Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 320–327

Procedia Computer Science

Technologies (ANT)

Pervasive Integrity Checking With Coupled Objects

Paul Couderc, Michel Banâtre, Fabien.Allard

INRIA Rennes, Campus Universitaire de Beaulieu, 35042 Rennes Cecex FRANCE

Abstract

Integrity checking is important in many activities, such as logistics, telecommunications or even usual tasks such as checking for someone missing in a group. While the computing and telecommunications worlds commonly use digital integrity checking, many activities from the real world do not benefit from automatic mechanisms for ensuring integrity. We propose the concept of coupled objects where groups of physical objects are tagged with RFID chips enabling pervasive and autonomous integrity checking.

Keywords:
Pervasive computing, security, integrity, RFID

1. Introduction

Checking for integrity of a set of objects is often needed in various activities, both in the real world and in the information society. The basic principle is to verify that a set of objects, parts, components or people remain the same along some activity or process, or remains consistent against a given property.

In the real world, it is a common step in logistic: objects to be transported are usually checked by the sender (for their conformance to the recipient expectation), and on delivery by the recipient. When a school gets a group of children to a museum, people responsible for the children will regularly check that no one is missing. Yet another common example is to check for our personal belongings when leaving a place, to avoid losts. While important, these verification are tedious, vulnerable to human errors, and often forgotten.

Because of these vulnerabilities, problems arise: E-commerce clients sometimes receive incomplete packages, valuable and important objects (notebook computers, passports etc.) get lost in airports, planes, trains, hotels, etc. sometimes causing tragic consequences.

While there are very few automatic solutions to improve the situation in the real world, integrity checking in the computing world is a basic and widely used mechanism: magnetic and optical storage devices, network communications are all using checksums and error checking codes to detect information corruption, to name a few.

The emergence of Ubiquitous computing and the rapid penetration of RFID devices enable similar integrity checking solutions to work for physical objects. In this paper, we present a general approach to pervasive integrity checking,

Email addresses: `paul.couderc@inria.fr` (Paul Couderc), `michel.banatre@inria.fr` (Michel Banâtre), `fabien.allard@senseyou.fr` (Fabien.Allard)

Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas. doi:10.1016/j.procs.2011.07.042

called *coupled objects*, and some of its possible applications. The paper is organized as follows: in the next section, we introduce the idea of *coupled objects*, and the motivations. Then we explain the concepts and the model of coupled objects. The fourth section discusses the implementation, some applications, and performance issues. Some related works are reviewed in fifth section. Finally, some perspectives are presented in conclusion.

2. Introducing coupled objects

2.1. An analogy with computer networks

Let's consider a typical E-commerce scenario: A client orders a set of items to an online store S. The items are shipped by a transport service. On delivery, the client will have to accept or reject the package after visually examining its conditions (package unsealed or broken, ...). Unfortunately, even when the package seems in good conditions, some items can still be missing inside. Moreover, it is impossible to determine the responsability for the problem:

1. The missing items could have not been shipped by the sender (the store)
2. The package could have been opened at one step in the transport, and some items stolen
3. The client could be of bad faith and pretend that some items are missing while it is not the case

Now consider what happens in a computer network: digital objects are fragmented into "packets", which can be transported independently of each other in the network. When they arrive at a destination point, packets are assembled together to rebuild the original object, which is checked for integrity. For this purpose, packets include additional information enabling error detection. Of course, networks are more complex than this simple view, with multiple encapsulation and fragmentation levels, but for the analogy with the logistic scenario the basic principle is sufficient.

We can consider a set of physical objects as "data" that are going to be transported and eventually separated at some occasions. At some point where the set of physical objects is assumed to be complete, integrity checks will take place. For instance, the transport service could check the integrity of the package before accepting it, and the client could do the same on delivery.

2.2. General principle of coupled objects

Essentially, coupled objects are a set of physical objects that defines a logical group. An important feature is that the group information is self contained within the objects themselves in such a way as that group properties checking (such as completeness) is possible just using the objects. In other words, the physical objects can be seen as fragments of a composite object. A trivial example could be a group made of a person, his jacket, his mobile phone, his passport and his cardholder.

More precisely, as an arbitrary object cannot describe by itself group information, we assume that a digital artifact including the group information is embedded on the object. A relevant implementation of this artifact is to use RFID technology, as we will see later: RFID tags with group information can be associated to each physical object member of the group.

Tagging physical objects with embedded chips is of course not new. But the originality in our approach is to avoid any central identification and to make the integrity checking self contained into the set of objects. Our goal was to propose a system that would support the following requirements :

1. ease of use and as low impact as possible on existing processes
2. scalability and reliability
3. low (or even no) dependance on information systems
4. privacy respect

We think it is important to stress the two latter requirements. Integration and interoperability issues can lead to death of emerging technologies or experimental systems: the cost of integrating something new into an operational infrastructure is very high, and dependence or impact on existing information systems should be as low as possible for a chance of acceptance.

Privacy concerns raise strong resistance to RFID technology [1]. As we will see, a core idea of coupled objects is to ensure standalone, anonymous operation and no dependence on databases or remote information systems.

3. Concepts and model of coupled objects

3.1. Some definitions and terminology

Considering a set of physical objects, or a set of parts of a physical object, we define a *coupled object* as a logical association of these objects (or parts), self contained within the objects.

These objects are called *fragments*: a fragment is simply a physical object augmented with an electronic artifact that provides a digital memory for the information required to describe the association of the fragment with the other fragments of the coupled object. This electronic artifact is called *tag*, in reference to the typical implementation using RFID tags.

Association or *coupling* is the process of writing the tag memories of all the fragments of a coupled objects with the appropriate information to make a coupled object: each fragment f_i is labeled with a piece of data g_i such that the set g_i is sufficient to describe the belonging of the fragment to this particular coupled object.

Checking is the process of reading the data $g_1, ..., g_n$ from the tags of a set of fragments $f_1, ..., f_k$ and determining whether $g_1, ..., g_n$ is a valid coupled object.

3.2. Data structure

An essential property of a coupled object is that the logical association between the fragments should be self contained within the fragments themselves, and the data in the tags should be self sufficient to describe the association.

A trivial solution would be to associate a unique identifier to each fragment, and describe the coupled object as the set of identifiers of all its fragments. However, this is not realistic given existing technological constraints: the memory size of the tag for each fragment would have to grow linearly with the number of fragments. The same is true for the quantity of data to read when checking a coupled object. Using for example 96 bits identifiers for the fragments, a ten fragments coupled objects would require 960 bits, which is beyond the capacity of the 512 bit memory tags in common use. Larger memory tags do exist, but this approach is not scalable to a large number of fragments. Moreover, the explicit enumeration of global identifiers in each fragment raises a threat for privacy, as it eases the tracking of group of objects and directly expose the association of group of objects.

Another solution would be to associate a unique identifier to each coupled object. But this would assume the existence of a global authority and service to allocate the identifiers. Once again, threats on privacy and confidentiality of coupled objects would be strong issues.

An autonomous and distributed alternative of the latter approach is to use a hashing or signature functions to generate quasi-unique fingerprint information describing a coupled object. We assume that each individual fragment has a specific information which is associated to it, which we will call ID_i. The information used to define the group is a function computed from the set of ID_i, typically a hash function: $H(ID_1, ...ID_2, ...ID_n)$. Using a hash function has several advantages :

1. group representation is compact
2. integrity check can be performed efficiently
3. group representation is meaningless and does not allow to deduce any relevant information

For a set of fragments f_i with respective identifiers id_i the association information stored on the fragments would be $g_i = H(id_1, ..., id_n)$, or simply g as g_i is the same for all the fragments of the same coupled object. The tag memory of each fragment include both the individual identifier id_i and the coupled object fingerprint g

3.3. Proxy fragment

In some cases, we would like to make a coupled object without having all the fragments physically together (meaning, in close proximity). This can be interesting for example in a logistic application if we want to create a coupled object including the sender and the recipient as fragments: the sender would like to ensure that his package would only be delivered to the right recipient, while the latter would like to only accept a shipment from the right sender. As the sender and the recipient would typically not be present together at shipping and delivery time, it seems not possible to create and check a coupled object in these situations.

For these cases, we introduce the notion of *proxy fragment*. If we have a physical fragment f_k with a tag containing the coupled object data g, a proxy fragment f'_k is simply another physical object with the same data g that is used elsewhere in place of f_k, where the latter cannot be physically present.

Actually, the proxy fragment can be virtual, in the sense that g has to be known at the checking time to validate the coupled object, but g is not necessarily on a tag: it can be provided to the checking process through another way: for example, it could be sent through a secured channel to the checking terminal.

A typical application would be the sender-recipient scenario: the sender S creates a coupled object made of two DVDs and a fragment representing the recipient. As the latter is not physically there for the coupling, a proxy fragment is used, such as a client number or a public key. Let's call R this identifier. The sender then creates the coupled object by generating a fingerprint $Q = H(id_1, id_2, R)$ where id_1 and id_2 are the identifiers for the two DVDs. This fingerprint is stored in all the physical fragments of the package, in this case on the tag attached to each.

Symmetrically at delivery, before opening the package, the recipient checks for the integrity of the coupled object: he makes his proxy fragment R available for the checking process, and the tags inside the package are read:

1. on each tag i, the individual identifier id_i
2. the coupled object fingerprint Q stored on all tags

$X = H(id_1, ..., id_n, R)$ is computed and checked against Q. If $X = Q$, the coupled object is considered valid and the package can be accepted by the recipient. Otherwise, either the recipient is not the right one or the set of items inside the package do not match the set of objects seen at coupling time.

3.4. *Coupled objects hierarchies*

The set of physical objects composing a coupled object could itself be used as a fragment in another, upper level, coupled object. This recursive process leads to hierarchies of coupled objects. A simple example is a first coupled object O_1 made of some parts of a bike: the wheels, the frame and the seatpost making a total of four fragments. Then, at an upper level we would like to create a coupled object O_2 made of two fragments: the bike and his owner.

Previously, we were using a flat addressing scheme with all the fragments at the same level. Coupled object trees require a hierarchical data structure to allow the independent validation of each tree or sub-tree. To this end, we have to introduce a secondary fingerprint in the tag memory: the primary fingerprint corresponds to the coupled object of the first level to which the fragment belongs to, while the secondary fingerprint corresponds to the coupled object of upper level to which the fragment belongs to (see figure 1).

4. Implementation

We designed and implemented several integrity checking systems based on the coupled objects model. These systems are using RFID UHF tags and readers complying with the EPC Global standard [2] (EPC gen2) . The primary applications that we were interested for coupled objects involved reliable checking of several moving tags at once. UHF RFID class1 gen2 tags were the most readily available solution featuring anti collision and providing acceptable performances in our context. This section discusses the implementation issues for the data structure mapping, the coupling and the checking processes.

4.1. *Memory allocation*

Beside the usual EPC 96-bit identifier that is used as the fragment identifier, the tags have a 512 bits writable user memory. This memory is used to store the fingerprints for the coupled objects. We were using `sha-256` hash function to generate the fingerprints, so given our user memory capacity coupled object hierarchy can be supported (two fingerprints are needed, as explained in section 3.4). See [3] for good discussion of hash functions in the context of RFID.

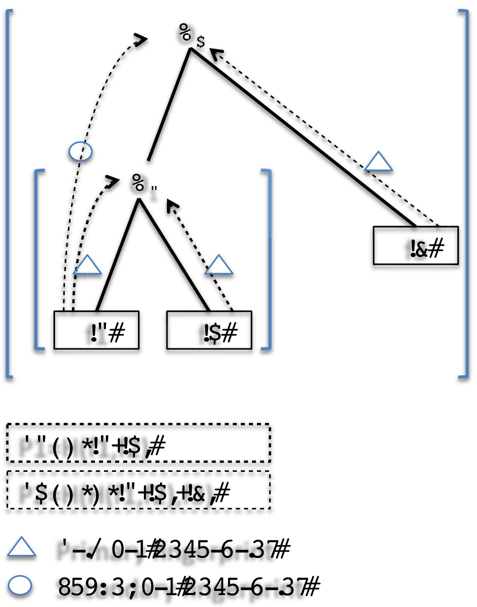

Figure 1: Mapping the tree structure on the fragments

4.2. *Coupling and checking*

The coupling process consists in the following steps: first, all the objects that are to be the fragments of a coupled objects are equipped with a tag, pre-programmed with a unique identifier (stored in the EPC identifier). Then all the fragments are placed in the range of a reader. The coupling software enumerates the tags using the appropriate "read inventory" inquiry, getting the set of identifiers. The fingerprint $P = H(id_1, ..., id_n)$ is computed, and then written in the user memory of each tag.

The checking process is similar: the coupled object to be checked is placed in the range of the checking reader. Then coupled object fingerprint P is read from the tag memories, along with the identifiers of the fragments from the EPC identifier. Of course, if the fingerprint stored for a given fragment is different from the fingerprint stored on the others, the checking fails at this point. Otherwise, $Q = H(i_1, ..., i_n)$ is computed and compared against P (stored on each fragment). The coupled object is valid when $P = Q$.

4.3. *Applications*

Several systems based on this protocol were implemented, for different purposes. A first one consists of RFID-controlled gates to warn people when some of their belongings are missing, such as when leaving a train, a plane or a hotel [4]. This assumes that a coupled object is created from the valuable objects of the user and himself. A strong point highlighted by this application is that the identifiers of the personal objects can be meaningless and often changed; the system can operate without compromising privacy.

A similar application is being experimented for airport workers that have to operate inside terminals, where tools are normally forbidden: special toolset equipped with tags configured as coupled objects are provided to the workers, which have to cross a checking gate when entering or leaving sensitive areas. Leaving a tool inside the sensitive area is detected when leaving as the toolset is not valid.

Another system demonstrates the application of coupled objects in logistics, in particular for E-commerce: it allows to check for the integrity of a package from its actual content, provided that the content is a coupled object, and the identities of the sender and the addressee. The standalone operation provided by the architecture avoid interoperability issues between the information systems of the clients and the logistic operator.

Finally, an experimental system is being deployed for bicycle security: some parts of a bicycle and the owner of the bicycle make a coupled object, and a protected area uses integrity checking to operate the opening of an automatic door. An advantage of this solution is that individual bikes do not need heavy and cumbersome physical protection: the protection is shared by the bikes inside the protected area.

4.4. *Performance and reliability*

Integrity checks requires to capture the data stored on all the tags that make a coupled object. Usually, two main approaches are possible: reading the tags sequentially, or using the “inventory read” primitive that readers provide to collect all the tags data at once. We called the first one *soft* as the read process is software controlled and the second one *hard* as it relies on the reader. The figure 2 shows the failure rate that we experienced with a Motorola XR480 reader.

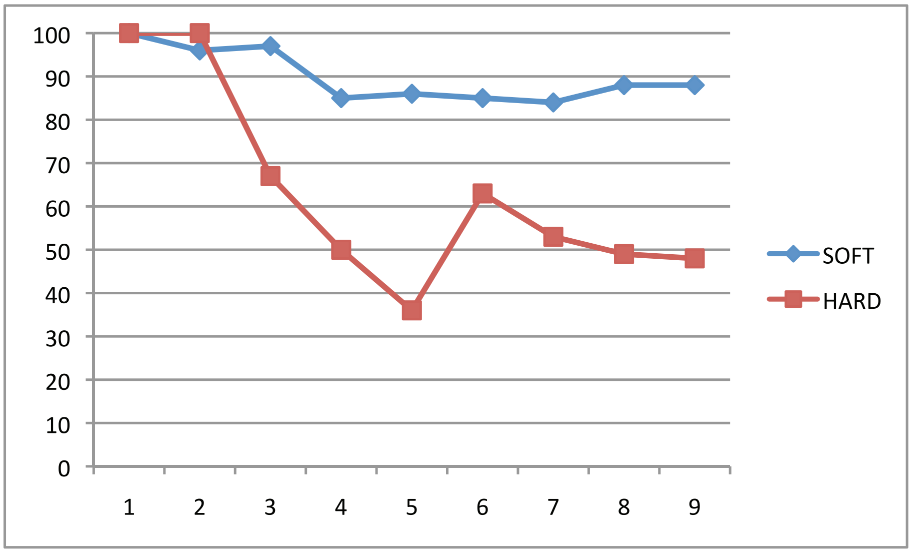

Figure 2: Percentage of successful read vs. number of tags

As we can see, the *hard* method using “inventory read” can miss a significant proportion of tags. The *soft* method is more reliable: as the number of tags increases, more than 80% of tags are still captured.

However, the capture time is constant for the *hard* method, while it increases linearly with the number of tags with the *soft* method. In order to read all the tags, we have to repeat the read process several times. We determined experimentally the time required for both method to capture a given set of tags, as shown on figure 3.

We can see that the *hard* method is more efficient, keeping below 2s reading up to ten simultaneous tags.

Tags can still be missed, and even with a low probability the occurrence of this problem, inherent to the RFID technology, has to be examined. Actually, in many application the coupled object architecture can make the read process *fail safe*, meaning that a missed tag will result in a false warning (seeing the coupled object as invalid), providing an opportunity to check again. The case where all the tags would be missed leading to a silent failure is very unlikely.

5. Related works

RFID is a hot topic with many issues given its broad application domain and emerging success in security, accountability, tracking, etc. However, coupled-objects principle differs from many RFID systems where the concept of identification is central and related to database supported information systems. In some works, the tag memories are used to store *semantic* information, such as annotation, keywords, properties [5, 6]. Our approach is in the line of this idea: RFID are used to store in a distributed way group information over a set of physical artifacts. The concept using distributed RFID infrastructure as pervasive memory storage is due to Bohn and Mattern [7].

Maintaining group membership information in order to cooperate with “friend devices” is a basic mechanism (known as *pairing* or *association*) in personal area networks (PAN) such as Bluetooth or Zigbee. Some personal

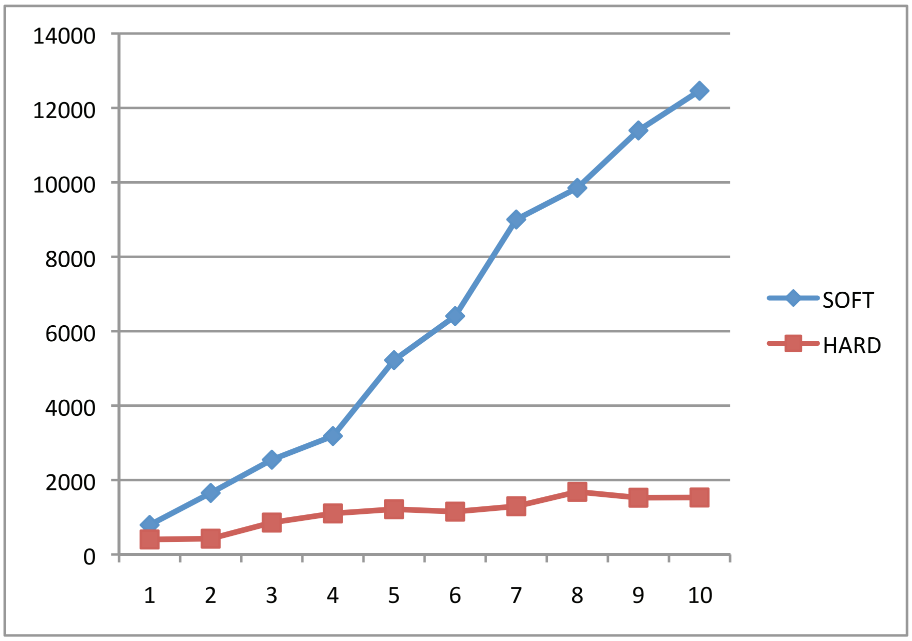

Figure 3: Time required (in ms) to read all the tags vs. number of tags

security systems based on PAN for luggages were proposed [8], which enable the owner to monitor some of his belongings, such as his briefcase, and trigger an alarm when the object is out of range. A major drawback of active monitoring is the energy power which is required, as well as potential conflicts with radio regulations that can exist in some places, namely in airplanes.

Still in the context of Bluetooth, RFID has also been used to store PAN addresses in order to improve discovery and connexions establishment time [9]. It can be seen as storing "links" between physical objects, such as in coupled objects, but without the idea of a fragmented group. Yet another variant is *FamilyNet* [10], where RFID tags are used to provide intuitive network integration of appliances. Here, there is a notion of group membership, but it resides on information servers instead of being self-contained in the set of tags as in Ubi-Check. Probably the closest concept to Ubi-Check is *SmartBox* [11], where abstractions are proposed to determine common high level properties (such as completeness) of groups of physical artifacts using RFID infrastructures.

6. Conclusion

We presented the principles of coupled objects, enabling the design of pervasive integrity checking solution for many applications. The strong points of this solution are its independence from any remote information system support or network support, and user's privacy respect as it is anonymous and does not rely on global identifiers.

In some applications where many tags have to be read at once on mobile objects, the performance of current hardware with respect to inventory read reliability and speed can be an issue. Another issue is security: the tags used in any RFID security solution should resist to tag cloning attacks or tag destruction attempt. This typically involve tag level cryptography logic, which require more execution cycles, more power, and more time to be read. These solutions are orthogonal to the coupled objects principles discussed here, which could make use of secured tags. This topic could lead to further research: as a coupled object is supported by a collection of tags, some of the costly cryptography mechanisms may be distributed over the set of tags, but such approaches would go outside current tags.

Other perspectives are other application scenarios; we are in particular examining green-IT solutions for waste recycling using coupled objects to ensure the quality of the returned materials. Finally, we are also examining the mapping of other complex data structures (beside trees) on a set of memory-limited tags considered as fragments.

References

[1] P. Peris-Lopez, J. C. H. Castro, J. M. Estévez-Tapiador, A. Ribagorda, RFID systems: A survey on security threats and proposed solutions, in: PWC, 2006, pp. 159–170.

[2] i. EPC GLobal, Class 1 generation 2 uhf air interface protocol standard "gen2" v.1.2.0 (2010).
URL `http://www.epcglobalinc.org/standards/uhfc1g2`

[3] M. Feldhofer, C. Rechberger, A case against currently used hash functions in RFID protocols, 2006, pp. 372–381.

[4] M. Banâtre, F. Allard, P. Couderc, Ubi-check: A pervasive integrity checking system, in: NEW2AN '09 and ruSMART '09: Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces, 2009, pp. 89–96.

[5] M. Banâtre, M. Becus, P. Couderc, Ubi-board: A smart information diffusion system, in: NEW2AN '08 / ruSMART '08: Proceedings of the 8th international conference, NEW2AN and 1st Russian Conference on Smart Spaces, ruSMART on Next Generation Teletraffic and Wired/Wireless Advanced Networking, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 318–329.

[6] T. D. Noia, E. D. Sciascio, F. M. Donini, M. Ruta, F. Scioscia, E. Tinelli, Semantic-based bluetooth-RFID interaction for advanced resource discovery in pervasive contexts, Int. J. Semantic Web Inf. Syst. 4 (1) (2008) 50–74.

[7] J. Bohn, F. Mattern, Super-distributed RFID tag infrastructures, in: Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI 2004), no. 3295 in Lecture Notes in Computer Science (LNCS), Springer-Verlag, Eindhoven, The Netherlands, 2004, pp. 1–12.

[8] R. Kraemer, The bluetooth briefcase: Intelligent luggage for increased security (2004).
URL `http://www-rnks.informatik.tu-cottbus.de/content/unrestricted/teachings/2004/`

[9] T. Salminen, S. Hosio, J. Riekki, Enhancing bluetooth connectivity with RFID, in: PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society, Washington, DC, USA, 2006, pp. 36–41.

[10] W. Mackay, M. Beaudouin-Lafon, Familynet: A tangible interface for managing intimate social networks, in: Proceedings of SOUPS'05, Symposium On Usable Privacy and Security, ACM, 2005.

[11] C. Floerkemeier, M. Lampe, T. Schoch, The smart box concept for ubiquitous computing environments, in: Proceedings of sOc'2003 (Smart Objects Conference), Grenoble, 2003, pp. 118–121.