Probabilistic Yoking Proofs for Large Scale IoT Systems

José M. de Fuentes*, Pedro Peris-Lopez, Juan E. Tapiador, Sergio Pastrana

Department of Computer Science, Universidad Carlos III de Madrid
Avda. Universidad, 30, 28911 Leganes, Madrid, Spain

Abstract

Yoking (or grouping) proofs were introduced in 2004 as a security construction for RFID applications in which it is needed to build an evidence that several objects have been scanned simultaneously or, at least, within a short time. Such protocols were designed for scenarios where only a few tags (typically just two) are involved, so issues such as preventing an object from abandoning the proof right after being interrogated simply do not make sense. The idea, however, is very interesting for many Internet of Things (IoT) applications where a potentially large population of objects must be grouped together. In this paper we address this issue by presenting the notion of Probabilistic Yoking Proofs (PYP) and introducing three main criteria to assess their performance: cost, security, and fairness. Our proposal combines the message structure found in classical grouping proof constructions with an iterative Poisson sampling process where the probability of each object being sampled varies over time. We introduce a number of mechanisms to apply fluctuations to each object's sampling probability and present different sampling strategies. Our experimental results confirm that most strategies achieve good security and fairness levels while keeping the overall protocol cost down.

Keywords: Yoking proofs, Grouping proofs, RFID security, IoT security

*Corresponding author
Email addresses: `jfuentes@inf.uc3m.es` (José M. de Fuentes), `pperis@inf.uc3m.es` (Pedro Peris-Lopez), `jestevez@inf.uc3m.es` (Juan E. Tapiador), `spastran@inf.uc3m.es` (Sergio Pastrana)

1. Introduction

In recent years, various technical developments coupled with the proliferation of smart devices and the widespread availability of communication networks are contributing to what is sometimes called a "smart world". This notion encompasses applications within smart cities, environment management, security and emergencies, e-health, and many others. As more and more devices are equipped with computing capabilities and network connectivity, the notion of an Internet-of-Things (IoT) is becoming increasingly real. Security aspects in the IoT are receiving much attention by the research community. The constrained computational capabilities, battery limitations, and network resources of these devices (e.g, sensors, actuators, or mobile phones) are challenging factors to propose suitable security mechanisms [1].

One of the underlying technologies in the IoT context is Radio-Frequency IDentification (RFID) [16]. RFID systems are composed of tags, readers, and a back-end database. The tags (or transponders) are small and affordable devices consisting of a microchip and an antenna that can be attached to almost every object (i.e., a person, an animal, or an item) for the purpose of its identification. The broad adoption of this promoting technology is subject to the development of security solutions [6, 11]. In this context, lightweight cryptography algorithms have been proposed to address different aspects of their security and privacy requirements [20, 21].

Mobility is an essential feature of many IoT devices [7]. Typically, a device may be moved from one point to another in a short time. Furthermore, there are applications such as stock control in a supermarket or asset tracking at hospitals that require to check the simultaneous presence of a group of devices. To address this need, several mechanisms have been proposed to build an evidence (proof) that a set of devices remain together at a certain time. Such proofs are termed "yoking proofs" when only two devices (e.g., tags or sensors) are involved [10]. Although they were initially conceived for just two devices, many authors have generalized them for larger population of devices, calling them "grouping proofs" [2, 19, 15].

Two relevant factors when analyzing a yoking/grouping proof scheme are its security and efficiency. On the one hand, these protocols must prevent an adversary to steal an IoT device (or borrow it for a sustained period of time) while passing unnoticed to the verifier (i.e., an off-line back-end database connected through a secure channel to the RFID reader). On the other hand, the protocol must be efficient in the sense that it should involve as little computational and network resources as possible for a complete protocol execution [18].

1.1. Contribution

Existing grouping proofs were designed taking into consideration that the tag population may consist of more than two devices but that it is not voluminous. To understand this, we next present a motivating scenario. Let us consider, for instance, a library manager who is interested in guaranteeing that all books from the permanent collection are inside the building at all times, no matter if they are being read in a different floor. A grouping proof is considered to attest that all books are together within the range area at a particular time. For this purpose, a single RFID tag is attached to every book. The problem arises when the amount of books to control is very large (e.g., more than 100,000 titles). In these conditions, the completion of the proof would take substantial time. It must be noted that such a time could be reduced by using several readers, but this would be impractical in large scale scenarios such as the one described before.

This particularity can be exploited by an adversary. Specifically, once the tag is interrogated, she could take the book and leave until its next participation in the protocol. Since the population is quite large, the absence time for the adversary may be significant. In this particular example, even it the protocol prevents the adversary from taking the book and remaining undetected, it still allow her to leave the protocol for prolonged periods of time at will. The situation is even worse if the entity communicating with the tags (referred to as the Reader) is untrusted. This is a typical assumption in yoking proof scenarios. In such a case, the Reader could leave the stolen book out of the protocol for the maximum possible time–polling it at the beginning of a round and at the end of the next one.

To the best of our knowledge, existing grouping proofs generalize the initially proposed yoking proof for a population of $N = 2$ tags to a proof valid for a group of tags ($N \geq 2$) just by chaining all messages received from the involved entities. Unfortunately, as discussed in the previous example, existing grouping proofs are inappropriate when dealing with a large set of devices such as those existing in a library, warehouse, or the thousands of tiny items in a jewelry shop.

In this paper, we present the first probabilistic yoking proof with the aim of balancing security and performance and being suitable for large-scale systems. At high level, the approach is based on dividing the set of IoT devices (e.g., RFID tags or sensors) into several subsets with low cardinality and poll each subset in an unpredictable manner. The scheme works iteratively in a number of rounds, in such a way that potentially different subsets are rebuilt on each protocol round, thus reducing the amount of information available for the attacker. Our proposed yoking proof also deals with a certain notion of fairness to ensure that all tags have been interrogated a similar amount of times once the proof has been built.

The core of the protocol is the design of the sampling process at each round. We present a number of mechanisms to probabilistically select a subset of tags at each round, and to update the associated sampling probabilities for the next one. The inclusion of one mechanism or another in a PYP protocol influences the overall performance and should be done carefully, particularly because there are non-trivial trade-offs among cost, security, and fairness, so optimizing one of them may degrade another. In order to illustrate this, we introduce five representative sampling strategies that combine different selection and updating mechanisms and discuss the experimental results obtained with them.

2. Related work

In 2004 Juels introduced the concept of yoking proofs. It consists of an evidence that a pair of tags have been scanned simultaneously by a reading device. It is commonly assumed that the proof is verifiable at some time by an off-line trusted party —the verifier. On the other hand, passive tags are not equipped with a battery, impeding the use of a clock on-board, and communications must be initiated by the reader [10].

Saito and Sakurari showed how Juels proof [10] is vulnerable against replay attacks [22] —complementary replay and new attacks such as DoS or impersonation can be found in [2] and [3], respectively. To compact replay attacks, the authors presented a new yoking proof based on time stamps and introduced the concept of grouping proof as the generalization of yoking proofs for a set of N tags (i.e., $N \geq 3$). Unfortunately, time stamps are predictable and Saito and Sakurari proof is also vulnerable to replay attacks as shown in [19]. In the same work, Piramuthu proposed a modified version of the above mention scheme in which timestamps are suppressed by random numbers . Although this new proof is no-vulnerable to the attacks suffered by its predecessors, the scheme resulted vulnerable against the so called multi-proof session replay attack [17], which exploits the symmetry of all the exchanged messages.

Many other proposals have appeared in the literature in the last years like the ones recently proposed in the medical context [5, 14]. Some proposals focus on a particular aspects of the protocol –in the following we list the most relevant ones. For instance, there are proofs, that apart from the evidence generation, that aim at offering an anonymous identification [3, 15]. The efficiency in the identification is the goal of other proofs which employ a tree structure and guarantee anonymity [4]. There exist proposals in which the participating tags can be read in any order, which is called an order-independent protocol [12]. In [13], the authors address the existence of race conditions when multiple reader/tags are presented or the number of participating tags is unknown. Finally, we urge the reader to consult [8] where Hermans and Peeters introduce a specific privacy model for yoking proofs.

In the following we use the term "yoking proofs", which is the initial name given by Juels, to refer to as the proof that a group of tags (i.e., $\{T_1, T_2, \cdots, T_k\}$ and $k \geq 2$) were scanned roughly at the same time and communicated to each other. It must be noted that given the one-to-one nature of RFID communications, scanning a set of tags nearly at the same time would require to have nearly the same amount of readers. As this is impractical in large scale contexts, yoking proofs are usually built as a chain of answers from the polled tags. Note, too, that several terms have been used for the same concept in previous works, the most common being: grouping proofs [22], existence-proofs [19], clumping-proofs [17] and coexistence-proofs [13].

3. Building Blocks for Yoking Proofs

This Section introduces the main elements of the proposed probabilistic yoking proofs. First, Section 3.1 describes the system model assumed in this work. Afterwards, Section 3.2 presents the properties that a yoking proof protocol must meet. Section 3.3 describes the baseline yoking proof, which is a simple mechanism to build these proofs. The notation in use throughout this paper is shown in Table 1.

3.1. Model

There are four entities in the considered scenario, namely the tags, the reader, the verifier, and the adversary. First, let us assume a tag population $\mathcal{T}$, where $T_i \in \mathcal{T}$ for $i = \{1, \cdots, N\}$ and being $N \gg 1$. Each tag has a secret key K_i and an unique identifier ID_i. On the other hand, the reader R is the entity in charge of interrogating all tags. The verifier V is the receiver of the yoking proof and it is responsible for determining several aspects of the protocol execution.

Both tags and V are considered trusted. Particularly, it is assumed that V knows the secret key K_i for all tags. On the other hand, the reader R is

Table 1: Notation

Symbol	Meaning
$\mathcal{T}$	Set of tags
$\mathbf{T}_i$	Tag i
$\mathbf{R}$	Reader
$\mathbf{V}$	Verifier
$\{ \mathbf{M} \}_k$	Encryption of M with key k
$\mathbf{PR}^i$	Result of the execution of round i
$\mathcal{PR}$	Resulting yoking proof
$\mathcal{S}^{(r)}$	Vector that contains 1 at position i if T_i participates at round r
$\mathcal{TS}^{(r)}$	Vector that contains a random permutation of all tags such that $\mathcal{S}_i^{(r)}=1$
$\mathcal{P}^{(r)}$	Vector of the probability of each tag T_i participating at round r
$\mathbf{ATP}_V$	Anonymous timestamp given by V
$\mathbf{ID}_i$	Identity of tag i
$\mathbf{RN}_i$	Random number
$\mathbf{K}_i$	Private key of tag T_i

untrusted entity and acts as a proxy passing messages between entities. As a consequence of its questionable behaviour, R does not necessarily need to follow the protocol as specified by V.

Regarding the adversary Adv, it is an entity that is able to see the whole protocol execution, as well as intercept, block and inject packets to and from any participating tag.

3.2. Desirable properties

A yoking proof protocol must meet the four main properties introduced below:

- Simultaneity. All tags should be scanned in a (usually short) time interval ρ_t.
- Dependability (against false positives). It must not be possible for Adv to build a proof that attests that a given absent tag T_i was present at round r.
- Dependability (against false negatives). It must not be possible for Adv to make the system work as if a given present tag T_i were absent at round r while passing unnoticed to V.
- Privacy preservation. Only V must be authorized to determine whether tag ID_i is present or not at each moment.

3.3. Baseline Protocol

The goal of a yoking proof is to generate an evidence that a tag population has been roughly read at the same time. Standard yoking proofs are only valid when the set of tags is not voluminous. In the following this sort of schemes are called as "Baseline Yoking Proof" (BYP). Our proposed baseline scheme is inspired on ISO/IEC 9782-2, and, in particular, the process for an iteration with a tag follows the two-pass unilateral authentication scheme [9].

We define init(.) as the process in which for all $T_i \in \mathcal{T}$, the tuple $\{ID_i, K_i\}$ is initialized. For the description of the protocol and because of its generality, we symbolize $\{X\}_K$ as the encryption of the message X with the key K to provide confidentiality and integrity. The description of the BYP scheme is presented at the top of the Algorithm 1. We assume that before starting the protocol, the reader R receives from the off-line verifier (V) an anonymous timestamp (ATP_V). On the other, for the first tag interrogation, we set the token m_0 to the $NULL$ value ($m_0 = NULL$).

Algorithm 1 Yoking Proof for Small Scale Environments

```
1: function init($\mathcal{T}$)
2:     for each tag $T_i$ do
3:         $T_i \leftarrow \{ID_i, K_i\}$
4:     end for
5: end function

6: function baseline($\mathcal{T}$) —BASELINE YOKING PROOF ($BYP$)
7:     for each tag $T_i$ do
8:         $R \rightarrow T_i : ATP_V, m_{i-1}$
9:         $T_i$: Generate a random number $RN_i$
10:                and compute $m_i = \{RN_i, ATP_V, m_{i-1}, ID_i\}_{K_i}$
11:        $T_i \rightarrow R : m_i$
12:    end for
13: return $PR = \{ATP_V, m_1, m_2, \cdots, m_N\}$
14: end function
```

As previously mentioned the BYP scheme is ineffective in environments in which tag populations are enormous (i.e.., $N \gg 10^2$), like the ones that we can find in the majority of stock applications. Let us assume an adversary Adv who can eavesdrop all the exchanged messages for the generation of a yoking proof and is able to identify the tags that have already been identified from a particular time. Under this situation, Adv could exploit the absence time (time interval between two interrogations of a target tag) to take away a tag that has just been read. Since the set of tags is enormous, the absence time is not negligible, contrary what occurs when we deal with small set of tags (e.g., the yoking proof generated between my passport and my luggage at the airport).

4. Probabilistic Yoking Proofs

Motivated from all the above, to the best of our knowledge, we present the first Probabilistic Yoking Proofs (PYPs), which are suitable for large scale population of IoT devices. Section 4.1 introduces the preliminary underlying concepts of PYP. Next, Section 4.2 describes the proposed protocol for building yoking proofs. The criteria to assess the performance of the proposal is presented in Section 4.3.

4.1. Definitions

In order to introduce the scheme, firstly, we need to provide some definitions. Let us assume a tag population $\mathcal{T}$, where $T_i \in \mathcal{T}$ and $i = 1, \cdots, N$ and $N \gg 10^2$. The coverage of T_i at round r of the protocol is 1, mathematically $V_i^{(r)} = 1$, if the tag has been involved in the protocol. The coverage $\mathcal{V}^{(r)}$ of the protocol for the whole tag population $\mathcal{T}$ is equal to 1 if all tags have participated at least one time in the protocol. That is, the coverage of the PYP at iteration r can be defined as follows, where $\lfloor . \rfloor$ function rounds a number to the next smaller integer.

$$\mathcal{V}^{(r)} = \lfloor \frac{1}{N} \sum_{i=1}^{N} V_i^{(r)} \rfloor$$

On the other hand, the PYP has a vector of probabilities $\mathcal{P}^{(r)}$ at the r-th iteration of the proof, where each element $P_i^{(r)}$ represents the probability that a tag participates in the proof at that round. Mathematically,

$$\mathcal{P}^{(r)} = [P_1^{(r)} \;\; P_2^{(r)} \;\; \cdots \;\; P_N^{(r)}]$$

It must be noted that $\mathcal{P}^{(0)}$ is set to a constant value for all tags using the $InitProb$ function. Depending on the above vector of probabilities and following some of the tag selection strategies ($\mathcal{TSR}$) introduced in detail in the next section ($\mathcal{TSR} \in \{D\text{-}DS,\ D\text{-}R\text{-}DS,\ U\text{-}D\text{-}DS,\ U\text{-}D\text{-}R\text{-}DS,\ UD\text{-}D\text{-}R\}$), the tags are chosen for their participation in the protocol. This process is symbolized by the select function whose input parameter is $\mathcal{P}^{(r)}$ and it is also conditioned by the chosen strategy $\mathcal{TSR}$ —i.e., $\mathsf{select}(\mathcal{P}^{(r)})$. The result of applying this function is a vector $\mathcal{S}^{(r)}$ that takes 1 value at the i-th position (i.e., $S_i^{(r)} = 1$) if T_i is sampled; otherwise a zero value is stored on that position. The 1-norm of $\mathcal{S}^{(r)}$ is less or equal to the number of tags in the population (i.e., $\sum_{i=1}^{N} S_i^{(r)} \leq N$). After the execution of the select function, the set of sampled tags is labelled as $\mathcal{TS}^{(r)}$ in such a way that $\mathcal{TS}^{(r)} \subseteq \mathcal{T}$. Particularly $\mathcal{TS}^{(r)}$ is formed by a random permutation of those tags such that $S_i^{(r)} = 1$.

Finally we define an updating mechanism, called update(), that uses as input parameters the vector $\mathcal{P}^{(r)}$ and the output $\mathcal{S}^{(r)}$ of the select function. The output of this function consists on an updated version of $\mathcal{P}^{(r)}$. That is,

$$P^{r+1}(\mathcal{T}) \quad = \quad \mathsf{update}(\mathcal{P}^{(r)}, \mathcal{S}^{(r)}) = [P_1^{(r+1)} \quad P_2^{(r+1)} \quad \cdots \quad P_N^{(r+1)}]$$

4.2. Protocol description

After introducing all definitions above, we next sketch the proposed PYP protocol. A step-by-step descripton is provided in Algorithm 2. Each tag is initialized before the protocol execution with its identity ID_i and its private key K_i. Once the protocol starts, the probability of participation for all tags is set to an initial (default) value. The protocol works iteratively, sampling a subset of tags at each iteration (round). Subsets are determined by V at each round.

PYP keeps running until full coverage is achieved. (Note that this criterion can be easily adjusted.) In each round, three main steps are performed. First, the verifier V selects which tags must participate, taking into account the current vector of probabilities $\mathcal{P}^{(r)}$. The set of selected tags is then permuted by V to make the tag participation time less predictable. The so-formed set of tags is sent to the reader R. Note that such a set contains only the MAC addresses of the tags to poll, thus hiding each tag's real ID_i. This is a common approach for tag selection purposes. Using these addresses, R executes the baseline protocol (see Section 3.3). Upon completion, R sends to V the resulting set PR of responses for all participating tags. The verifier then updates the vector of probabilities and prepares for the next round.

Upon reaching full coverage, V is able to build the final result of PYP by chaining all intermediate results (PR). In this process, V can determine if the protocol execution meets the required settings, particularly if all tags have been sampled at each round in the specified order.

Algorithm 2 Probabilistic Yoking Proof for Large Scale Environments

```
function init($\mathcal{T}$)
    for each tag $T_i$ do
        $T_i \leftarrow \{ID_i, K_i\}$
    end for
end function
function probabilistic($\mathcal{T}$) —PROBABILISTIC YOKING PROOF (PYP)
    $\mathcal{P}^{(0)} = \mathsf{InitProb}(\mathcal{T}))$
    while $\mathcal{V}^{(r)} \neq 1$ do
        V: $\mathcal{S}^{(r)} \leftarrow \mathsf{select}(\mathcal{P}^{(r)})$
        V: $\mathcal{TS}^{(r)} = Permute(\{S_i^{(r)}/S_i^{(r)} == 1\})$
        V: $V \rightarrow R$: $\mathcal{TS}^{(r)}$
        R: $PR^{(r)} \leftarrow BYP(\mathcal{TS}^{(r)})$ –Execute Baseline Yoking Proof for $\mathcal{TS}^{(r)}$
        R: $R \rightarrow V$: $PR^{(r)}$
        V: $P^{(r+1)} \leftarrow \mathsf{update}(\mathcal{P}^{(r)}, \mathcal{S}^{(r)})$
        $r \leftarrow r + 1$
    end while
    V: $\mathcal{PR} = \{PR^1, PR^2, \cdots\}$
end function
```

4.3. Performance criteria

An optimal PYP protocol must be both efficient and secure. Furthermore, since the select() function may sample each tag several times, it would be desirable to guarantee that by the end of the protocol all tags have been interrogated the same number of times. We next describe each of these criteria in detail and discuss specific ways of quantifying them.

Cost The cost C of a PYP protocol is defined as the total number of messages sent by the reader to a tag during the proof construction. Note that in most protocols this is half the number of actual messages exchanged, since each interrogation implies an answer message from the tag to the reader. In this paper, we will express the cost as an overhead factor with respect to the BYP protocol. Since a population of $|\mathcal{T}| = N$ tags would require N interrogations by the BYP, a cost C means $C \cdot N$ interrogations.

Security Let Π_1 and Π_2 be two PYP protocols, and let A_1 and A_2 be the probability density functions of the times—measured as number of protocol interrogations—between two consecutive samplings of the same tag for Π_1 and Π_2, respectively. Intuitively, Π_1 is more secure than Π_2 if:

1. $\mathbb{E}(A_1) < \mathbb{E}(A_2)$, where $\mathbb{E}(A)$ represents the expected value of A; and
2. $\mathrm{Unc}(A_1) > \mathrm{Unc}(A_2)$, where $\mathrm{Unc}(A)$ is a measure of the uncertainty of A. This can be quantified using a measure of dispersion such as the standard deviation, or as the entropy:

$$H(A) = -\sum_x A(x) \cdot \log A(x) \tag{1}$$

The rationale is simple. On the one hand, the less the expected number of messages between two consecutive interrogations, the less the time an adversary could leave the protocol undetected. On the other hand, the langer the uncertainty about how those times are distributed, the more unpredictable is for an adversary to guess for how long he may withdraw.

Both the mean and the uncertainty could be combined into a single security measure, for instance:

$$\mathrm{Sec}(A) = \frac{\omega_u \cdot \mathrm{Unc}(A)}{\omega_e \cdot \mathbb{E}(A)} \tag{2}$$

where ω_u and ω_e represent appropriate scaling factors. This, however, may give rise to unreasonable comparisons among proposals, since both the mean and the uncertainty should be simultaneously minimized and maximized, respectively.

Fairness A PYP is perfectly fair if all tags are interrogated exactly the same number of times after the proof completion. Roughly speaking, if $I(x)$ is the probability distribution function of the number of times each tag has been interrogated, fairness could be measured as the distance from I to a Dirac delta distribution $\delta(\frac{C}{N} - x)$ centered in $\frac{C}{N}$. Any statistical distance could be used for this purpose (e.g., the well-known Kolmogorov-Smirnov test):

$$D = \max_x |I(x) - \delta(\frac{C}{N} - x)| \tag{3}$$

5. Secure tag sampling strategies

In this section, we introduce a family of probabilistic sampling techniques for the yoking protocol described in Section 3. We first describe a number of mechanisms that are used to adjust the sampling probability for each tag after being selected or not selected for a round. Subsequently we describe various strategies that combine these mechanisms.

5.1. Basic sampling mechanisms

All the tag sampling strategies proposed below rely on the idea of selecting tags following a classical Poisson sampling process. This is a simple selection process where each tag T_i in the population is sampled according to an independent Bernoulli trial. The probability of tag T_i being selected at round r is denoted by $P_i^{(r)}$ (recall Section 3.3). This probability is particular to each tag and is updated after each round. The purpose of such updates is to provide the overall protocol with an adequate level of security and fairness while keeeping the cost down.

We have explored several mechanisms to conduct the Poisson sampling process and to update the probabilities, both of sampled and unsampled tags. We next discuss is detail these mechanisms and their associated rationale:

Punish participation As the yoking proof will be completed only after all tags have participated at least once, a simple way of minimizing the overall cost of a protocol execution—measured as the total number of messages exchanged—consists of reducing the sampling probability P_i of all tags selected at a round r. Thus, once a tag participates in a round, its sampling probability is modified so that it is less likely to sample it in the following rounds. This can be done by simply decrementing its probability by a fixed amount:

$$P_i^{(r+1)} = P_i^{(r)} - \Delta_d \tag{4}$$

Reward non-participation Analogously to the previous case, another way of minimizing the protocol execution cost consists of boosting the sampling probability of each tag that has not participated in a round. Thus, all tags that are not sampled in a round become more likely to be sampled in the next one. As in the case above, this can be done by incrementing the sampling probability by a fixed amount:

$$P_i^{(r+1)} = P_i^{(r)} + \Delta_u \tag{5}$$

Randomize probability updates The calculation of punishment and rewards in the two mechanisms described above is a critical aspect, as it may impact both positively and negatively different performance criteria. For instance, if the sampling probability of a participating tag is reduced too aggresively, the cost would be certainly minimized, but an adversary would be provided with more information—in a probabilistic sense–about when it would be called again. Furthemore, the amount by which each tag's probability is increased or reduced should not remain fixed, as it would be difficult to keep this parameter secret from an observer. An alternative would be to randomize it, for example by a random amount inversely proportional to $P_i^{(r)}$. In the case of rewards to non-participating tags, the increase is given by:

$$P_i^{(r+1)} = P_i^{(r)} + U(0, 1 - P_i^{(r)}) \tag{6}$$

while for punishments we would have:

$$P_i^{(r+1)} = P_i^{(r)} - U(0, P_i^{(r)}) \tag{7}$$

where $U(a,b)$ represents a random number uniformly sampled in the interval $[a,b]$. This would contribute to make the sampling process less predictable to an observer.

Double sampling Even with randomized probability updates, basic Poisson sampling could still provide the adversary with some valuable information about when to safely leave (i.e., right after being called) and when to return before his absence being detected. In order to introduce further uncertainty for the adversary, the sample at each round may consist of both *some* of the tags selected by Poisson sampling *and* some of the tags that were *not* selected for this round. A simple method to produce such a combined sample is the following:

1. Let s_i^p be result of the Poisson sampling experiment for tag T_i, with $s_i^p = 1$ if T_i is selected and $s_i^p = 0$ otherwise.
2. Let s_i^b be result of a Bernoulli sampling experiment for tag T_i with fixed probability P_B for all tags, with $s_i^b = 1$ if T_i is selected and $s_i^b = 0$ otherwise.
3. Tag T_i is selected if $s_i^p \oplus s_i^b = 1$, where $\oplus$ represents the XOR logical operator.

The procedure described above modifies the actual probability of a tag being selected, as to be finally sampled, it must be sampled either by

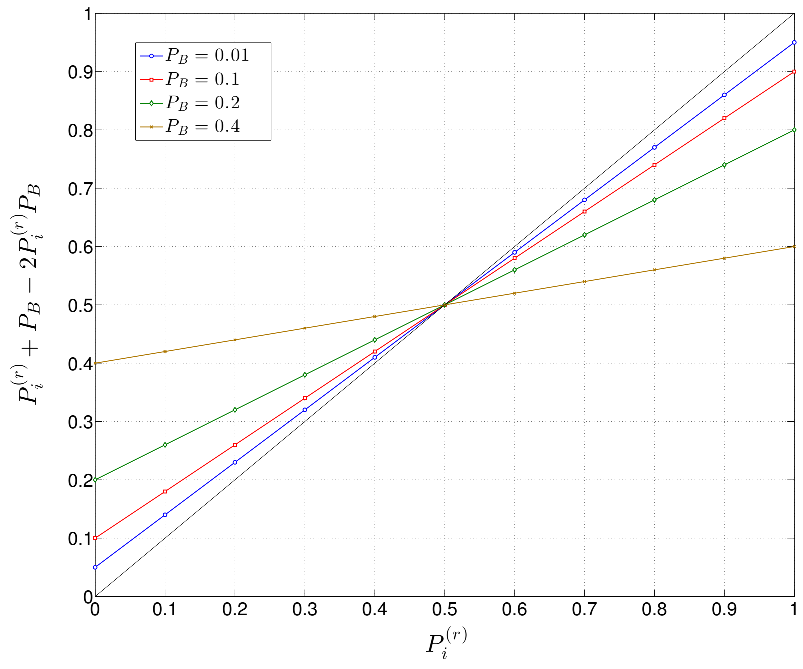

Figure 1: Effect of the double sampling strategy as a function of P_i and P_B.

the Poisson or by the Bernoulli process, but not by both. As these are conducted independently, the overall probability of being sampled at round r is given by

$$P_i^{(r)} + P_B - 2P_i^{(r)} P_B \tag{8}$$

For a fixed P_B value, the right-hand side term $P_B - 2P_i^{(r)} P_B$ affects differently to each tag depending on its $P_i^{(r)}$: those with lower probabilities get their chances increased while those with higher probabilites get their chances reduced (see Fig. 1).

5.2. Select and update strategies

The mechanisms described above can be combined in different ways to build sampling strategies with different levels of security, fairness, and cost. Algorithm 3 shows the general structure of the select() and the update() functions. In the former, a configuration parameter DS controls whether double sampling is conducted or not. As for the update() function, param-

Algorithm 3 General select() and update() functions. Global configuration parameters (DS, P_B, U, D, R, Δ_u, and Δ_d are not specified as inputs.

```
function select($\mathcal{P}^{(r)}$)
    $\mathcal{S}^{(r)} = \emptyset$
    for each tag $T_i$ do
        $s_i^p \sim$ Bernoulli($P_i$)
        $s_i^b \sim$ Bernoulli($P_B$)
        if ($\overline{\text{DS}}$ and $s_i^p = 1$) or (DS and $s_i^p \oplus s_i^b = 1$) then
            $\mathcal{S}^{(r)} \leftarrow \mathcal{S}^{(r)} \cup \{T_i\}$
        end if
    end for
return $\mathcal{S}^{(r)}$
end function
```

```
function update($\mathcal{P}^{(r)}, \mathcal{S}^{(r)}$)
    for each tag $T_i$ do
        if (D and $T_i \in \mathcal{S}^{(r)}$) then
            if (R) then
                $\delta = U(0, P_i^{(r)})$
            else
                $\delta = \Delta_d$
            end if
            $P_i^{(r+1)} = \max\{0, P_i^r - \delta\}$
        else if (U and $T_i \notin \mathcal{S}^{(r)}$) then
            if (R) then
                $\delta = U(0, 1 - P_i^{(r)})$
            else
                $\delta = \Delta_u$
            end if
            $P_i^{(r+1)} = \min\{1, P_i^r + \delta\}$
        end if
    end for
return $\mathcal{P}^{(r+1)}$
end function
```

eters U, D, and R determine if upward (U) and downward (D) probability updates are carried out, and whether these are randomized (R) or not.

After a set of experiments, we have chosen five representative tag sam-

Table 2: Tag selection strategies and the sampling mechanisms they incorporate.

Strategy	**Punish Participation**	**Reward Non-participation**	**Randomized Updates**	**Double Sampling**
D-DS	●			●
D-R-DS	●		●	●
U-D-DS	●	●		●
U-D-R-DS	●	●	●	●
U-D-R	●	●	●	

pling strategies that combines the mechanisms described above in different ways. The name of each strategy reflects the type of select() and update() strategy. For example, U-D-DS means that double sampling (DS) is conducted; that the probability of participating tags is reduced (D); that the probability of non-participating tags is increased (U); and that, in both cases, such updates are by a fixed amount (no R). Table 2 provides a summary of the mechanisms included in each one of them.

6. Evaluation

We next discuss various empirical results related to the performance of the five variants of sampling strategies introduced above. We first present the cost and fairness of each strategy, as most of the discussion about the results would be later very helpful for the security analysis.

All results have been obtained by simulation using a software prototype of the probabilistic yoking proof protocol. Experiments with different tag populations were conducted. For simplicity, and to facilitate the comparison with the baseline grouping proof protocol, all results discussed next correspond to a population of $|\mathcal{T}| = 1000$ tags. As it will be clear later, the conclusions are easy to extrapolate to populations of an arbitrary size. As for the parameters, we have experimentally determined that the best results were obtained for low values of Δ_u, Δ_d, and P_B (e.g., around 0.2). The figures provided in this section correspond to averages over various executions.

The last part of this section is devoted to determining whether the proposal meets the desired properties introduced in Section 3.2.

6.1. Cost

Table 3 shows the overhead incurred by each sampling variant as a multiplying factor with respect to the baseline yoking proof protocol (BYP). As

Table 3: Cost of each sampling strategy measured as the overhead with respect to the baseline selection protocol.

Strategy	**Cost** (C)
Baseline	1.0
D-DS	5.5
D-R-DS	5.6
U-D-DS	6.4
U-D-R-DS	3.3
U-D-R	4.8

it can be observed, the inclusion of one sampling mechanism or another has a significant impact in the overall cost of the protocol. Several conclusions can be drawn:

1. The U-D-R and U-D-R-DS variants constitute the cheaper strategies, requiring 3.3 and 4.8 times more messages than the baseline protocol. This is reasonable, as the inclusion of both the U and D mechanisms makes it easier for the protocol to sample rather disjoint subsets of tags at each round. Overall, this reduces the total number of messages required to sample all tags. Furthermore, double sampling is crucial to further reduce the cost.
2. When both U and D are included, non-randomized updates make the protocol very slow—compare U-D-R-DS (3.3) to U-D-DS (6.4). There is a simple explanation for this. Consider a tag $\mathcal{T}_i$ that at some round r has a very high probability $P_i^{(r)}$ of being sampled. If it gets sampled, at round $r+1$ its probability is decreased by Δ_d. If at round $r+1$ it does not get sampled, then $P_i^{(r+2)}$ is again increased. The overall effect is that after a few rounds most tags get sampling probabilities reasonably high and the overall strategy degenerates to an almost purely Bernoulli sampling (i.e., with all tags having roughly the same probability).
3. Finally, strategies that only apply either U or D—in our case, D-DS and D-R-DS—have a relatively high cost too. The reason for this is also straightforward. As non-participating tags are not rewarded, the sampling process progressively reduces the probability of all tags until it becomes uniformly low. At this point, the strategy degenerates to a Bernoulli sampling as in the case above. Randomization does not have much influence here as it does not prevent this effect from happening.

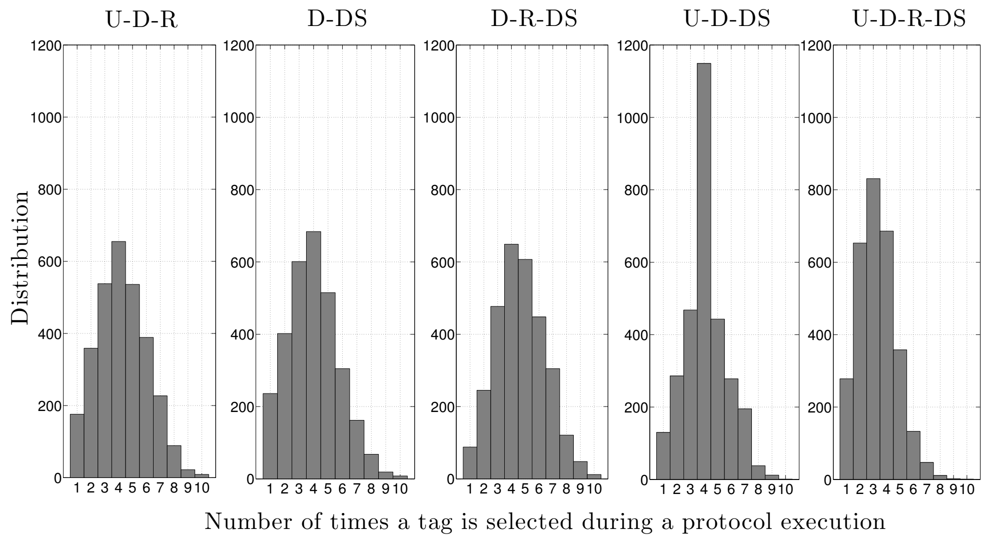

Figure 2: Fairness results for each sampling strategy.

6.2. Fairness

Fig. 2 shows the distribution of the number of times a tag is interrogated during a full protocol execution. These distributions must be interpreted considering the cost of each strategy (Table 3). Recall that, ideally, all tags should participate the same number of times—although, as discussed in Section 4.3, this would negatively impact the security of the scheme.

In general, all strategies are reasonably fair: for a population of 1000 tags and a number of messages ranging from 3300 (U-D-R-DS) to 6400 (U-D-DS), the number of times the same tag is interrogated ranges between 1 and 10, with the great majority of them falling in the $[1, 7]$ interval. All distributions are slightly biased towards the left (i.e., the have positive skew), meaning that most tags are interrogated less than the optimal number of times. An exception is the U-D-DS strategy, which performs significantly better than the others in terms of fairness. This is a side effect of the situation described in the previous section, namely that it rapidly degenerates to a purely Bernoulli process, which is optimal fairness-wise.

6.3. Security

We have first measured the number of messages between two consecutive interrogations of the same tag for all strategies, averaging the result over several simulations. The obtained distributions are shown in Fig. 3

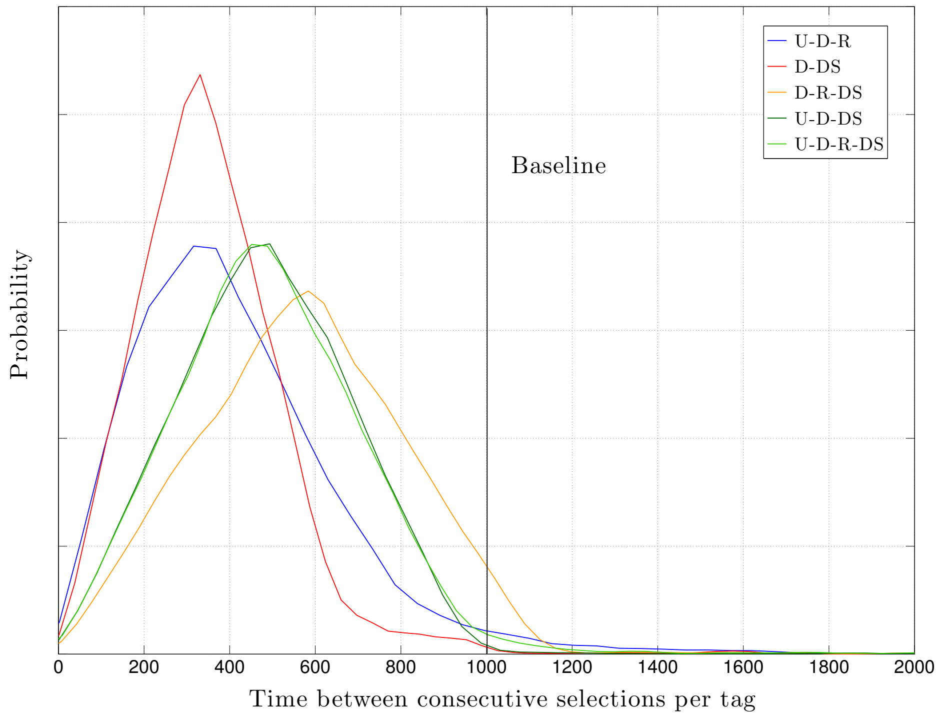

Figure 3: Distribution of the number of messages between two consecutive interrogations of the same tag.

and provide a graphical view of the different security levels offered by each strategy. For comparison purposes, the equivalent "curve" for the baseline strategy is also depicted.

The first noticeable fact is that all distributions are roughly normal with varying means and standard deviations. Both parameters are related to the security of each strategy. On the one hand, the lower the mean, the less time an adversary can leave without being detected. On the other hand, the larger the standard deviation, the more the chances of being interrogated sooner or later than expected (i.e., the more the uncertainty about the average number of messages he can miss). Even though these and other parameters about the shape of the distribution could be easily combined into a single security measure, in what follows we choose to discuss them separately for each strategy.

To facilitate the comparative analysis, Table 4 shows the mean, the stan-

Table 4: Representative statistics of the distribution of the number of consecutive messages an adversary can leave without being detected.

Strategy	**Mean**	**Std**	$\mathbf{Q_1}$	$\mathbf{Q_2}$	$\mathbf{Q_3}$	**H**
D-DS	362	217	239	338	448	3.32
D-R-DS	578	269	403	571	738	2.92
U-D-DS	497	268	341	483	629	3.19
U-D-R-DS	497	241	344	481	631	2.96
U-D-R	450	405	244	377	544	2.93

dard deviation, the quartiles Q_1, Q_2, and Q_3, and the entropy for the five distributions. In terms of low mean and high standard devition, the best two strategies are D-DS and U-D-R, respectively. In the D-DS case a tag is interrogated, on average, every 362 messages, with a standard deviation of 217. In the case of U-D-R, the average number of messages between consecutive interrogations increases to 450, but the uncertainty for the adversary is greater as the standard deviation also increases to 405. The remaining three strategies are slightly worse than these two in terms of time between consecutive interrogations, although their standard deviation is better than that of the D-DS strategy. In all cases, the situation for the adversary is considerably worse than in the baseline scenario, in which the mean is 999 messages with no deviation.

Fig. 4 shows the cumulative distribution functions for the probability distributions discussed above. From the adversary's point of view, these curves can be easily interpreted as the probability of being interrogated after a certain number of messages since the last time he participated in the protocol. The different security levels provided by each strategy are more noticeable here and reinforce the views discussed above.

6.4. Properties assessment

We next discuss whether the proposed PYP protocol meets the simultaneity and dependability properties that are required for all yoking proofs protocols. Each property is discussed below:

- Simultaneity. After a correct protocol execution, tags are sampled in the rounds and in the order determined by V. In this way, V (which is a trusted entity) ensures that all tags are scanned within a predefined time interval.

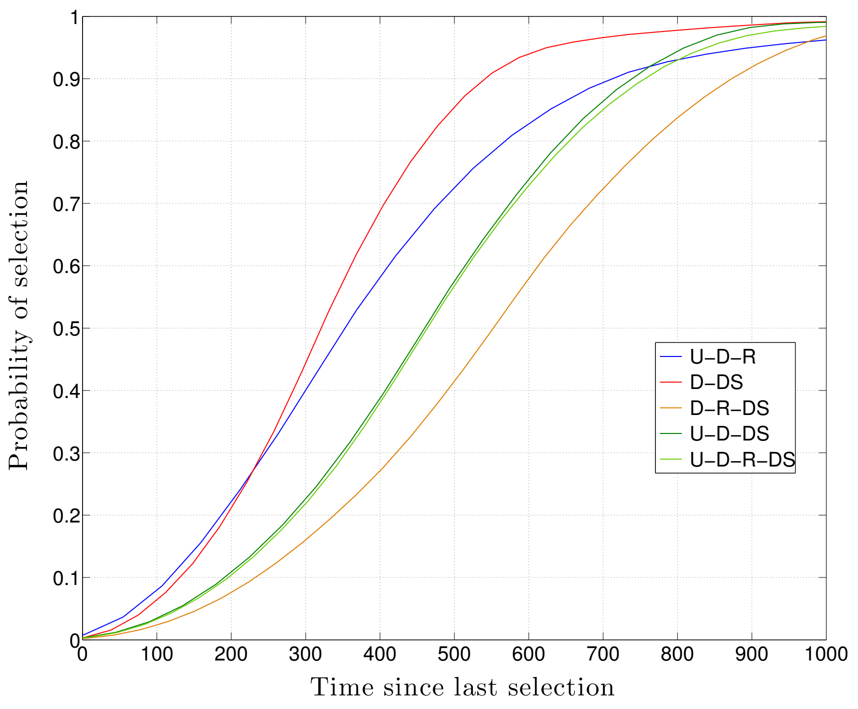

Figure 4: Probability of interrogating a tag as a function of the number of messages since the last time it was interrogated.

- Dependability (against false positives). Adv is unable to build a proof on behalf of an absent tag since all responses are encrypted using a key which is unknown to Adv.
- Dependability (against false negatives). Given that V receives the result of each round (PR), it is able to check whether all tags that should take part in the round did so. Thus, any false negative would be noticed by V.
- Privacy preservation. Given that only V knows the real identities ID_i of each tag, Adv is unable to know if a given tag is present or not at each round. It must be noted that this information is also unknown to R. Particularly, TS only contains the MAC addresses of the tags to be polled. Moreover, their answers are encrypted with K_i, so it is not able to reveal the response contents.

7. Conclusions

In many IoT scenarios it is necessary to construct an evidence that several objects have been scanned simultaneously by a reader. As introduced by Juels in 2004, classical yoking proofs constructions achieve this in a secure and efficient manner. However, such protocols were designed for applications where only a few tags (typically just two) are involved. As a result, issues such as the order in which each tag participates in the protocol are not relevant, nor it is the question of how many times a tag should be interrogated, as the proof completion time is often too little to worry about a tag leaving right after answering to the reader. This is not the case for many IoT applications where a potentially very large population of objects must be grouped together efficiently and guaranteeing that objects do not abandon the protocol undetected.

In this paper, we have introduced the notion of probabilistic yoking proofs (PYP) to address this issue. The proof itself is built as in classical yoking constructions. The key idea in a PYP consists of selecting at each round a subset of tags according to a Poisson sampling process where the sampling probability of each object varies over time. We have introduced various sampling mechanisms that attempt to balance security and efficiency, and have proposed different sampling strategies that combine them. Our experimental results suggest that some of these strategies give rise to PYP protocols with a very good trade-off among security, efficiency, and fairness.

Future work will be focused on two aspects. First, the proposal will be extended for more complex scenarios in which multiple readers could take part in the protocol. Second, a formal privacy assessment will be conducted over the protocol.

References

[1] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[2] L. Bolotnyy and G. Robins. Generalized "yoking-proofs" for a group of rfid tags. In *Third Annual International Conference on Mobile and Ubiquitous Systems: Networking Services*, pages 1–4, 2006.

[3] M. Burmester, B. Medeiros, and R. Motta. Provably secure grouping-proofs for rfid tags. In *Smart Card Research and Advanced Applications*, volume 5189 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg, 2008.

[4] H.-Y. Chien and S.-B. Liu. Tree-based rfid yoking proof. In *International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 1, pages 550–553, 2009.

[5] H.-Y. Chien, C.-C. Yang, T.-C. Wu, and C.-F. Lee. Two rfid-based solutions to enhance inpatient medication safety. *Journal of Medical Systems*, 35(3):369–375, 2011.

[6] R. Doss, S. Sundaresan, and W. Zhou. A practical quadratic residues based scheme for authentication and privacy in mobile rfid systems. *Ad Hoc Networks*, 11(1):383 – 396, 2013.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[8] J. Hermans and R. Peeters. Private yoking proofs: Attacks, models and new provable constructions. In *Radio Frequency Identification. Security and Privacy Issues*, volume 7739 of *Lecture Notes in Computer Science*, pages 96–108. Springer Berlin Heidelberg, 2013.

[9] ISO. Information technology – security techniques – entity authentication – part 2: Mechanisms using symmetric encipherment algorithms, iso/iec 9798-2:2008. International Standard, 2nd ed., 1999.

[10] A. Juels. "yoking-proofs" for rfid tags. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops,*, pages 138–143, 2004.

[11] Yi-Pin Liao and Chih-Ming Hsiao. A secure ecc-based rfid authentication scheme integrated with id-verifier transfer protocol. *Ad Hoc Networks*, 18(0):133 – 146, 2014.

[12] Y. Lien, X. Leng, K. Mayes, and J.-H. Chiu. Reading order independent grouping proof for rfid tags. In *IEEE International Conference on Intelligence and Security Informatics*, pages 128–136, 2008.

[13] C.-C. Lin, Y.-C. Lai, J. D. Tygar, C.-K. Yang, and C.-L. Chiang. Coexistence proof using chain of timestamps for multiple rfid tags. In *Advances in Web and Network Technologies, and Information Management*, volume 4537 of *Lecture Notes in Computer Science*, pages 634–643. Springer Berlin Heidelberg, 2007.

[14] Q. Lin and F. Zhang. Ecc-based grouping-proof rfid for inpatient medication safety. *Journal of Medical Systems*, 36(6):3527–3531, 2012.

[15] N.-W. Lo and K.-H. Yeh. Anonymous coexistence proofs for rfid tags. *Journal of Information Science and Engineering*, 26(4):1213–1230, 2010.

[16] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516, 2012.

[17] P. Peris-Lopez, J. C. Hernandez-Castro, J. E. Tapiador, and A. Ribagorda. Solving the simultaneous scanning problem anonymously: Clumping proofs for rfid tags. In *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, pages 55–60, 2007.

[18] P. Peris-Lopez, A. Orfila, J. C. Hernandez-Castro, and J. C. A. van der Lubbe. Flaws on rfid grouping-proofs. guidelines for future sound protocols. *Journal Network and Computer Applications*, 34(3):833–845, 2011.

[19] S. Piramuthu. On existence proofs for multiple rfid tags. In *ACS/IEEE International Conference on Pervasive Services*, pages 317–320, June 2006.

[20] B. R. Ray, J. Abawajy, and M. Chowdhury. Scalable rfid security framework and protocol supporting internet of things. *Computer Networks*, 67(0):89 – 103, 2014.

[21] Y. B. Saied, A. Olivereau, D. Zeghlache, and Laurent M. Lightweight collaborative key establishment scheme for the internet of things. *Computer Networks*, 64(0):273 – 295, 2014.

[22] J. Saito and K. Sakurai. Grouping proof for rfid tags. In *19th International Conference on Advanced Information Networking and Applications*, volume 2, pages 621–624, 2005.