Formal Verification of a Key Establishment Protocol for EPC Gen2 RFID Systems: Work in Progress

Wiem Tounsi, Nora Cuppens-Boulahia, Frédéric Cuppens, Joaquin Garcia-Alfaro

Institut Télécom, Télécom Bretagne,
CS 17607, 35576 Cesson-Sévigné, France
forename.surname@telecom-bretagne.eu,
joaquin.garcia-alfaro@acm.org

Abstract. The EPC Class-1 Generation-2 (Gen2 for short) is a standard Radio Frequency Identification (RFID) technology that has gained a prominent place on the retail industry. The Gen2 standard lacks, however, of verifiable security functionalities. Eavesdropping attacks can, for instance, affect the security of monitoring applications based on the Gen2 technology. We are working on a key establishment protocol that aims at addressing this problem. The protocol is applied at both the initial identification phase and those remainder operations that may require security, such as password protected operations. We specify the protocol using the High Level Protocol Specification Language (HLPSL). Then, we verify the secrecy property of the protocol using the AVISPA model checker tool. The results that we report show that the current version of the protocol guarantees sensitive data secrecy under the presence of a passive adversary.

1 Introduction

The RFID technology is one of the most promising advances in current pervasive infrastructures. It allows contactless identification of tagged objects and people. The EPC Gen2 standard, short-hand for the Electronic Product Code (EPC) Class-1 Generation-2 [8], is a proper example of this technology. Most promising Gen2 scenarios relay on passive tags designed with very basic capabilities. Gen2 tags derive their transmission and computational power from the signal of an interrogating device. Albeit they can perform some basic arithmetic operations, they are characterized with a very minimalist design, specially with regard to the implementation of security functionalities. For this reason, the security of the EPC Gen2 technology is gaining great attention in both industry and academia [9],[10]. One important focus of research is the inclusion of verifiable secure key establishment protocols, to prevent eavesdropping attacks [12].

In this regard, we present in this paper a work-in-progress protocol that aims at addressing the aforementioned objective. We present the protocol and describe its translation into High Level Protocol Specification Language (HLPSL), a specification language for formalizing protocols and security goals based on Lamport's Temporal Logic of Actions (TLA) [13]). Finally, we evaluate the translated description with the AVISPA model checker tool, a well known EU funded software verification framework for the automatic validation of security protocols [1]. The initial results that we report show that the protocol guarantees secrecy in the presence of a passive adversary.

Paper Organization. Section 2 motivates our work. Section 3 describes the protocol. Section 4 presents some necessary assumptions. Section 5 overviews the AVISPA tool and the HLPSL format. Section 6 models the protocol in HLPSL and summarizes the results of the automatic verification process. Section 7 concludes the paper.

2 Motivation

Flawed Gen2 Security Model. The Gen2 specification considers some special operations that require reader authentication, such as tag memory writing, decommission of tags, and tag self-destruction. They require the communication of a password prior the tag execution. Such a password must be sent via the insecure reader-to-tag channel. Since this channel is more likely to suffer from eavesdropping attacks than the tag-to-reader channel, the specification proposes to protect the exchange as follows:

1. READER $\longrightarrow$ TAG : Key-request
2. TAG $\longrightarrow$ READER : Key
3. READER $\longrightarrow$ TAG : Password $\oplus$ Key

The reader informs in Step 1 that it is waiting for a key necessary to obscure the following exchange (cf. Step 3) that will eventually contain the required password to grant the execution of the operation. The key can also be used in order to obscure the contents of the remainder parts of the operation (e.g., to protect the data associated to a memory writing operation). The key is generated by the tag as a random bit string, and transmitted in Step 2 in plaintext to the reader. This is done via the tag-to-reader channel which, in principle, is expected to have an eavesdropping range much lower that the reader-to-tag channel. This exchange supposes that an adversary eavesdropping the reader-to-tag channel cannot capture the sensitive data (either the password or the contents of the password-protected operation). However, it is straightforward that an adversary capable of eavesdropping the tag-to-reader channel using special hardware devices (e.g., readers with high sensitive receivers and multiple antennas), or predicting the output of the random bit generator of the tag (e.g., based on a flawed EPC Gen2 pseudorandom generator), can simply obtain such a sensitive data by applying the obtained key and an Exclusive-OR operation.

Lack of Formally Verified Security Enhancements. There have recently been many approaches focusing on enhancing the security of the EPC Gen2 technology [10]. Some of these approaches propose physical solutions, including new cryptographic primitives on-board of the tags; others propose straighforward protocols, that remain to be adapted to the Gen2 contsraints. However, a great number of solutions have been reported as insecure. For instance, recent cases of authentication techniques for EPC Gen2 were reported vulnerable by Li and Wang in [15]; by Li and Deng in [14]; and by Cao, Bertino, and Lei in [6]. These cases show the lack of formal verification of new security techniques for EPC Gen2, which we consider deemed necessary.

3 Proposed Key Establishment Protocol

The protocol aims at establishing a secure communication between a reader and a tag without any advance exchange of a secret key. This assumes the existence of a shared generation function denoted by KeyGen. The KeyGen function generates for each exchanged message a secret key named *derived key* (e.g., K_{Der-i}) relying on some initialization values (e.g., a master key K_{Master}, a timestamp t). The validity of the derived keys depends on the validity of the master key. This validity is decided by a third entity (not described in this work) through a computational approach after a fixed number of use. In the sequel, we present the three main stages of the protocol.

1. Reader Authentication. The reader must first prove its identity to the tag. We assume that preserving on-board tag generated data is the aim of our protocol. Thus, we consider that readers are likely to be dishonest rather than tags. Moreover, the constrained capacity of the tag (mainly, the lack of energy) to follow all the communication process or to disturb it, weaken the possibility for a tag to play the role of an adversary. The reader authentication steps are described below.

1. READER $\longrightarrow$ TAG : ReadID
2. TAG $\longrightarrow$ READER : TagID
3. READER $\longrightarrow$ TAG : TagID $\oplus$ K_{Der0}
4. TAG $\longrightarrow$ READER : K_{Der1}

The reader requests the identification of the tag (denoted by the ReadID command) in Step 1. As a result, the reader receives in Step 2 the tag identification (denoted as TagID). The reader verifies the existence of this identification in its related database and extracts the last *state* of the identified tag (e.g., master key, timestamps, internal vectors). This *state* allows to derive a new derived key named K_{Der0}, based on the master key K_{Master}. The result of this generation is XORed with the received TagID and sent to the tag in Step 3. Upon receiving the message TagID $\oplus$ K_{Der0}, the tag generates a new key from the shared KeyGen function and checks the equality of this new generated key and the received value. If they are equal, the tag generates a new derived key with the same function and sends it as an acknowledgement to the reader in Step 4.

2. Master Key Assignment. The reader distributes now a new master key to refresh the protocol state on the two sides.

5. READER $\longrightarrow$ TAG : ($K_{new-Master}$,t) $\oplus$ K_{Der2}
6. TAG $\longrightarrow$ READER : $K_{new-Der0}$

The reader sends in Step 5 a new calculated master key concatenated with a timestamp t, XORed together with K_{Der2} which is generated using the initialized master key. The tag, in turn, calculates a new K'_{Der2} and checks the correctness of the sent K_{Der2} by applying an XOR operation as follow: ($K_{new-Master}$,t) $\oplus$ K_{Der2} $\oplus$ K'_{Der2}. If K_{Der2} = K'_{Der2}, the tag obtains ($K_{new-Master}$,t) using the nilpotency property (cf., Section 4), and recognizes the shared timestamp t to eventually deduce the $K_{new-Master}$. Finally, the tag updates its variables and acknowledges in Step 6 a new derived key using the KeyGen function initialized by the new received master key.

3. Remainder Gen2 Operations. The system can now continue with other Gen2 operations, such as reading or writing data from/to the tag. For instance, the following steps exemplify the protocol part associated to a *Write operation*:

7. READER $\longrightarrow$ TAG : $\mathtt{op}_{\mathtt{request}} \oplus \mathtt{K}_{\mathtt{new\text{-}Der1}}$
8. TAG $\longrightarrow$ READER : $\mathtt{K}_{\mathtt{new\text{-}Der2}}$
9. READER $\longrightarrow$ TAG : (data,t) $\oplus$ $\mathtt{K}_{\mathtt{new\text{-}Der3}}$
10. TAG $\longrightarrow$ READER : $\mathtt{K}_{\mathtt{new\text{-}Der4}} \oplus \mathtt{op}_{\mathtt{reply}}$

The reader sends in Step 7 a request command named $\mathtt{op}_{\mathtt{request}}$ XORed with a new derived key. $\mathtt{op}_{\mathtt{request}}$ is a command to have the permission for writing. The tag checks the authentication of the reader by verifying $\mathtt{K}_{\mathtt{new\text{-}Der1}}$ and acknowledges the request by sending in Step 8 a new derived key generated using the KeyGen function. The reader sends in Step 9 the data to be written. Upon accepting the data, the tag replies in Step 10 with a new derived key XORed with a predefined command $\mathtt{op}_{\mathtt{reply}}$ to acknowledge the operation.

4 Assumptions prior the Verification Process

We want to decide whether the protocol ensures the secrecy of some sensitive terms (e.g, the master key that can reveal the sequence of the derived keys) under a number of assumptions, namely the encryption model and the adversary capabilities. By ensuring secrecy, we expect that the adversary cannot deduce from his initial knowledge of the protocol and the environment of its execution and from the set of messages sent in a given execution more than what he is permitted to know.

Encryption Model. The hypothesis of perfect encryption inspired by the Dolev-Yao model [7], has been long assumed in the modeling of cryptographic protocols. This hypothesis idealizes the cryptography used in the functions. It allows to overcome the complexity of these functions by embedding the algebraic properties of the cryptographic primitives into black boxes (i.e., the only way to decrypt a message is to know the encryption key). This assumption has allowed to find numerous logical flaws in protocols (e.g., in the Shamir 3-pass protocol [18]).

We relax the perfect encryption hypothesis by exposing the cryptographic primitives used in our protocol and analyzing them according to their algebraic properties. We explicitly define an encryption method based on a one time pad using the XOR operator (e.g., a message m encrypted with a key k is denoted as $m \oplus k$). The XOR operator is known to have four properties that constructs a deduction rules for the adversary:

1. $x \oplus y = y \oplus x$ (Commutativity)
2. $x \oplus (y \oplus z) = (x \oplus y) \oplus z$ (Associativity)
3. $x \oplus 0 = x$ (Neutral element)
4. $x \oplus x = 0$ (Nilpotency)

Adversary Capabilities. We consider a passive adversary. The introduction of the properties of the XOR operator that weaken the assumption of a perfect encryption extends the capabilities of the adversary and strengthen his deduction possibilities. The adversary has an entire control of the communication channels. It can harvest his knowledge

by using, at first, the information he initially knows about the participants, the network characteristics and the algebraic properties of the protocol (e.g., the standards used in the communication or the key generation algorithm) and, at second, by eavesdropping the different messages sent in the network to use them as input of his deduction process.

5 The AVISPA Model Checking Tool

5.1 A Brief Presentation

AVISPA [1] is a suite of applications commonly used for automated validation and verification of cryptographic protocols. It maintains a library of security protocol specifications written in the HLPSL language (e.g., 54 IETF protocols are tested). For each protocol, the expected security properties and the possible attacks found are described. The AVISPA framework is composed of several modules. A translator called HLPSL2IF for transforming HLPSL specifications to a low level specification with IF language (Intermediate Format) and four different verification backends to analyze the IF specifications. These backends are named: On the Fly Model Checker(OFMC) [3], Constraint-Logic based Attack Searcher (CL-AtSe) [21], SAT based Model-Checker (SAT-MC) [2] and Tree Automata based Protocol Analyser (TA4SP) [4].

Each backend has its own options and parameters to define before the verification. After the verification process, the output describes the result, and under what conditions it has been obtained. The output format is common to all backends of the AVISPA tool. In the `SUMMARY` section; it indicates if the protocol is safe, unsafe, or if the analysis is inconclusive. In a second section titled `DETAILS`, the tool explains under what conditions/reasons the protocol is declared safe/unsafe/inconclusive. The next sections, `PROTOCOL`, `GOAL` and `BACKEND` recall the name of the protocol, the goal of the analysis and the name of the back-end used, respectively. Finally, some possible comments and statistics of the execution are described and the trace of the attack (if any) is printed in an Alice&Bob notation which means that a given goal has been violated.

5.2 The HLPSL Format

The protocol and the verification assumptions are specified in the High Level Protocol Specification Language (HLPSL) [5]. HLPSL is a specification language for formalizing protocols and security goals based on Lamport's Temporal Logic of Actions (TLA) [13]). The language, developed in the context of the AVISPA framework [1], is a role-based language focusing on roles rather than on messages exchange. Roles can be `basic` (cf., agent roles) describing the action of an agent during the execution of the protocol or `composed` (cf., session and environment roles) describing scenarios of basic roles to model an entire protocol run including the adversary model.

Basic Roles. Figure 1(a) shows how the basic role is generally structured. Each basic role declares its name (designed by `A`), its initial information or parameters (denoted by `param`) and the agent playing the role (denoted by `ag`). The basic role can declare a set of local variables (denoted by `L`). The `init` section assigns the initial values to the local variables, if required.

(a)
```
role A(param) played by ag
def= local L
init Init
transition
evt.0 ^ evt.1 = | > act.1
evt.2 = | > act.2
...
evt.n = | > act.n
end role
```

(b)
```
role S (param)
def= const C
composition
R1 (param-1) ^ . . . ^ Rn
(param-n)
end role
```

(c)
```
role environment (param)
def= const C
intruder_knowledge = IN
composition
S1 (param-1) ^ . . . ^ Sn
(param-n)
end role
```

(d)
```
goal
secrecy_of sec_kn
end goal
```

Fig. 1. HLPSL main elements. (a) Basic role structure. (b) Session role structure. (c) Environment role structure. (d) Secrecy in the goal section.

The `transition` section describes changes of the agent state. It consists of a trigger (e.g., `evt.2`) and an action (e.g., `act.2`) to be performed when the trigger event occurs. The `=|>` symbol separates the two phases.

Composed Roles. Composed roles combine basic roles, either in parallel or in sequence. HLPSL defines two composed roles: the session role and the environment role. Actions, in composed roles, are not defined in a `transition` section like in basic roles. Rather, a `composition` section is defined to instantiate other roles `Ri` or `Si` (with sets of parameters `param-i`) that run in parallel (cf., Figures 1(b) and 1(c)).

The session role (named `S`), instantiates in its composition section the basic roles and the different channels relating them while the environment role instantiates in its composition section all the sessions to be run (referred by `Si`). The environment role is called the main role, as it declares the global constants (denoted by `C`) and defines the intruder knowledge (i.e., the information that the adversary knows and eavesdrops, denoted by `IN`).

Security Properties. HLPSL provides an independent section to declare the security properties required, named `goal`. The goal declaration can be done either by using predefined macros of the security properties (secrecy, weak authentication, strong authentication) or by using Linear Temporal Logic formulas [13]. We are interested in the secrecy property (cf. reference [5] for its complete definition).

The property is added to the honest basic role and identified by `protocol_id` type. It is declared later in the goal section. For example, assuming that `sec_kn` is the name of the secret term, the `secret(Knew',sec_kn,A,B)` expression is added in the honest player of the role claiming that `Knew'` must be a secret term and that it is known only by the two agents `A` and `B`. The constant `sec_kn` is declared later in the `goal` section as seen in the Figure 1(d).

6 Automatic Verification of the Protocol

Modeling the Protocol and the Assumptions in HLPSL. The specification of both the protocol (cf., Section 3) and the verification assumptions (cf., Section 4) is described into five HLPSL sections: two sections for the basic roles (the tag and the reader) and two sections for the composed roles (the session and the environment roles). A special section is dedicated to the secrecy goal. Due to space restrictions, we only show in Figure 2 one basic role (the RFID role) beginning by Step 3, one composed role (the environment role) and the goal section[1].

We assume that the reader and the tag execute some key generation function, initialized by the master key (K or Knew) and an internal state (Instate), to generate the derived keys. The output value of this function is used as an input of the next execution, defining a succession of internal states and of derived keys. For that aim, we use an HLPSL function of type hash_fun to specify the generation function called Keygen. This function is supposed to be known to the intruder. Thus, the intruder is able to calculate the successors of Keygen(Li,Instate), called also the derived keys, and can

[1] Full HLPSL code available at: `http://j.mp/FPS2011`

```
RFID role
role tag (B,A : agent,
Keygen : hash_func,
Tagid,Opreq,Tmp : text,
Snd,Rcv : channel (dy))
played_by B def=


local
State : nat,
L,L1,L2,L3,L4,L5,L6,L7 : message,
Instate: text,
Opwr   : text,
Kold   : text,
Knew   : text

init
State:=0

transition

1. State=0
^Rcv(xor(Tagid,L'))  =|>
State':=1
^Instate' :=new()
^L1':= Keygen(L',Instate')
^Snd(L1')

2. State=1
^Rcv(xor((Knew'.Tmp),L2')) =|>
^State':=2
^Instate' :=new()
^L3':= Keygen(Knew,Instate')
^Snd(L3')

3. State=2
^Rcv(xor((Opreq.Tmp),L4')) =|>
State' :=3
^Instate' :=new()
^L5':= Keygen(L4',Instate')
^Snd(L5')

4. State=3
^Rcv(xor((Opwr'.Tmp),L6')) =|>
State' :=4
^Instate' :=new()
^L7':= Keygen(L6',Instate')
^Snd(L7')

Environment role
role environment() def= local
Snd, Rcv: channel(dy)
const
sec_kn: protocol_id,
a, b: agent,
keygen: hash_func,
tagid,opreq,tmp  : text,
l,l1,l2,l3,l4,l5,l6,l7: message

intruder_knowledge = {a,b,tagid,
opreq,tmp,keygen,l,l1,l2,l3,l4,
l5,l6,l7}

composition
session(a,b,keygen,tagid,opreq,tmp)

goal
secrecy_of sec_kn, sec_opwr
end goal
```

Fig. 2. Specification sketch of the protocol in HLPSL.

(a)

```
% OFMC
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  FPS2011Protocol.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 2.18s
  visitedNodes: 1715 nodes
  depth: 8 plies
```

(b)

```
%CL-AtSe
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  FPS2011Protocol.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 3 states
  Reachable  : 3 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

Fig. 3. Evaluation results. (a) OFMC results. (b) CL-AtSe results.

invert the outputs of the Keygen function only if he knows/deduces all the entries of Keygen (e.g., the updated internal state (Instate)). In the environment role, as channels in AVISPA are until now of only type (dy) relying on the Dolev-Yao intruder model [7], we define a passive adversary by adding to his knowledge all the data transmitted in the network in addition to the knowledge of the key generation function and the standards used in the communication, without attributing him an active role to play in the composition section. Finally, in the goal section, the Master key and the data written on the tag are specified as (sec_kn) and (sec_opwr) respectively, to be checked for secrecy requirements.

Obtained Results. We have used the OFMC and CL-AtSe backends of the AVISPA framework. Both the OFMC and CL-AtSe backends support analyzing protocols with Exclusive-OR properties. They do it for a bounded number of sessions. The backends are called with the default options.

Results have reported the protocol as safe (cf. Figure 3), meaning that the stated security goals (cf. Section 6) are successfully checked by the OFMC and CL-AtSe backends for a bounded number of sessions. Therefore, we can affirm that our protocol satisfies the secrecy of the sensitive data, named the master key and the data written in the tag, with respect to a passive intruder, as specified in the environment role.

7 Conclusion

Motivated by the security flaws of the EPC Gen2 RFID specification, and the lack of formally verified security enhancements for this technology, we are working towards the specification and deployment of a key establishment protocol for Gen2 systems. In this paper, we have described the specification and formal verification of an early version of the protocol using the AVISPA model checking tool. We have presented how we described the protocol using the HLPSL format, and verified the secrecy property of the protocol under the presence of a passive adversary. We have then showed how we used two of the verification algorithms implemented in the AVISPA tool to analyze the security of the main cryptographic primitive used in the proposed protocol.

As future perspectives, we aim at continuing the evaluation of the protocol under the presence of an active adversary. In this regard, more security goals must be considered, particularly for guaranteeing the authenticity of the reader. We also plan to define the appropriate key generation functions, and their introduction to the constrained environment of a Gen2 tag. This contribution shall allow us to decide about the refresh period of the established keys.

Acknowledgments — The authors graciously acknowledge the financial support received from Institut TELECOM through its *Future et Rupture* program; and from the Spanish Ministry of Science and Innovation (grants TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER-INGENIO CSD2007-00004 ARES).

References

1. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P.-C. Heam, O. Kouchnarenko, J. Mantovani, S. Moeersheim, D. von Oheimb, Michael R., J. Santiago, M. Turuani, L. Vigano, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. *17th International Conference on Computer Aided Verification (CAV'05)*, pp. 281–285, 2005.
2. A. Armando and L. Compagna. A SAT–based Model Checker for Security Protocols. *9th European Conference on Logics in Artificial Intelligence (JELIA)*, pp. 730–733, 2004.
3. D. A. Basin, M. Sebastian, L. Vigano OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.
4. Y. Boichut, P. C. Heam, O. Kouchnarenko, and F. Oehl. Improvements on the genet and klay technique to automatically verify security protocols. *Automated Verification of Infinite-State Systems (AVIS'2004), joint to ETAPS'04*, pp. 1–11, 2004.
5. Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drieslma, J. Mantovani, S. Modersheim and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. *Workshop on Specification and Automated Processing of Security Requirements (SAPS'04)*, pp.193–205, 2004.
6. T. Cao, E. Bertino, and H. Lei. Security Analysis of the SASI Protocol. *IEEE Transactions on Dependable and Secure Computing*, 6(1):73–77, 2008.
7. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29):198–207, 1983.
8. EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz. *Available at http://www.epcglobalinc.org/standards/.*
9. J. Garcia-Alfaro, M. Barbeau, E. Kranakis. Security Threats on EPC Based RFID Systems *5th International Conference on Information Technology: New Generations (ITNG 2008)*, pp. 1242-1244, 2008.
10. J. Garcia-Alfaro, M. Barbeau, E. Kranakis. Security Threat Mitigation Trends in Low-cost RFID Systems. *2nd SETOP International Workshop on Autonomous and Spontaneous Security SETOP 2009)*, pp. 193-207, 2009.
11. J. Garcia-Alfaro, M. Barbeau, E. Kranakis. Secure localization of nodes in wireless sensor networks with limited number of truth tellers *Communication Networks and Services Research Conference, (CNSR'09)*, pp. 86–93, 2009.
12. A. Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal On Selected Aeas In Communications,* 24(2):381–394, 2006.

13. L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994
14. T. Li and R.H. Deng. Vulnerability Analysis of EMAP-An Efficient RFID Mutual Authentication Protocol. *2nd International Conference on Availability, Reliability and Security*, pp.238–245, 2007
15. T. Li and G. Wang. Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols. IFIP International Federation for Information Security, 2007, pp.108–120.
16. J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. A Practical Implementation Attack on Weak Pseudorandom Number Generator Designs for EPC Gen2 Tags *Wireless Personal Communications*, 59(1):27-42, July 2011.
17. J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags *Financial Cryptography and Data Security*, LNCS vol. 6054, pp. 34–46, 2010.
18. B. Schneier. Applied Cryptography Second Edition: protocols, algorithms, and source code in C. *J. Wiley & Sons, Inc.,* 1996.
19. T. Staake, F. Thiesse and E. Fleisch. Extending the EPC Network — The Potential of RFID in Anti-Counterfeiting, *ACM Symposium on Applied Computing*, pp. 1607–1612, 2005.
20. W. Tounsi, J. Garcia-Alfaro, N. Cuppens-Boulahia, F. Cuppens. Securing the Communications of Home Health Care Systems based on RFID Sensor Networks. *8th Annual Communication Networks and Services Research (CNSR) Conference*, pp. 284–291, 2010.
21. M. Turuani. The CL-Atse protocol analyser. *17th International Conference on Rewriting Techniques and Applications (RTA'06)*, pp. 277–286, 2006