.1155/2011/575171

Research Article

"Who Counterfeited My Viagra?" Probabilistic Item Removal Detection via RFID Tag Cooperation

Mauro Conti,[1] Roberto Di Pietro,[2, 3] and Angelo Spognardi[4]

[1] *Computer Science Department, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands*
[2] *Mathematics Department, Università di Roma Tre, 00146 Roma, Italy*
[3] *Computer Engineering and Mathematics Department, UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, 43700 Tarragona, Spain*
[4] *Computer Science Department, Università "La Sapienza" Roma, 00198 Roma, Italy*

Correspondence should be addressed to Roberto Di Pietro, dipietro@di.uniroma1.it

Received 16 June 2010; Revised 13 September 2010; Accepted 21 November 2010

Academic Editor: Damien Sauveron

We leverage RFID tag cooperation to enforce tampering detection. That is, we provide a set of probabilistic protocols that detect the absence of a tag from a system composed of a set of tags and a reader. Our proposals are able to detect which tag and for how long it has been taken away from the system. The grain of the detection can be tuned with respect to the resources available on the tags. Another merit of our solutions is to provide a proof-of-concept that a small level of cooperation among tags can further extend the range of applications RFID can support, possibly opening new veins of research. The proposed protocols fit the resource constraints of the several classes of RFID available on the market. In particular, the memory requirement ranges from few memory slots to a number of memory slots that is proportional to the number of rounds the presence of a tag is going to be checked. Computation is just one hash per round. This fully fledged set of protocols is thought to trade off the detection grain with the resources on the tag: the finer the item removal detection grain, the more resources a protocol requires. A thorough analysis for the removal detection probability is provided. Finally, extensive simulations support the analytical results, showing the viability of the proposed solutions.

1. Introduction

The possibility to embed inexpensive wireless devices within essentially any object, keeps RFID systems attracting interest from both Academia and Industry. Applications of this technology are increasing, and its evolution is pushed ahead by the intent to substitute bar codes, but also to provide exclusive and practical services, such as supply chain automation, transportation payments, access control, electronic credit cards, product tracing, animal identification, library, and health care to name a few [1].

One of the most attractive features of RFID systems—together with their low cost—is the miniaturization of tags. Tags are very small devices able to communicate via Radio Frequency (RF) with a reader: tags usually do not carry any battery, but can harvest energy from the electric field generated by readers. Once a tag receives RF waves, it converts them into Direct Current (DC) power, so that it can operate its internal circuitry. This way, a tag is able to reply to the reader transmitting its information via modulated backscattering [2], that is basically modulating the incident RF signal by passively switching the reflection characteristics. The reply provided by the tag is usually relayed via the reader to a database server and analyzed to identify the tag. The literature about RFID systems offers many proposals and protocols to enhance the security and to protect privacy of RFID users [3, 4]. However, much less effort has been made to explore the opportunities offered by this technology when tags are allowed to cooperate. The aim of this work is to illustrate the feasibility of a simple but effective form of collaboration among RFID tags mediated by the reader. The collaboration is leveraged to propose a set of probabilistic protocols aimed at detecting the absence of the items tags are attached to, under a specific (yet quite common) deployment scenario. We consider that the absence of a tag may be the first step to item tampering (as discussed in Section 3).

Throughout this paper, we will assume the supply chain management scenario synthesized in Figure 1. However, this scenario is simply an example of a possible application of our protocols. In particular, we are interested in providing a cheap and effective way to check for the integrity of items after their shipment. We assume that: each item has an RFID tag attached to it and during the packaging the items are grouped together (e.g., into a container). It could be the case that during the delivery process some of these items are subtracted, tampered with or replaced, then put back in the container. The sender could be interested in having some sort of assurance to detect the occurrence of the above threat. A real example of this scenario can be a consequence of the rules introduced in 2006 by the American FDA: certain drugs must be traced with RFID during their supply chain. Our solutions provide a probabilistic assurance that the items have not been removed from the initial configuration—this event being detected even if items are put back later on. Note that should an item be simply removed (and not put back), our solution would detect such an event as well.

Contribution. In this paper, we propose a proof-of-concept solution, accompanied by thorough analysis and supporting simulations, that leverages cooperation among tags of an RFID system. In particular, our proof of concept addresses a relevant practical problem: enforcing the integrity of the configuration of a set of RFID tags (i.e., of the objects they are attached to). We provide a set of probabilistic protocols that, leveraging the reader as a relay and synchronization point, implement a low degree of cooperation among tags achieving the above goal. The different protocols trade off the grain of the tampering detection (as clarified in Section 3) with an increase in the memory slot to be on board the tag. The proposed protocols are fully viable with current RFID tags: memory requirements range from a single memory slot of few hundreds bits to several memory slots of such size; computation is mainly a hash function per round, and, communications amount to at most two messages per round. Thorough analysis shows the effectiveness of our solutions. Simulation results support the analytical findings.

Organization. Next section reports on related work. In Section 3, we introduce the assumptions used in this work. An overview of our proposed solutions is given in Section 4, while the detailed description and the analysis of the proposed protocols can be found in Section 5. In Section 6 we report and discuss the simulation results that support the analytical results. Finally, Section 7 presents some concluding remarks.

2. Related Work

Security and privacy of RFID systems are challenging research issues [4]. The research community has produced a relevant corpus of work on these challenges and many protocols have been proposed. One of the most known protocols for RFID privacy and identification was proposed by Molnar and Wagner [5]. Several solutions have thereafter been proposed to address identification, authentication, and tag privacy [6–9]. In particular, the different proposals either guarantee different protocol features (e.g., forward secrecy and scalability) or fix security problems discovered in previous protocol versions. Many privacy preserving protocols make use of the idea to keep tags and server synchronized over a hash chain: namely, they share a one-time value that is updated at each authentication, using a one-way function [10–18].

In particular, relevant research efforts have addressed identification, authentication, and tag privacy: as noticed in [19], from 2002 to 2009 more than 200 papers have been published, analyzing security and privacy of RFID systems. Also, around 150 proposal have been introduced, studying the information flow between the tags and the reader. For updated references on RFID security and privacy, the RFID Security and Privacy Lounge maintained by Gildas Avoine is an established reference point for the scientific community [3].

Our work leverages cooperation among tags and proposes four distributed protocols in order to enforce the integrity of a set of objects equipped with RFID tags.

Previous work in this direction has been inspired by the problem of generating “yoking-proof”, that is, proofs that two [20] (or more [21, 22]) tags have been scanned at the same time. However, previous solutions required reader interactions to define a “chain” of readings of the tags, and querying them accordingly to their order in that chain. Hence, the integrity check depends on the reader performing queries in a specific order. This implies the reader to know in advance the tags to be queried. We observe that this could rise an information leakage, in case the reading is outsourced. Compared to yoking proof [20–22], our solution can provide a more fine-grained detection, since we are able to find which tag and for how long it has been taken away from the system. Furthermore, unlike previous works, our protocols are completely distributed among the RFID tags, that is, the proof that the tags have been together—what we refer as the integrity of the tags set—is proved offline, by querying the tags themselves. Spreading of sensible information all over the tag memory is an approach used also in [23]. In that work, the authors

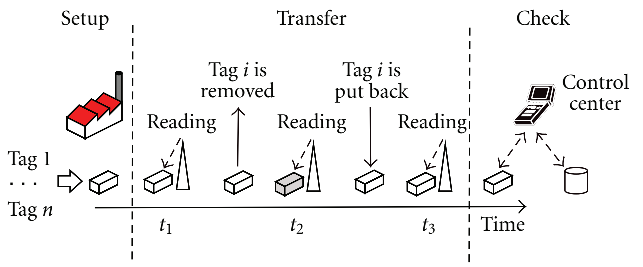


FIGURE 1: Example of application: supply chain management scenario.

propose to realize a key deployment using the tags to carry the shares of a (n, n)-secret sharing scheme [24]. Only who is able to scan all the n tags will be able to recover the secret key.

Previous studies [25] already highlighted that a track and trace system such as [26] might not be sufficient to face the problem of goods counterfeit. In fact, the solution in [26] bases the check on the fact that a given product (with a given ID) cannot be sold in two different places (e.g., UK and Nigeria). However, if the counterfeit is done by the final retailer, or the intermediate carriers do not update the EPC network [27] database, the check will fail. In [25], the authors propose to extend the EPC network to include an authentication mechanism, making it unfeasible to just copy the tag's ID. In our solution, we also use a tag authentication mechanism: each tag has a secret value shared only with the control center—but not with other tags, neither with the reader, that in our case can be untrusted. Furthermore, our solution is not dependent on any standard like the EPC one [25]. That is, our system would work also independently from the EPC network. In the same direction of proposing a protocol against tag counterfeiting, the work in [28] introduces a smart and effective clone detection mechanism for RFID. It is based on verifying the correctness of sequences of two time-consecutive events in the tag history within the supply chain. The proposed system focuses on clone detection and assumes a chain of trusted partners that actively and honestly cooperate to protect the system.

There are currently different standards for RFID technology from EPC [27] and ISO [29]. EPC standards are, (i) Class 0 (Ultra High Frequency-UHF); (ii) Class 1 (High Frequency-HF 13,56 Mhz/UHF); Class 1 Gen 2 (UHF). There are also ISO standards for generic parameters (ISO 18000-1) and for different air interface parameters (standards ranging from 18000-1 to 18000-7 for air interface, varying from 135 kHz to 5.8 GHz). Furthermore, there are ISO standards for proximity cards (ISO 14443), for vicinity cards (ISO 15693), and close-coupled cards (ISO 10536). An example of RFID tag implementing the standard ISO 15693 is the NXP SLI-S ICS54 with 2048 bits of memory [30] and a maximum read distance of 1–1.5 meters. This multiplicity of resources RFID tags are equipped with motivates our effort to provide a range of protocols; choosing the appropriate one requires to find out the best trade off between the available resources on tag and the desired detection grain—that is, the amount of information detailing the system violation.

As for the computation effort required, the major computational task assigned to tags is the computation of a hash function. Note that the feasibility of implementing a hash function on RFID tag has been proved in [31–33], to cite a few.

3. System Model and Assumptions

In this section, we describe the system model and the assumptions used in this work, together with the adversary ($\mathcal{ADV}$) capabilities.

3.1. System Model. In our problem setting, we have a system composed of n items and an untrusted reader. We assume that each of these items has a tag stuck to it. The tags are used to guarantee the integrity of the traced items, namely to check if an item has been removed from (and, possibly, reinserted into) the system. From now on, we refer to "tampering detection" assuming that if an item has been removed from the others, it has been tampered with.

The reader splits time in intervals (also called *rounds*) of equal length and queries the tags once at each interval (i.e., tags receive energy from the reader). The reader does not have any kind of cryptographic material but acts only as an "energy provider" and a relay node for the tags to cooperate. We do not require the reader to be networked. Hence, (1) we can use the system offline (for instance, during shipping, getting rid of expensive satellite or cellular link-based-check); (2) we do not need to provide the entity charged to transfer the items with any kind of information related to the tags; (3) we do not need the reader pass through any special setup or authentication phase with any other entity.

As we are designing our solution for checking the integrity of a set of items (to which tags are attached), we require each tag to be not easily detachable from the item. Should this tampering occur, we assume it is detectable—for example, via physical observation of the item. This assumption is fundamental, since an $\mathcal{ADV}$ could detach the tag from its companion item and leave only the tag in the system bringing away the item. After the corruption of the object, $\mathcal{ADV}$ could bring the item back and finally repositioning the tag on the item: the tampering would not be detected by our protocol. Note that the above requirement can be currently satisfied: solutions do exist able to detect the removal of a tag from the corresponding item [34].

The reading of multiple tags takes a period of times related to the number of tags that provide a reply. This is due to the anticollision protocols the readings are based on. In this work, we assume that if a tag was read during a reading, then it was present during the whole duration of the reading period. As assumed in other works [35, 36], if a tag is present, it will always terminate the protocol within a certain interval of time.

The life cycle of the envisaged system is composed of three main phases (Figure 1).

(i) Setup. This phase takes place just once, in a protected environment, where the configuration of tags and reader can be considered safe. In this phase, the tags receive their initial setup and are configured with the selected protocol. Tags are also stuck to the items they should protect the integrity of. In the following, we will use the term tag to refer to the combination of the item plus the tag—this assumption is based on the fact that the two cannot be separated, otherwise this event will be detected. The reader is also programmed to read (i.e., to provide energy to) the tags with the appropriate interval duration. As it will be clear in the following, the reading frequency has a direct relationship with both the resources required on tags and the compromising detection probability. We face the impact of a bad/faulty interval length configuration in Section 3.2.

(ii) Transfer. The transfer phase represents the shipping of the tags: during the transfer, tags are supposed not to leave the area the reader provides energy to. We assume $\mathcal{ADV}$ can compromise the tag only during this phase (see Section 3.2). We assume that some sort of further, loose physical surveillance is in place, that corresponds to the area lightened by the reader. This requirement is needed not to prevent $\mathcal{ADV}$ from removing some items, but to prevent $\mathcal{ADV}$ from compromising items in situ. That is, we assume that $\mathcal{ADV}$ can compromise some or even several items but, to escape the loose physical surveillance, it has to bring them outside the supervised area—where it could later return to redeposit the compromised object. During the shipping, at each interval the reader provides energy and relays tag answers. In particular, we say that the integrity of the set of tags is violated if $\mathcal{ADV}$ subtracts a tag from the system for a given consecutive number of queries issued by the reader, and this event is not detected by the integrity protocol. Note that the case where an item is subtracted and not put back in the system is just a special case of the compromise where $\mathcal{ADV}$ puts the item back.

(iii) Check. The last phase consists in checking the integrity of the tags: tags' memory slots devoted to detect compromising are downloaded in a trusted environment (the control center) and their content processed. The aim of this phase is to detect any violation of the integrity of the system that could have happened during the transfer phase.

3.2. Adversary Model. Our aim is to preserve the tags integrity against an $\mathcal{ADV}$ detailed in the following, that can act during the Transfer phase. $\mathcal{ADV}$ aims to keep a tag away from the others in a stealthy way—$\mathcal{ADV}$ does not want its rogue activities to be detected during the check phase. $\mathcal{ADV}$ is able to eavesdrop all the communication within the system.

Considering that $\mathcal{ADV}$ is able to compromise the reader and then also to alter the duration of the intervals, the following dreadful attack could take place. $\mathcal{ADV}$ could stop the reader's activity, let us say at round $(d - j)$, and compromise tag at will—the compromising taking j rounds. Later on, when compromising activities are accomplished, it could reactivate the reader and run the intended protocol at a faster pace, so as to recover the j rounds lost. Once these j rounds are recovered (assuming it takes ℓ slow-paced rounds), the reader will be then set to the normal operating cycle, so that from round $d + \ell$, the tampering will not be detectable. Note that this attack is successful since tags do not have their own clock, relying on the reader for synchronization purposes.

To cope with this threat we envisage both software and hardware solutions, both focused at posing an upper bound at the speed tags can reply. As for the hardware solutions, one could assume to have tags equipped with a condenser that does not allow the tags to reply before a given quantum of time. As for software solutions, we can assume that tags can introduce a delay via software operations—busy waiting—, such as executing a cycle composed of skip-like operations.

Table 1: Notations.

Notation	Description
$\mathcal{ADV}$	Adversary
t	Current round—issued by the reader
id_a	ID of tag a
id_a^t	ID of tag a, at a given round t
k_a	Shared random seed between tag a and control center
n	Number of tags in the system
R	Number of readings of the Transfer phase
p	Probability for a tag to send its message
q	Probability for a tag to store a message sent by another tag
s	Actual number of tags sending a message ($p = s/n$)
r	Actual number of messages considered by a tag ($q = r/n$)
$f(\cdot)$	Pseudorandom function
$H(\cdot)$	Hash function

We observe that it is out of the scope of this paper to design a protocol that guarantees the authentication of the queries issued by the reader or that preserves the privacy of the tags. Also the authentication of tags is not required, since any use of a bogus tag would be detected by the control center. In fact, any tag outside the system would be unable to comply with the protocol, since it would not have received a valid setup (as it will be clear in Section 4.2). However, interested readers could refer to [18] for a solution to the authentication problem. Similarly, we do not consider any kind of DoS attack, as they will make the whole system ineffective and would be eventually detected by the control center. Finally, remember that the adversary does not tamper with the RFID tags. As assumed in similar works [20–22], a simple tamper-evidence and the effort required to extract the secret information are sufficient deterrent to such attack in many situations. Moreover, even if we relax this assumption, to perform such an attack the adversary might need to remove the aimed tag from the system (e.g., to bring it in a lab), and such removal would be detected by our solutions.

3.3. Notation. In Table 1 we summarize the notations used in this paper. The notations for p, q, s, and r are related to a single reading issued by the reader.

4. TIP Protocols Overview

In this section, we introduce our Tag Integrity Preserving protocols (TIP). Note that the detection provided by all these protocols is probabilistic. That is, an attack is detected with a given probability only, that depends on system parameters. However, note that these parameters are tunable by the owner of the system, so that it can decide the level of assurance required.

All the protocols presented in this paper are based on the main idea that at each round: a subset of tags can broadcast their presence (via the reader, that can be seen acting as both an energy provider and a transponder), and, that a subset of the tags in the system can keep trace of the value relayed by the reader. How to select the set of tags that can broadcast their presence, the set of tags that have to record the broadcast value, and the amount of information stored on recipient tags are all parameters that characterize the proposed protocols. We underline that the presented protocols evolve in different ways with time, depending on the initialization seeds and the specific protocol operations. Knowing the random values every tag is initialized with, the control center is always able to check the integrity of the system. In fact, it can mimic and reconstruct the expected evolution of both the communications and the exchanged values.

In the following, we first discuss a naïve solution—that helps introducing the caveats of the application—, next we give a general presentation of our proposed protocols, while a detailed description is provided in Section 5.

4.1. Naïve Solution. A simple and effective way to keep trace of the tags that are present in the system is to continuously read all the tags and to make all tags logging all the replies. In this way, every time $\mathcal{ADV}$ gets a tag apart from the system, the violation is detected and traced by all the tags. Even if effective, this solution has two main drawbacks, (i) it generates too much traffic between the tags and the reader, and, (ii) it requires for any tag a number of memory slots that is equal to the overall number of rounds the system is supposed to last. To make the solution practical, we should reduce the amount of tags that are allowed to reply to a reader query, as well as the information to be stored on tags at any round.

The first problem can be mitigated introducing a probabilistic protocol: every tag replies to a reader query with probability p. Hence, reducing the amount of generated traffic by a factor p—on the average. This approach is the basic component of all our TIP protocols. As it will be clear in the following, introducing this probabilistic approach has a three-fold effect: it reduces the number of replies to a reader query; it reduces the amount of storage required on tags, and, it makes unpredictable to $\mathcal{ADV}$ which are the tags that are going to reply to a given reading issued by the reader. This last point is a major problem for the adversary. In fact, to predict if a tag will answer to a given reading, $\mathcal{ADV}$ has no other ways that physically compromise the tag to access its memory. More details about this aspects will be explored in Section 4.2.

A possible way to further reduce the storage requirement is the use of aggregation, namely to fuse all the listened replies in one digest string. This aggregated value can be delivered to the control center as an evidence of the presence of a tag. This technique will be adopted in the digest answer proposal. Another way to mitigate the storage requirement for a single tag is to resort again to a probabilistic approach. That is, a tag logs a reply from the reader with probability q, hence reducing the storage requirement by a factor q—on the average. This approach is used in both the dynamic counter and the logger protocols.

```
begin
  (1) The reader sends a query, that is essentially the current
        round $t$ (also referred to as time in the following);
  (2) Tag $id_a$ determines if it has to reply to the current
        reading. The tag either goes to step (3) (if it has to
        reply) or to step (4) (otherwise);
  (3) If the above test is passed, the tag computes a string
        and replies to the reader;
  (4) The reader first receives all the replies,
        and then broadcasts each of them;
  (5) A tag $id_b$ receives the strings logically sent by the other
        tags—but physically relayed by the reader;
  (6) Tag $id_b$ first decides whether to store and process the
        received values; later it updates its internal variables to
        trace other tags, and get ready for the next round;
end
```

Algorithm 1: General behaviour of proposed protocols.

4.2. Solutions Overview. While we propose different protocols, all of them rely on the same rationales. In particular, the common behavior of the proposed protocols can be summarized in the Algorithm 1.

In step (2) and for each reading, tag id_a has a small probability (p) to reply with its own ID. This probability depends (in a pseudorandom way) on the value t sent by the reader and a value related to the internal state of the tag, namely a random seed shared with the control center. On average, only a small fraction of all tags will reply for each reading, generating a fraction of the traffic of the naïve solution. More in detail, a tag will send its id_a if and only if $f(t, k_a) = \text{true}$, being $f(\cdot)$ a pseudorandom function. For instance, $f(\cdot)$ could be defined as:

$$f(t, k_a) = \text{true} \Longleftrightarrow H(t, k_a)\left(\bmod \left\lceil \frac{1}{p} \right\rceil\right) = 0, \quad (1)$$

where $H(\cdot)$ is a hash function. In this way, $f(\cdot)$ returns *true* only with probability p.

In step (6), tags decide to trace an answer independently of the others and in a way that is unpredictable to an adversary. This step is implemented in different ways, in the different proposed protocols. As it will be clearer in the next sections, even if $\mathcal{ADV}$ forges the answers of a given tag, it will be unable to foresee which answers the given tag should trace, without knowing its internal memory state.

Eventually (e.g., when the Transfer phase ends, with reference to the example of Section 1) the control center queries the tags in the system to check for integrity. The data received by the tags should allow the control center to determine if the system integrity has been preserved. The check result comes with a given probability of accuracy as well as guaranteeing different properties, depending on the particular protocol used—as described later.

Please, observe that steps (3), (4), and (6) are particularly dependent on the specific proposed protocol. In particular, we propose four TIP protocols.

(i) *Digest Answer (Section 5.1)*. In this protocol each tag will aggregate all the strings (from the reader) generated by the tags that passed the internal sending test.

(ii) *Static Counter (Section 5.2)*. In this protocol, each tag will trace only a fixed subset of other tags—the IDs of these tags being preloaded in the Set-up phase. If a tag has passed the internal sending test (check described in step (2) of Algorithm 1), it just sends out its ID in plain text (we remind that privacy and confidentiality are out of the scope of this paper). The tag ID does not change for the different reading. If a tag has to trace a listened ID (e.g., id_a), it just increments a counter dedicated to id_a every time string id_a is broadcast via the reader.

(iii) *Dynamic Counter (Section 5.3)*. In the dynamic counter protocol each node does not have a fixed set of other nodes to trace. Instead, for each reading, the set of nodes a tag must trace is pseudorandomly determined by the current reading parameters (e.g., the time t) and a secret random seed shared between the tag and the control center only. For the tags to be traced, the corresponding counter is incremented. The random seed is needed to avoid a simple cloning attack, as detailed in Section 5.3. Note that this protocol on the one hand increases the number of required counters (compared to the static counter protocol) but on the other hand increases the grain of detection, as it will be clear later on.

(iv) *Logger (Section 5.4)*. In this protocol, the tags to be traced are pseudorandomly selected as in the dynamic counter. However, each time a tag is traced, a new memory slot is allocated recording the current tag ID and the value t. This protocol requires more storage on the tags when compared to the previous protocols. However, the granularity of the tracing is finer.

We observe that, how further discussed in Section 6, the different proposed protocols have different costs and features; for example, the digest answer protocol is not resilient to tag reading failures (in fact, this would lead to a sort of de-synchronization between tag and reader), while the other protocols are.

5. TIP Protocols Description

In this section, we give the details of each of the proposed TIP protocols. Note that the integrity check will take place at the control center. In particular, the memory of each tag will be securely collected at the control center, and each of these memories will be checked against its expected state computed by the control center, mimicking the protocol evolution. Indeed, protocol evolution is completely deterministic, given the values loaded on tags at the Set-up phase: the use of pseudorandom functions in combination with known random seeds enables the control center to completely reconstruct all the pseudorandom values of the involved functions.

```
        Data:
        $\text{id}_a^t$ {id of tag $a$ at time $t$}
        $k_a$ {shared random seed of tag $a$}
(1) begin
(2)     $t \leftarrow$ ReceiveTimeFromReader;
(3)     if CheckAnsweringTime ($t$, $\text{id}_a^t$) then
(4)         send ($\text{id}_a^t$);
(5)     end
(6)     $Z = \text{id}_a^t$;
(7)     $S \leftarrow$ ReceiveStringsFromTags;
(8)     foreach id $\in S$ do
(9)         $Z = Z \oplus \text{id}$;
(10)    end
(11)    $\text{id}_a^{t+1} = H(k_a \| Z)$;
(12) end
```

Algorithm 2: Digest-answer.

5.1. Digest Answer. This first proposal relies on the fusion of the replies provided by tags. In particular, the digest answer protocol is detailed in Algorithm 2 and works as follows. As for all TIPs, the first step (i.e., step (1) mentioned in Section 4.2) is the reader query, that comes with the time value t. The tag a executing the protocol, receives the value t (Algorithm 2, line 2). Then, a checks if it has to reply to the current reading (line 3). In this case, it sends its id_a for the current time t (line 4). As for tracing other tags (step (6) mentioned in Section 4.2), tag a receives the strings sent out by the tags that passed the internal test on whether to send or not their ID—these string being relied by the reader—(Algorithm 2, line 7). Then, a makes the XOR of all the received strings (lines 8–10), together with its own current id_a (line 6). Finally, tag a prepares itself for the next round, that is it updates its own ID. This is done by hashing the composition of values k_a and Z (line 11).

Analysis. Let us denote by p the probability that a tag passes the check required to send its ID. Hence, if we denote by F the event that a tag contribution will not be XOR-ed with the value Z of Algorithm 2 for L consecutive time round, we have that:

$$\Pr[F] = (1 - p)^L. \tag{2}$$

Note that p has a straight influence on the number of tags that send their ID: the average number of such tags is $\mu = np$. Indeed, the higher p, the higher the frequency a tag is traced, the higher the possibility to detect a compromise. Figure 2 helps devising the relationship among these competing parameters. However, note that the higher p, the higher the number of tags that the reader is required to serve.

```
Data:
id_a {id of tag a}
k_a {shared random seed of tag a}
T_a {set of id tag to trace assigned to tag a}
(1) begin
(2)     t ← ReceiveTimeFromReader;
(3)     if CheckAnsweringTime (t, k_a) then
(4)         send (id_a);
(5)     end
(6)     S ← ReceiveStringsFromTags;
(7)     foreach id ∈ S do
(8)         if id ∈ T_a then
(9)             Increase-counter (id);
(10)        end
(11)    end
(12) end
```

Algorithm 3: Static counter.

```
Data:
id_a {id of tag a}
k_a {shared random seed of tag a}
(1) begin
(2)     t ← ReceiveTimeFromReader;
(3)     if CheckAnsweringTime (t, k_a) then
(4)         Send (id_a);
(5)     end
(6)     S ← ReceiveStringsFromTags;
(7)     foreach id ∈ S do
(8)         if CheckIfToTraceNow (id, k_a, t)
            then
(9)             Increase-counter (id);
(10)        end
(11)    end
(12) end
```

Algorithm 4: Dynamic counter.

5.2. Static Counter. In the static counter protocol each tag traces a fixed set of other tags—preloaded in the Set-up phase. The protocol is detailed in Algorithm 3. The tag a executing the protocol listens to the time t (line 2); checks if it has to participate in the current reading (line 3); if the check succeeds, it sends out its ID (line 4). Then, it listens to all the strings sent by the other tags (line 6). Finally, for each listened ID that a is supposed to trace (line 8), it increases the corresponding counter (line 9).

The step on line 3 performs a pseudorandom process to decide if the tag has to answer to the query. This process is based on the value t (provided by the reader) and on k_a, the random seed shared by the tag a with the control center. The value k_a is used to avoid a simple attack, where $\mathcal{ADV}$ can substitute a genuine tag with a bogus one. In fact, without the random seed $\mathcal{ADV}$ could forge a bogus tag just eavesdropping and replicating a honest tag ID. The bogus tag could hence be used in place of the honest one. With the use of k_a, $\mathcal{ADV}$ cannot replicate the correct behavior of a tag just forging the eavesdropped ID.

In order to provide each tag with the set of IDs it is responsible for, the system should be suitably set up before shipping, with a simple procedure that: scans all the tags, designs and assigns the set of IDs each tag has to trace, and activates the TIP mechanism. This is particularly useful since the assignment of the set of IDs can be customized right before shipping, either if tags are put in pallets, or in boxes, or in stands.

Analysis. Let us assume that each tag is traced by at least one other tag (i.e., $\bigcup_{i\in\mathcal{S}} T_i = \mathcal{S}$, where $\mathcal{S}$ is the complete set of tag identifiers in the system and T_i is the set of tags traced by tag id_i). If we define the event G as: "a counter will not be increased", than event G shows the same probability to occur of the event F in (2), that is:

$$\Pr[G] = (1-p)^L. \tag{3}$$

However, with respect to the previous protocol, in this case we have a different number of tags that trace a given ID, and a different level of *grain* of the tracing. In the following we analyze both this differences.

Analysis. The number of tags tracing a given ID depends on how the IDs to be traced are assigned to tags. We analyze two choices: the first is a deterministic one; the second is a probabilistic one. In the former case, the administrator partitions the tags in such a way that a tag is traced by at least another tag. In the latter case, each tag has to be assigned a number of IDs equal to $c\log n$ ($c > 1$) for the probability that a tag is not traced to be upper bounded by n^{-c}. Note that the deterministic assignment can help saving memory on tags; indeed, one ID to be stored on each tag is the bare minimum to provide coverage of the entire tag set. In the second case, trace redundancy is provided, but at the expense of a (little) higher memory overhead.

As for the *grain* of the tracing, compared to the digest answer, in this solution each tag has a counter for every tag to trace. Hence, it is possible to keep trace of the number of times *a specific tag* has been recorded OR (equivalently) it did not answer. Finally, we remind that the number of tags traced by a tag is a protocol parameter.

5.3. Dynamic Counter. The steps of the dynamic counter protocol are described in Algorithm 4. The operations are basically the static counter protocol until the tag listens to the string broadcast by the other tags (Algorithm 4, line 6). In fact, the difference between this protocol and the previous one is in the way broadcast IDs are managed. While the static counter protocol traces a fixed set of tags for all the readings, the dynamic counter protocol pseudorandomly selects the tags to be traced at each round. In particular, the set of tags that tag a has to trace for the current round is determined using a pseudorandom function and the shared random seed (resp. *CheckIfToTraceNow* and k_a of line 8). Hence, using the dynamic counter protocol, the attack of substituting a genuine tag with a bogus one is even more challenging for $\mathcal{ADV}$.

If we denote with M the event that a tag a is traced by at least another tag in the system (incrementing the related counter), two independent conditions have to be verified for this event to occur, (1) tag a is required to pass the probabilistic filter in the transmission phase, and, (2) receiving tags have to select a for tracing. Let us denote with q the probability that a replying tag is traced by at least one receiving tag. Hence by

$$\begin{aligned}\Pr[M] &= \Pr[(\mathit{Pass\ Tx.filter}) \wedge (\mathit{Pass\ Rec.filter})] \\ &= \Pr[\mathit{Pass\ Tx.filter}] * \Pr[\mathit{Pass\ Rec.filter}] \\ &= p * (1 - \Pr[\neg \mathit{Pass\ Rec.filter}]) \\ &= p * \left(1 - (1-q)^{n-1}\right) \\ &\approx p * (1 - \exp\{-qn\}).\end{aligned} \tag{4}$$

Finally, the probability of the event Q (i.e., the tag is not traced for L consecutive rounds) can be computed as:

$$\Pr[Q] = (1 - \Pr[M])^L \approx (1 - p * (1 - \exp\{-qn\}))^L. \tag{5}$$

5.4. Logger. The logger protocol is detailed in Algorithm 5. The logger protocol is similar to the dynamic counter protocol but differs with respect to the operations it performs when tracing a tag. In particular, each time a tag has to be traced, a memory slot is allocated, recording both the ID of the traced tag, as well as the current value t—via the *RecordTimeForId()* function (Algorithm 5, line 9).

Analysis. The analysis of this case is equal to the analysis of the dynamic counter previously shown. Indeed, for the ID of the tag to be stored, it has to pass two (independent) filters. The first screening is decided within the tag itself; the second one requires other tags to decide to store the ID of the received tags.

However, note that on the one hand the *grain* of the tracing dramatically improves. Indeed, not only the ID of a traced tag is stored, but the information related to the round where the tracing took place is stored as well. This can reveal exactly when and for how long a given tag has been removed (namely for which readings it was missing). On the other hand, the amount of required resources increases, since every time a tag is recorded a memory slot containing both the ID of the recorded tag and the current value for t is allocated.

```
        Data:
        id_a {id of tag a}
        k_a {shared random seed of tag a}
(1) begin
(2)     t ← ReceiveTimeFromReader;
(3)     if CheckAnsweringTime (t, k_a) then
(4)         Send (id_a);
(5)     end
(6)     S ← ReceiveStringsFromTags;
(7)     foreach id ∈ S do
(8)         if CheckIfToTraceNow (id, k_a, t)
            then
(9)             RecordTimeForId (id,R_t);
(10)        end
(11)    end
(12) end
```

Algorithm 5: Logger.

6. Simulations and Discussion

In this section, we report the simulation results for our protocols and expound a side-by-side comparison of each proposal introduced above.

6.1. Simulation Results. To support the analysis provided in Section 4, we simulated a typical environment of n tags that have to be traced: at each reader query, a tag has probability p to reply, sending its ID. As stated above (see (2)), low values for p increase the $\mathcal{ADV}$ probability to keep a tag far from its logging-tags without being detected, while higher values reduce the chances of a successful attack. The simulation has mimicked the evolution of the system. Each reported point in the following figures is the averaged value of 200 simulations. The assessed parameters were: the average number of times a tag replied; how many times and for how long a tag did not reply to a query—silent rounds, and, the number of memory slots needed by a tag (that we will refer as *queue length* in the following). In particular, the silent rounds of a tag are critical since they can be exploited by an $\mathcal{ADV}$ to remove a tag without being detected by our protocols. Those rounds, then, can be used to evaluate the detection probability of a successful attack. As for the parameters that influenced the simulation, the probability p—that is, the probability for a tag to broadcast its ID—was set equal to s/n, where s is the average number of tags that are supposed to broadcast their ID—a design parameter. Another relevant parameter is the probability q—that is, the probability that a tag's broadcast ID is retained, together with (for the logger protocol only) the time of the broadcast (t)—set to r/n, where r is the average value of tags that are required to store that value (tuple for the logger protocol).

Static Counter. Figure 2 compares the escaping detection probability for attacks lasting a given period against the static counter protocol: the x axis represents the duration of the attack in terms of number of rounds; the y axis is the probability $\mathcal{ADV}$ has to escape detection. The figure contains different plots, since we compared analytical values (obtained with (2)) with the simulated results, for several values of parameter p and $n = 500$. We can observe how the p value directly affects the detection probability: detection probability of attacks with the same length dramatically increases with the increase of p (i.e., attacks of 50 rounds

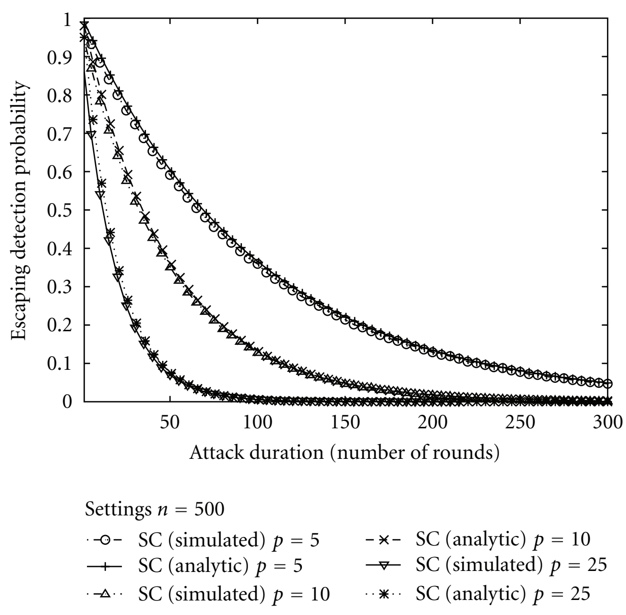


FIGURE 2: Static counter (SC) effectiveness.

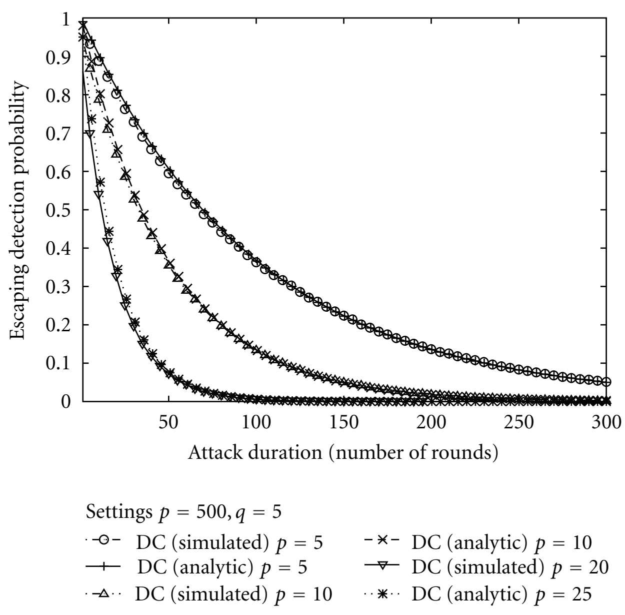


FIGURE 3: Dynamic counter (DC) effectiveness.

remain undetected with probability .59, .35 and .07 for an answer probability of .01, .02, and .05, resp.).

Dynamic Counter. Figure 3 shows the results of the same test when adopting the dynamic counter scheme. In this experiment, we fixed $n = 500$ and $r = 5$. Hence, on the average, each answering tag has been logged by 5 tags (logging probability is $q = r/n = 0.01$). The simulated results are compared with the analytical values obtained with (5).

Static versus Dynamic Counter. To show the intuition that both static counter and dynamic counter obtain almost the same security performances, we compared further simulation results in Figure 4. The plots were obtained fixing the two parameters $s = 5$ and $r = 5$ while simulating the system for several values of n—yielding $p = q = 0.05, 0.01$, and 0.005 for $n = 100, 500$, and 1000, respectively. Observing the plots, it can be seen that the difference (as for detection capability) between dynamic counter and static counter is almost negligible. This fact confirms the result of (4).

Static Counter. To highlight the relevance of probability p in the effectiveness of the proposed solution—as well as the induced overhead—, we can observe the plots in Figure 5. In these plots, we reported the results of further experiments carried out for the static counter scheme: we fixed the parameter s (to $s = 5$ and $s = 25$, resp.) and evaluated the detection probability varying the attack duration and the number of tags. Since s is fixed and the reply probability is $p = s/n$ than the larger n, the smaller the answer probability and, eventually, the longer the silence periods of each tag. Comparing the two curves, on the one hand it is evident that for small values of s, the probability that a long lasting attack escapes detection is high even for small values of n. On the other hand, with $s = 25$ the probability that attacks of more than 100 rounds are undetected drops under 0.1 even for $n = 1000$. Considering that with $p = 25$, on the average only 25 IDs are transmitted at any round, we can conclude that the proposed scheme has a very good trade off between generated traffic and effectiveness.

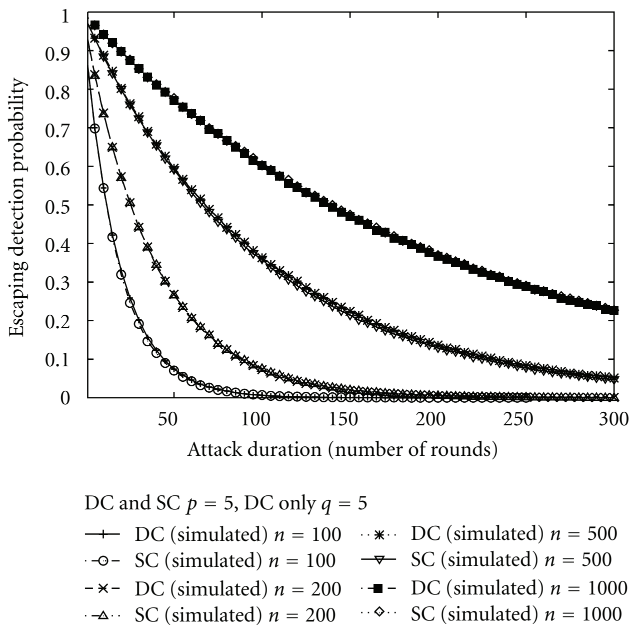
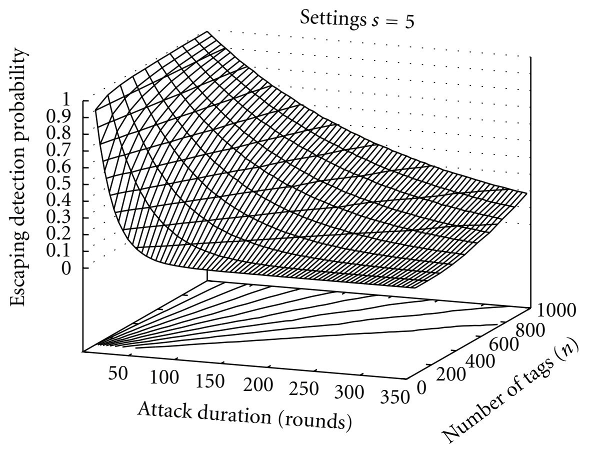
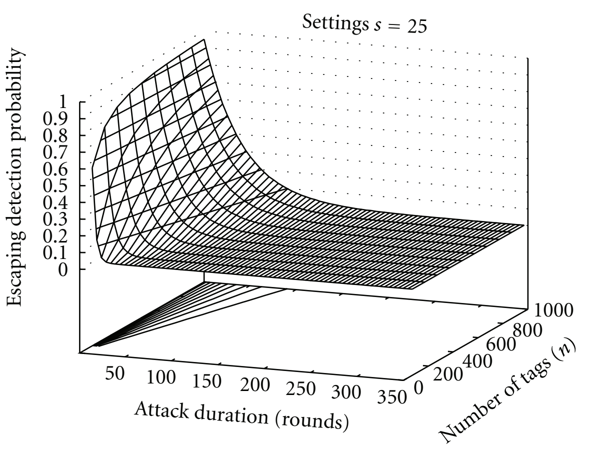


FIGURE 4: Static counter (SC) and dynamic counter (DC) security performances comparison.

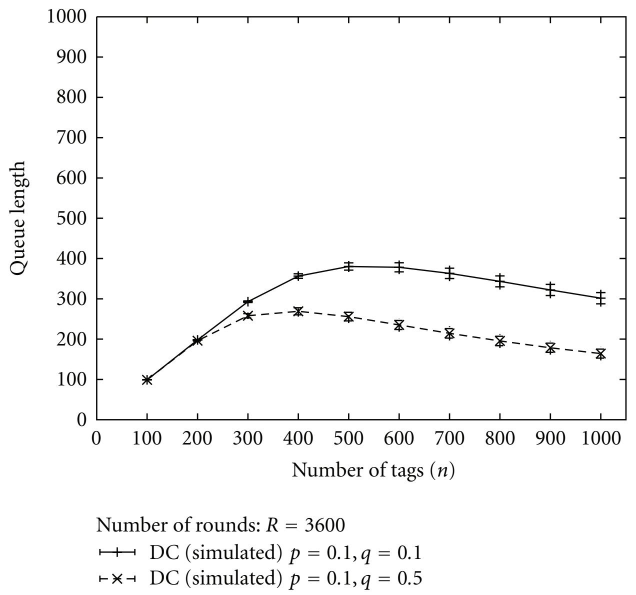


(a) SC $s = 5$

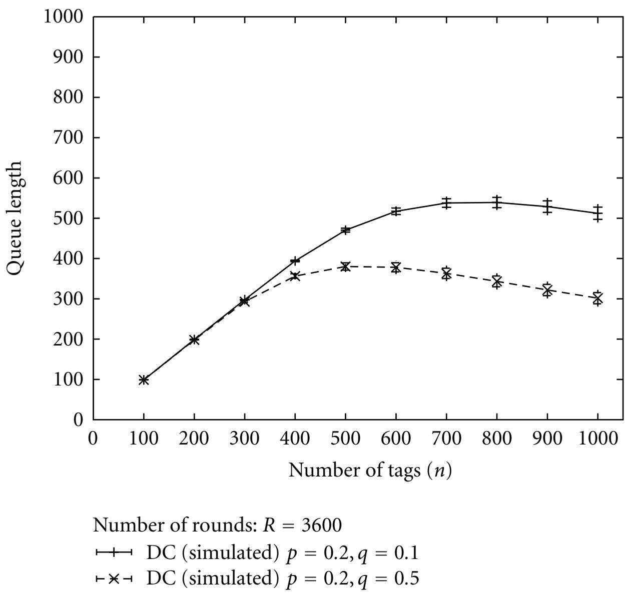


(b) SC $s = 25$

FIGURE 5: Static counter (SC) security performance: effects of parameter p.



(a) DC $s = 10$



(b) DC $s = 20$

FIGURE 6: Dynamic counter (DC), experiments to show the impact of parameter n on tag's queue length. Average value and standard deviation.

Dynamic Counter (and Logger). Figure 6 summarizes simulation results about the estimation of queue length for the dynamic counter —note that the same results can be extended to the logger protocol, as seen from the analysis in Section 5.4. As predicted in Section 5.3, as times goes by, every tag has to keep trace of new tags' IDs and, then, extend the length of its queue. The plots in figure show the average queue size for different settings. To obtain these plots, we simulated the life of the system for a period of $R = 3,600$ rounds (we recall that the actual length of a round has to be tuned according to security requirements) and counted the number of IDs each tag logged in that period. In each simulation, we fixed the parameters s and r while increasing the number of tags in the system, to see how the parameter n affects the queue length. The plots show the average value obtained with 200 simulations performed with different pseudorandom seeds and also report the experienced standard deviation—error bars. The figures show that the standard deviation is really small: this means that reported average values are accurate and that queue length has little fluctuation among the tags. Note that queue lengths are equivalent for the settings $s = 10, r = 10$ and $s = 20, r = 5$, as a direct consequence of the analysis in Section 5.3. Another consideration is that, since $R = 3,600$, for $n = 100, 200$ every tag eventually traces any other tag at least once: for those parameters, queue length tends to

Table 2: Protocols comparison: properties.

Protocol	Integrity	Tampering detection grain	No detection probability	Resilience to tag reading failure
Digest answer	yes	low	$(1-p)^L$	No
Static counter	yes	medium	$(1-p)^L$	Yes
Dynamic counter	yes	medium	$(1-p(1-\exp(-qn)))^L$	Yes
Logger	yes	high	$(1-p(1-\exp(-qn)))^L$	Yes

Table 3: Protocols comparison: overhead.

Protocol	Queue length (bit)	Messages sent	Messages received	Computations (hash)
Digest answer	$O(1)$	$R \cdot p$	$R + R(n-1) \cdot p$	R
Static counter	$O(1)$	$R \cdot p$	$R + R(n-1) \cdot p$	R
Dynamic counter	$O((n-1) \cdot (\log R/n))$	$R \cdot p$	$R + R(n-1) \cdot p$	$R + R(n-1) \cdot p$
Logger	$O(R \cdot (\log n + \log R))$	$R \cdot p$	$R + R(n-1) \cdot p$	$R + R(n-1) \cdot p$

the maximum value, that is $n-1$. This confirms the intuition that extending the lifetime of the system (namely increasing the number of rounds R), causes every tag to log any other tag.

6.2. Protocol Comparison. In Tables 2 and 3, we compare the proposed protocols. In particular, while all the proposed protocols can detect (probabilistically) whether the system integrity has been preserved, they can do that with different degrees of accuracy (Table 2) and incurring different overhead (Table 3). We recall that R is the number of rounds that the system has to monitor.

Table 2 shows a comparison on the protocols properties and summarizes the detection probability discussed in the previous sections. The digest answer protocol can only detect whether the integrity has been preserved or not; however, it does not provide any information neither on the tags that have been compromised, nor on the round when the compromising took place. Both the static counter and the dynamic counter protocols provide information on which of the tags have been compromised, but not on the round of compromising. Finally, the logger protocol is able to detect which tags have been compromised as well as the corresponding round the compromising took place.

As for the detection probability, both the digest answer and the static counter are characterized by the same success rate in detecting compromising. Also the dynamic counter and the logger have the same detection success probability. Moreover, the detection probability of both logger and dynamic counter is slightly lower than the detection probability provided by digest answer and static counter. However, note that as n increases the differences between these two probabilities become quickly negligible—the difference reduces exponentially fast. The last column of Table 2 describes the resilience of the protocol against a tag reading failure. In particular, the digest answer protocol is not resilient to a single reading failure. Assume the reader does not get the answer of a tag (for any reason, including communication failure), this would result in a completely useless output for the digest answer protocol. In fact, the tags set will result as tampered while only a single read has been lost. Furthermore, the protocol would not be able to give any other useful information—for example, to prove the presence of all the other tags rather than the one for which the communication failure occurred. However, all the other protocols would be able to give some useful output even in the case a failure happens during a single tag reading. As an example, in the static counter, the protocol would still be able to correctly trace the presence of all the other tags not affected by the reading failure.

Table 3 shows the comparison of the protocols in terms of the size of the queues—that is, the information stored by each tag—and other overheads for each tag in terms of number of sent messages, received messages, and computations—accordingly to Algorithms 2, 3, 4, and 5. While the number of messages sent and received is the same for all the four protocols, queues length differ. As for the messages, note that the number of messages sent by each tag is just influenced by the probability p. As for the messages received, note that at each round the reader broadcasts the value t—that accounts for the addendum R—, while each tag has to examine the messages received (i.e., relied by the reader) by $(n-1)p$ other tags per round, hence accounting for the other addendum $R(n-1)p$.

As for the queue length, both the digest answer and the static counter require just a constant number of memory slots to be executed. As for the dynamic counter, the worst case scenario depicts a situation where a tag has initialized a memory slot for each of the other tags in the system and each tag has been traced an equal number of times. Hence, the memory requirements account to $(n-1)(\log R/n)$ bits. The logger is the more memory demanding protocol. Indeed, it requires a number of memory slots that is proportional to the number of rounds (R) sustained by the system. That is, for each round and for each traced tag, it requires to trace the tag's ID, as well as the current value for t. Hence, requiring an upper bound of $R(\log n + \log R)$ bits.

Finally, as for computations we have decided to focus on the more expensive function performed by the tag: hash computation. We have decided to count neither the XOR operations (required by the digest answer) nor the counter increasing (required by both static counter and dynamic counter) since the corresponding cost in terms of energy consumption and required time is negligible when compared to the hash function. Again, we have that the four protocols can be divided in two classes. Digest answer and static counter belong to the first one, that is characterized by a number of hash computations that is equal to the number of rounds experienced by the system. The second class is composed by dynamic counter and logger. For them, computation overhead is given by R hash, plus a number of hash that is a fraction of $(n-1)$, that is $(n-1)p$.

In summary, when it comes to select a protocol for a real deployment, the choice mainly depends on two competing parameters: the grain of the desired detection and the amount of memory available on tag. Indeed, to have a fine grain of tampering detection (i.e., to know which tag and at which round it has been compromised), either the dynamic counter or the logger protocol can be selected; the latter providing the finest grain of tampering detection. However, the features of the logger comes at a cost of a few more computations and a higher memory requirement. More precisely, the dynamic counter memory requirement is, in the worst case, $O((n-1)\cdot\log R)$—assuming a tag is tracing all the others $(n-1)$ tags, and for each of the R readings all the relative counters are incremented. On the other hand, the memory requirement for the logger is, in the worst case, $O(R\cdot(\log(n)+\log R))$—that is, also assuming a tag is tracing all the other $(n-1)$ tags, and for each of the R readings all the information are stored: tag id (size of $\log n$) and current round (size of $\log R$).

If the available tags only have few memory slots, the choice must be oriented towards either the digest counter or the static counter, that trade off a coarse grain of detection with just a constant number of memory slots. In particular, the former requires the least amount of resources, while providing just a bit of information: whether the integrity of the system has been compromised or not.

7. Concluding Remarks

In this paper, we propose to leverage the RFID reader as a communication relay, to have tags cooperating. The effectiveness of this approach has been shown addressing an exciting application for RFID: tampering detection. In particular, we have provided a set of probabilistic protocols that can detect the absence of a given tag from a specific—yet quite common—deployment scenario. The protocols can leverage the features of the several classes of RFID available on the market. In particular, the memory required range from few memory slots to a number of memory slots that is proportional to the number of rounds the presence of a tag is going to be tested. The more resources a protocol requires, the finer is the grain of the tampering detection.

A thorough analysis for the tampering detection probability is provided. Finally, simulations support the analytical results. We believe that the proposed solution based on tags cooperation could open up further research directions in the field.

Acknowledgments

This work was partly supported by the Spanish Ministry of Education through projects TSI2007-65406-C03-01 “E-AEGIS” and CONSOLIDER CSD2007-00004 “ARES”, and by the Government of Catalonia under Grant 2009 SGR 1135. We would like to thank the anonymous reviewers for their comments that helped improving the paper.

References

[1] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, John Wiley & Sons, New York, NY, USA, 2003.

[2] C. Dehollain, M. Declercq, N. Joehl, and J.-P. Curty, “A global survey on short range low power wireless data transmission architectures for ISM applications,” in *Proceedings of the International Semiconductor Conference*, pp. 117–126, 2001.

[3] “Gildas Avoine,” 2009, http://www.avoine.net/rfid/.

[4] A. Juels, “RFID security and privacy: a research survey,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, 2006.

[5] D. Molnar and D. Wagner, “Privacy and security in library RFID: issues, practices, and architecture,” in *Proceedings of the ACM Computer and Communications Security (CCS ’04)*, pp. 210–219, 2004.

[6] A. Juels and S. Weis, “Authenticating pervasive devices with human protocols,” in *Proceedings of the Advances in Cryptology (CRYPTO ’05)*, vol. 3126 of *Lecture Notes in Computer Science*, pp. 293–308, 2005.

[7] H. Gilbert, M. Robshaw, and H. Sibert, “An active attack against HB^+—a provably secure lightweight authentication protocol,” Cryptology ePrint Archive, Report 2005/237, 2005.

[8] J. Bringer, H. Chabanne, and E. Dottax, “B^{++}: a lightweight authentication protocol secure against some attacks,” in *Proceedings of the IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU ’06)*, vol. 2006, pp. 28–33, 2006.

[9] S. Piramuthu, “HB and related lightweight authentication protocols for secure RFID tag/reader authentication,” *Proceedings of the Collaborative Electronic Commerce Technology and Research (CollECTeR ’06)*, pp. 239–247, 2006.

[10] G. Tsudik, “YA-TRAP: yet another trivial RFID authentication protocol,” in *Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW ’06)*, vol. 2006, pp. 643–645, 2006.

[11] M. Burmester, T. Van Le, and B. De Medeiros, “Provably secure ubiquitous systems: universally composable RFID authentication protocols,” in *Proceedings of the Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm ’06)*, pp. 1–10, 2006.

[12] G. Tsudik, “A family of dunces,” in *Proceedings of the 7th Workshop on Privacy Enhancing Technologies (PET ’07)*, vol. 2006, pp. 45–61, 2007.

[13] T. van Le, M. Burmester, and B. de Medeiros, “Forward-secure RFID authentication and key exchange,” Cryptology ePrint Archive, Report 2007/051, 2007.

[14] R. Di Pietro and R. Molva, “Information confinement, privacy, and security in RFID systems,” in *Proceedings of the 12th European Symposium On Research In Computer Security (ESORICS ’07)*, pp. 187–202, 2007.

[15] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic approach to "Privacy- Friendly" tags," in *Proceedings of the MIT RFID Privacy Workshop*, 2003.

[16] G. Avoine and P. Oechslin, "A scalable and provably secure hash-based RFID protocol," in *Proceedings of the International Workshop on Pervasive Computing and Communication Security (PerSec '05)*, vol. 2005, pp. 110–114, 2005.

[17] M. Conti, R. Di Pietro, L. V. Mancini, and A. Spognardi, "RIPP-FS: an RFID identification, privacy preserving protocol with forward secrecy," in *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '07)*, pp. 229–234, 2007.

[18] M. Conti, R. di Pietro, L. V. Mancini, and A. Spognardi, "eRIPP-FS: enforcing privacy and security in RFID," *Security and Communication Networks*, vol. 3, no. 1, pp. 58–70, 2010.

[19] S. Spiekermann and S. Evdokimov, "Critical RFID privacy-enhancing technologies," *IEEE Security and Privacy*, vol. 7, no. 2, Article ID 4812158, pp. 56–62, 2009.

[20] A. Juels, ""Yoking-Proofs" for RFID tags," in *Proceedings of the IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04)*, pp. 138–143, 2004.

[21] L. Bolotnyy and G. Robins, "Generalized "Yoking-Proofs" for a group of RFID tags," in *Proceedings of the Annual International Conference on Mobile and Ubiquitous Systems*, vol. 2005, pp. 1–4, 2006.

[22] M. Burmester, B. Medeiros, and R. Motta, "Provably secure grouping-proofs for RFID tags," in *Proceedings of the 8th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications (CARDIS '08)*, pp. 176–190, 2008.

[23] A. Juels, R. Pappu, and B. Parno, "Unidirectional key distribution across time and space with applications to RFID security," in *Proceedings of the 17th Conference on Security Symposium (USENIX '08)*, pp. 75–90, 2008.

[24] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[25] T. Staake, F. Thiesse, and E. Fleisch, "Extending the epc network: the potential of rfid in anti-counterfeiting," in *Proceedings of the ACM Symposium on Applied Computing (SAC '05)*, vol. 2, pp. 1607–1612, 2005.

[26] B. King and X. Zhang, "Securing the pharmaceutical supply chain using RFID," in *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE '07)*, pp. 23–28, 2007.

[27] "EPCGlobal," http://www.epcglobalinc.org/home.

[28] D. Zanetti, L. Fellmann, and S. Capkun, "Privacy-preserving Clone Detection for RFIDenabled Supply Chains," in *Proceedings of the EEE International Conference on RFID*, vol. 2006, pp. 37–44, 2010.

[29] "International Organization for Standardization," 2009, http://www.iso.org.

[30] "NXP," 2009, http://www.nxp.com.

[31] M. O'Neill, "Low-cost SHA-1 hash function architecture for RFID Tags," in *Proceedings of the Conference on RFID Security*, 2008.

[32] R. -I. Paise and S. Vaudenay, "Mutual authentication in RFID: security and privacy," *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS '08)*, pp. 292–299, 2008.

[33] A. Shamir, "Squash: A new one-way hash function with provable security properties for highly constrained devices such as RFID tags," in *Proceedings of the Conference on RFID Security*, p. 136, 2007.

[34] G. Karjoth and P. Moskowitz, "Disabling RFID tags with visible confirmation: clipped tags are silenced," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES '05)*, pp. 27–30, 2005.

[35] J. Myung, W. Lee, J. Srivastava, and T. K. Shih, "Tag-splitting: Adaptive collision arbitration protocols for RFID tag identification," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 763–775, 2007.

[36] Amin N. and P. W. Lin, "Anti-collision protocol development for passive RFID tags," in *Proceedings of the 7th WSEAS International Conference on Applied Informatics and Communications*, pp. 393–397, 2007.