Security Protocols for Low Cost RFID Tags:
Analysis and Automated Verification of
Proposed Solutions
Esteban Masobro Garcia

Technical Report
4 March 2015

Information Security Group
Royal Holloway University of London
Egham, Surrey, TW20 0EX
United Kingdom
Security Protocols for Low Cost RFID Tags: Analysis and Automated Verification of Proposed Solutions

Esteban Masobro Garcia, SRN 070395509
Supervisor: Dr Konstantinos Markantonakis
Royal Holloway, University of London

Submitted as part of the requirements for the award of the MSc in Information Security
March 2014
Declaration

I declare that this assignment is all my own work, and that I have acknowledged all quotations from the published or unpublished works of other people.

I declare that I have also read the statement on plagiarism in the General Regulations for Awards at Graduate and Masters Levels for the MSc in Information Security (Section 9) and in accordance with it I submit this project report as my own work.

Please sign here to show that you have read the above:

Signature:

Date:
To my family and friends.
Acknowledgements

First and foremost, I would like to express my most sincere gratitude to my supervisor, Dr Konstantinos Markantonakis, for his continued support, his efforts, and his trust in my work. He suggested the topic of this project, reviewed the drafts I submitted, pointed me to useful sources, and taught me invaluable lessons throughout the whole project process. The project would not have been possible without him.

I would also want to take advantage of this opportunity to thank all staff at the Information Security Group at Royal Holloway for a unique combination of academic excellence and support at all times during the years leading up to this project. Apart from my supervisor himself, I will always feel indebted to Professor Keith Martin, Dr Carlos Cid, Dr Colin Walter, Dr Michael J. Ganley, Frederik Mennes, Dr Raja Naeem Akram, Martin Warren, Matthew Joyce, and Dr Lizzie Coles-Kemp. I must also express my heartfelt gratitude to Claire Hudson and Daniel Miller for their kind and effective attention during these years. Finally, my appreciation goes to all my fellow students, who contributed to creating an atmosphere characterised by mutual support, hard work and fun.

I also want to express my respect for the relevant experts I contacted occasionally for advice during my years of study, and who were so encouraging and willing to help: Professor Bart Preneel, Dr Veronique Cortier, David Gurlet, Mahdi Asadpour, Wiem Tounsi, Sarah Abu Gazalah and Mario Maawad. I would also like to thank Roger Marshall and Sherman Sounders for proofreading this document for English language.

Last but not least, many thanks to my family and friends for their understanding, patience and support during a rewarding and enjoyable time for me.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Short Background to the Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Statement of Objectives and Rationale</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Methods Used to Achieve the Objectives</td>
<td>4</td>
</tr>
<tr>
<td>1.4 How this Report is Structured</td>
<td>4</td>
</tr>
<tr>
<td>2 Background on RFID Security and Privacy</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Preliminary Concepts</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Security Properties</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Non-protocol Proposals</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Summary</td>
<td>9</td>
</tr>
<tr>
<td>3 Literature Review on RFID Security Protocols</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Proposals Using Public Key Cryptography</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Proposals Using Non-Standard Cryptography</td>
<td>12</td>
</tr>
<tr>
<td>3.2.1 Ultralightweight Authentication Protocols</td>
<td>13</td>
</tr>
<tr>
<td>3.2.2 Lightweight Authentication Protocols</td>
<td>13</td>
</tr>
<tr>
<td>3.2.3 Proposals Based on the LPN Problem</td>
<td>14</td>
</tr>
<tr>
<td>3.2.4 Minimalist Cryptography</td>
<td>14</td>
</tr>
<tr>
<td>3.3 Proposals Using Symmetric Key Cryptography</td>
<td>14</td>
</tr>
<tr>
<td>3.3.1 Protocols Providing Linear Complexity</td>
<td>16</td>
</tr>
<tr>
<td>3.3.2 Protocols Involving the Reader in the Refreshment</td>
<td>17</td>
</tr>
<tr>
<td>3.3.3 Protocols Providing Sublinear Complexity</td>
<td>17</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>18</td>
</tr>
<tr>
<td>4 Analysis of RFID Security Protocols</td>
<td>20</td>
</tr>
<tr>
<td>4.1 Criteria for the Selection of Security Protocols</td>
<td>20</td>
</tr>
<tr>
<td>4.2 First Case Study: The Henrici and Muller’s Protocol</td>
<td>21</td>
</tr>
<tr>
<td>4.2.1 Protocol Description</td>
<td>21</td>
</tr>
<tr>
<td>4.2.2 Security Analysis</td>
<td>22</td>
</tr>
<tr>
<td>4.2.3 Suggestions for Improvement/Fixes</td>
<td>24</td>
</tr>
<tr>
<td>4.3 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol</td>
<td>28</td>
</tr>
</tbody>
</table>
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2.5 Desynchronisation Resistance</td>
<td>100</td>
</tr>
<tr>
<td>A.2.6 Summary</td>
<td>102</td>
</tr>
<tr>
<td>A.3 Non-protocol Proposals</td>
<td>102</td>
</tr>
<tr>
<td>A.3.1 Killing the Tags</td>
<td>102</td>
</tr>
<tr>
<td>A.3.2 Put the Tags to Sleep</td>
<td>103</td>
</tr>
<tr>
<td>A.3.3 Blocking</td>
<td>103</td>
</tr>
<tr>
<td>A.3.4 Proxying</td>
<td>103</td>
</tr>
<tr>
<td>A.3.5 Policy Measures</td>
<td>104</td>
</tr>
<tr>
<td>A.3.6 Summary</td>
<td>104</td>
</tr>
<tr>
<td>B Literature Review on RFID Security Protocols</td>
<td>106</td>
</tr>
<tr>
<td>B.1 Proposals Using Public Key Cryptography</td>
<td>106</td>
</tr>
<tr>
<td>B.1.1 Suitability for Low-Cost RFID Tags</td>
<td>106</td>
</tr>
<tr>
<td>B.1.2 Protocol Proposals</td>
<td>107</td>
</tr>
<tr>
<td>B.2 Proposals Using Non-Standard Cryptography</td>
<td>109</td>
</tr>
<tr>
<td>B.2.1 Ultralightweight Authentication Protocols</td>
<td>109</td>
</tr>
<tr>
<td>B.2.2 Lightweight Authentication Protocols</td>
<td>110</td>
</tr>
<tr>
<td>B.2.3 Proposals Based on the LPN Problem</td>
<td>113</td>
</tr>
<tr>
<td>B.2.4 Minimalist Cryptography</td>
<td>115</td>
</tr>
<tr>
<td>B.3 Proposals Using Symmetric Key Cryptography</td>
<td>116</td>
</tr>
<tr>
<td>B.3.1 Protocols Providing Linear Complexity</td>
<td>116</td>
</tr>
<tr>
<td>B.3.2 Protocols Involving the Reader in the Refreshment</td>
<td>117</td>
</tr>
<tr>
<td>B.3.3 Protocols Providing Sublinear Complexity</td>
<td>119</td>
</tr>
<tr>
<td>B.3.3.1 Hash-Chain-Based</td>
<td>119</td>
</tr>
<tr>
<td>B.3.3.2 Efficient Database Structures</td>
<td>120</td>
</tr>
<tr>
<td>B.3.3.3 The YA-TRAP family</td>
<td>122</td>
</tr>
<tr>
<td>C Analysis of RFID Security Protocols</td>
<td>126</td>
</tr>
<tr>
<td>C.1 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol</td>
<td>126</td>
</tr>
<tr>
<td>C.1.1 Suggestions for Improvement/Fixes</td>
<td>126</td>
</tr>
<tr>
<td>C.1.1.1 Asadpour et al Protocol</td>
<td>126</td>
</tr>
<tr>
<td>D Classification of RFID Security Protocols</td>
<td>128</td>
</tr>
<tr>
<td>D.1 Analysis of the Classification</td>
<td>128</td>
</tr>
<tr>
<td>D.1.1 Presentation of the Results</td>
<td>128</td>
</tr>
<tr>
<td>D.1.2 Example Application</td>
<td>135</td>
</tr>
</tbody>
</table>
E Tools for the Automated Formal Verification of Security Protocols

E.1 Motivation for the Formal Verification of Security Protocols .................................. 137
E.2 Background on Formal Verification of Security Protocols ........................................ 138
E.3 Overview of Surveyed Tools ..................................................................................... 140
   E.3.1 AVISPA/AVANTSSAR .................................................................................. 140
      E.3.1.1 The AVISPA Tool ................................................................................ 140
      E.3.1.2 The AVANTSSAR Platform ................................................................. 143
   E.3.2 Proverif ............................................................................................................ 146
   E.3.3 Scyther ............................................................................................................ 150
   E.3.4 Casper/FDR ..................................................................................................... 152
List of Figures

1.1 Architecture of an RFID system. Picture taken from [1] .............................. 2
4.1 The Henrici and Muller Protocol .............................................................. 23
4.2 Modified Henrici and Muller Protocol ....................................................... 27
4.3 Alomair et al Protocol ................................................................. 30
4.4 Modified Alomair et al Protocol ............................................................... 35
4.5 The $C^2$ Scheme ................................................................. 37
4.6 Modified $C^2$ Scheme ................................................................. 45
B.1 The Original Schnorr Scheme ............................................................... 107
B.2 The Okamoto Scheme ................................................................. 108
E.2 HLPSL Main Elements. Picture taken from [5] ........................................ 143
E.7 Casper/FDR Workflow. Picture taken from [9] ....................................... 155
E.8 Example Casper Input File. Picture taken from [10] ................................ 157
List of Tables

5.1 Protocol Comparison Table - Part 1 - Case Studies and Our Improved Versions ....... 49
5.2 Protocol Comparison Table - Part 2 - Representative Sample of Protocols from Sections 3.3.1 and 3.3.2 ................................................. 50
5.3 Protocol Comparison Table - Part 3 - Representative Sample of Protocols from Section 3.3.3 51
6.1 Suggested Features for the Improvement of the Tools - Current Coverage - Comparison Table 65
A.1 The Multilayered Nature of RFID Privacy - Comparison between Deursen and Avoine et al Models ......................................................... 96
RFID technology offers a number of advantages that make it stand out amongst available auto-ID technologies. It has already been applied in many areas, from large retail stores to car immobilisers and even human beings. Nevertheless, the upcoming widespread adoption of RFID requires the incorporation of suitable privacy and security measures. Security protocol proposals constitute a major mechanism to achieve that goal. However, it must be emphasised that security protocol design is an error-prone task. Consequently, it is essential to have automated tools at our disposal that can formally verify, to some extent, those protocols. The importance of a proof of security is vital in a field where security and privacy are under the spotlight. This project sets out to work on and contribute knowledge to this subject area.

In order to pursue the aforementioned aim, this project features several objectives, the description of which accurately delineates the work that has been done. In the first place, a literature review is carried out on RFID security and privacy, with a focus on protocol proposals. At the same time, a number of security properties which have been found to cover adequately the security and privacy requirements of the field are identified. This is followed by three very detailed case studies, in the form of security protocols, along with suggestions for their improvement. At this point, the project turns to the classification of fifteen security protocols. The resulting comparison is critically analysed. Finally, four tools for the automated formal verification of security protocols are examined, leading to several proposals for the improvement of these tools to better meet the requirements of security protocols for low-cost RFID tags.

The contribution of this project has been three-fold. Firstly, an improved version has been presented for each of the three protocols examined as case studies: namely the Henrici and Muller protocol [11], the Alomair et al protocol [12], and the $C^2$ scheme [13]. Secondly, as already mentioned, a classification of fifteen security protocols is presented and critically analysed. Finally, as also noted, there are a number of suggestions for the improvement of the tools for the automated formal verification of security protocols. These improvements have to do with the ability of the tools to better capture the requirements of security protocols for low-cost RFID tags.
Chapter 1

Introduction

This chapter is intended to introduce the reader to our work, whose main aim is to try to contribute to security and privacy to the RFID field.

We start by providing a short background to the problem. The chapter goes on to state objectives and motivation. Then the methods used to achieve the objectives are briefly described.

Finally, the introduction concludes by explaining how this report has been structured.

1.1 Short Background to the Problem

This section presents a short background to the problem object of this work. We introduce the general purpose, architecture and basic operation of an RFID system. Having done that, we are in a better position to explain why the technology has to overcome security and privacy issues before its widespread adoption. We then note the fact that a great deal of proposals have been made, but we focus on contributing an improvement on protocols for low cost RFID tags. Finally, the error-prone nature of informal proof of security protocol design is highlighted, together with the possibility of using tools for their automated formal verification.

RFID, abbreviation for Radio Frequency Identification, is an Auto-ID (automatic identification) technology. Auto-ID technologies include, apart from RFID systems: barcode systems, optical character recognition, biometrics and smart cards amongst others [1].

The architecture of a traditional RFID system can be described as comprising five components: a large set of resource-constraint tags; a computationally powerful backend system; a set of computationally powerful readers; a communication channel between backend server and readers; and a communication channel between reader and tags. See also figure 1.1 below.

The basic operation in the simplest scenario is as follows: the tag is interrogated by a reader when it enters the reader’s locality. Then the tag replies by sending its ID to the reader. The reader forwards the ID to the backend system. The backend system, having a database with information for all tags, is now able to identify the tag and optionally obtain related information about it.

RFID is widely acknowledged as a technology which will be deployed pervasively, offering significant functional advantages across a wide range of applications, from the tracing of tagged products throughout the supply chain, to pet and drug identification, to name just a few [14, 15, 16].

However, security and privacy issues have to be conveniently addressed before that age becomes a reality. For instance, in the basic scenario described above, an adversary within eavesdropping range could intercept the static identifier emitted by a tag, and thus trace it, and its holder. Furthermore, if that tag
identifier could be linked to information related to the object it is attached to, say a drug name, the adversary could make an educated guess on the holder’s health, which would be a clear invasion of privacy.

As a result, adequate measures are needed to reduce this threat to acceptable levels. Researchers in academia and industry have risen to the challenge and have proposed a wealth of solutions. An online repository can be found at [17]. Some of them, e.g. see [18, 19, 20, 21, 22, 23, 24, 25, 26, 27], have been oriented to the design of suitable security protocols replacing the vulnerable basic operation procedure described above. Some others, e.g. see [3, 28, 29], have been oriented to the design of non-protocol approaches, or a combination of both. In this work, we are mainly concerned with the study of protocol proposals.

In particular, we focus on low-cost RFID tags. In [30], Weis succinctly defines low-cost tags as simple five-cent EPC-type tags. That definition is twofold, namely their cost is around five cents, and their characteristics are similar to those of the EPC-type tags:

1. The first defining characteristic of a low-cost tag, according to [30], is that its cost must be around the five-cent figure if they are to become viable for wide-spread deployment [31, 32, 33]. For us, an important implication is that the complexity of the security mechanisms that we are able to implement on the tag is very constraint. As a result, lightweight solutions to security and privacy for low-cost tags are needed.

2. The second defining characteristic of a low-cost tag, according to [30], is that its characteristics are similar to those tags compliant to the EPCglobal Class-1 Gen-2 standard [34]. EPCglobal [35] is a standards organisation. It aims to see the price of low-cost tags dropping to the 5 cents figure [32], and has developed the EPCglobal Class-1 Gen-2 standard. The interested reader is referred to appendix A.1.3, where the main characteristics of Class-1 Gen-2 tags are overviewed.

The security offered by the standard is, however, very weak, as will also be shown in this work, notably, in the aforementioned appendix A.1.3, and in section 3.2.2. In the latter section, there are several proposals aimed at both ensuring compliance with the standard and improving its security, e.g.
1.2 Statement of Objectives and Rationale

Unfortunately, attaining security and privacy under the constraints of the EPC Class-1 Gen-2 standard is, as will be evident, almost impossible [38].

Nevertheless, even after appropriate security protocols have been designed to meet the demands of resource-constraint low-cost tags, there still remains a further issue: namely, protocol design is an error-prone task. A clear example of this fact is the case of the Needham-Schroeder Public-Key Protocol [39]. It was not until around seventeen years later that it was broken and fixed by Lowe using the tool Casper/FDR [40].

Several methodologies have appeared to try and assist the security protocol designer, including some automated verification tools, such as AVISPA/AVANTSSAR [6, 41], Proverif [42], Scyther [43] or Casper/FDR [44, 45, 46]. A protocol whose security has been formally proven, to some extent, offers a much greater degree of confidence, which, in applications characterised by security sensitivity, is essential.

To contribute to the task of designing security protocols for low-cost RFID tags, this project pursues the objectives described in the next section.

1.2 Statement of Objectives and Rationale

This section comprises a statement of the aims of this project, together with the reasoning behind them.

The following are the objectives our work sets out to achieve:

1. **Review relevant existing literature on RFID security and privacy, with a focus on protocol proposals.**
   This literature search is considered essential so that we can gain the necessary degree of perspective on what has already been done in the field. It is only when this knowledge has been obtained that we are able to contribute critical analysis.

2. **Provide a justified selection of security properties significant to the RFID field.** It must be noted the chosen security properties should adequately cover the security and privacy requirements of the RFID environment.

3. **Provide a justified selection of a number of security protocols as case studies.** The inclusion of case studies, in the form of protocols, is going to be central. They are chosen according to justified criteria.

4. **Offer an analysis of how each case study meets, or fails to meet, the selected security properties, together with suggestions for its possible improvement.** This objective will be addressed as each case study is presented. It is essential to have a clear understanding of the techniques used by each protocol to meet the security properties it satisfies, or, conversely, why it fails to meet them. That is a necessary condition to being able to identify and suggest possible improvements.

5. **Classify a justified selection of security protocols by both their achieved security properties and other performance parameters.** It is important to consider the relationship between protocols because we can gain useful insights from it. For instance, we might observe that some protocols satisfy one property at the expense of another, whereas some others take the opposite strategy. Our case studies and their possible improved versions will be included in the classification.
1. INTRODUCTION

6. Review of a selection of tools for the automated formal verification of security protocols. This literature review is a necessary condition to situate us in a position to attempt to provide suggestions for the improvement of these tools in order to better meet the requirements of the RFID field, as the next objective states.

7. Provide suggestions for the improvement of the reviewed tools to better capture the requirements of RFID protocols. To the best of our knowledge, tools for the automated formal verification of security protocols have been designed to address protocols in general. As a result, one of the aims of this work is to attempt to provide suggestions for the improvement of these tools to better capture the requirements of the RFID field.

In the next section, the methods used to achieve the objectives are described.

1.3 Methods Used to Achieve the Objectives

To achieve the aforementioned objectives, several methods are used:

1. A literature search is needed to obtain information on RFID security and privacy and a selection of tools for the formal verification of security protocols. As a result, sources of information such as the following are used:
   1. Books on RFID technology, e.g. Finkenzeller’s *RFID Handbook* [1], and RFID security, e.g. Cole and Ramasinghe’s *Networked RFID Systems and Lightweight Cryptography* [47].
   2. Research papers on RFID security and privacy, as well as on formal verification of security protocols, with a focus on automated tools.
   3. The Internet, following prudent use recommendation, can be an excellent source of information. Apart from sites of researchers and organisations, it will be necessary to have access to web sites and web interfaces for automated tools.
   4. It is also worth considering making contact with relevant experts for specific information which directly concerns them.

2. Several case studies are included, each one featuring an RFID security protocol, as already noted in this introduction in the previous section.

This introductory chapter comes to an end in the following section, where the structure of this report is described.

1.4 How this Report is Structured

The rest of this report is structured as follows:

1. Chapter 2 sets out to provide a sufficient background on RFID security and privacy to ensure that the reader has a solid foundation to proceed with subsequent chapters. It starts by offering a series of preliminary concepts. Then a study is presented of the security properties we have considered most relevant. The chapter concludes with an overview of a representative sample of non-protocol measures proposed to provide security and privacy to RFID.
1.4 How this Report is Structured

2. Chapter 3 provides a literature review of protocols intended to address security and privacy for low-cost RFID tags. Proposals are subdivided into three different categories, depending on the type of cryptography used: public key, non-standard and symmetric key. From this point on, the project work focuses on protocol proposals only.

3. Chapter 4 starts with a presentation of the criteria used to select three representative protocols which will be used as case studies. Each of the chosen protocols is described. There follows an analysis of the way it meets, or fails to meet, each of the studied security properties. After that, there are some suggestions for its improvement/fixes, leading up to the presentation of an improved version for each of them.

4. Chapter 5 provides a classification of RFID security protocols. It includes our case studies, their respective improved versions, and a selection of other protocols from the literature review. This classification is measured against the studied security properties and a number of performance parameters. The aim of the chapter is to derive further analysis from the comparison.

5. Chapter 6 sets out to provide a review of a selection of tools for the automated formal verification of security protocols. Firstly, the need for formal verification of security protocols is explained, and background information is provided. Subsequently, there is an overview of four salient and representative tools: AVISPA/AVANTSSAR [6, 41], Proverif [42], Scyther [43], and Casper/FDR [44, 45, 46]. Once the necessary background has been established, we attempt to provide suggestions for the improvement of the tools to better capture the requirements of security protocols for low-cost RFID tags.

6. Chapter 7 concludes this work by drawing conclusions and providing pointers to future work.
In the next chapter, more detailed background on RFID security and privacy is offered, so that the reader can have a solid foundation to subsequently proceed with the rest of this work.
Chapter 2

Background on RFID Security and Privacy

This chapter is concerned with the provision of sufficient background on RFID security and privacy. This necessarily implies some brief excursions into the basics of the technology. The aim is to familiarise the reader with the topic so that they are well equipped to approach the rest of this work.

Due to space restrictions mainly, but also because it is background information, most of the material prepared is supplied as appendices. The chapter is subdivided into three sections. Section 2.1 introduces some preliminary concepts. Then section 2.2 specifies the different security properties that we consider in this work. Finally, section 2.3 offers an overview of a representative sample of non-protocol proposals to address RFID security and privacy. For each section, an appendix is referenced where the corresponding material is studied. The material in the appendix is also summarised in the section for the reader’s convenience.

Even though it should be possible to understand this work without reading the material in the appendices, the reader is encouraged to take the time to read them in order to broaden their understanding of the topic. It is also emphasised that this work has been written so that no deep knowledge on the subject area is required. Nonetheless, some familiarity with information security is assumed.

2.1 Preliminary Concepts

The main aim of this section is to provide some essential preliminary background on RFID security and privacy so that the reader is well equipped to deal with the rest of this work. The material corresponding to this section on preliminary concepts is studied in appendix A.1. Even though it must be understood as supplementary material, the reader is encouraged to take the time to read it, unless already familiar with the topic. It is our view that examining it will enrich their understanding of the subject area. For the reader’s convenience, the material in the appendix is summarised below.

In the first place, three main architectural models for RFID security and privacy are identified. The first one concerns scenarios where there is a need to identify tags offline. The second one addresses cloud computing architectures for RFID. The third one is the traditional model. We will limit ourselves to the latter. Within the traditional model, as essential background, its main components are described: a large set of resource-constraint tags; a computationally powerful backend system; a set of computationally powerful readers; a communication channel between backend server and readers; and a communication channel between reader and tags. Once more, we further restrict ourselves by focusing on low-cost passive tags only.
2.2 Security Properties

In addition, the communication channel between readers and tags is identified as insecure, and therefore the central focus of our attention.

It is noted that several standards have been developed to try to ensure interoperability and baseline functionality for different RFID applications. The EPCglobal Class-1 Gen-2 standard \cite{34} is highlighted, and the main characteristics of Class-1 Gen-2 tags succinctly listed: they are passive; contain a unique identifier (the EPC); provide Cyclic Redundancy Code (CRC) and Pseudo-Random Number Generator (PRNG) capabilities; and offer security limited to certain sensitive operations only. Unfortunately, the security of Class-1 Gen-2 tags is also noted to be very weak.

There follows a list of the salient characteristics which make RFID functionally particularly attractive, such as the uniqueness of the identifier associated with an object, or the fact that it requires neither line-of-sight nor preestablished positioning. In addition it is noted that RFID systems have already been deployed in many real-world applications: for instance, large retail stores, drug identification or car immobiliser systems. At this point, it is observed that several factors hinder widespread deployment of RFID, including the cost of the tags, and privacy threats.

The appendix concludes with the important observation of the multilayered nature of RFID security and privacy. It is pointed out that privacy must be enforced at every layer of the RFID communication model. The different layers are described and examples provided of how privacy can be breached at each one of them. In the literature, there are different proposals to define the layers of the RFID communication model. Two well-known models are cited, namely the one by Avoine et al \cite{48} and the one by Deursen \cite{49}. Both of them define three layers: physical, communication and application. However, the levels in both models are not equivalent.

In the next section, the different security properties that we consider in this work are addressed.

2.2 Security Properties

This section is intended to introduce the different security properties that we understand that must be considered when the topic of security protocols for low-cost RFID tags is addressed. The material corresponding to this section on security properties is studied in appendix A.2. Once more, even though it must be understood as supplementary material, the reader is encouraged to take the time to read it. It is our view that an increased understanding of the subject area would be gained if some time was devoted to this. For the reader’s convenience, the material in the appendix is summarised below. It is pointed out that the material in the appendix has been prepared so that for each one of the properties considered, a definition is given, further discussion provided, and their practical significance explained.

In the first place, \textbf{identification} is addressed as a basic requirement for any RFID security protocol. It is stated that it will make it possible for the system to obtain the identity of a tag. Then, it is observed that identification protocols might be sufficient in many RFID applications which do not require additional security assurances. However, some other applications will demand stronger security. Therefore, further security properties must be considered.

Next, both \textbf{unilateral and mutual entity authentication} are considered. With regard to unilateral entity authentication, it is pointed out that it allows one of the parties in the protocol to have the assurance...
that the identity of the other party is as claimed. This assurance is only valid at the moment the protocol ends successfully. In addition, it is explained that mutual entity authentication is satisfied when unilateral entity authentication holds in both directions. Entity authentication is noted as essential in many RFID applications requiring proof of the authenticity of the objects the tags are attached to. [50] offers excellent further treatment of the terminology and techniques involved in identification and entity authentication.

Subsequently, the issue of privacy properties is addressed. These come under the spotlight when the widespread adoption of RFID is considered. The interested reader is referred to appendix A.1.5 for further information. As a result, under privacy, four properties are considered, namely anonymity, universal untraceability, existential untraceability and forward untraceability.

The first one is **anonymity**. It is stated that anonymity consists in the concealment of the identity of an entity involved in some process [50]. It is crucial to ensure that the tag does not leak any information which can lead to the revelation of some information related to its owner.

The second one is **untraceability**. It defined as the ability to prevent other parties from learning one’s current or past location [51]. Two levels of untraceability are differentiated, namely universal and existential. These terms have been borrowed from [12]. **Universal untraceability** is said to be satisfied when two responses from a given tag cannot be correlated if one of them occurs before a valid authentication session and the other one afterwards. In addition, it is said that **existential untraceability** holds when two responses from a given tag cannot be correlated, irrespective of whether a valid authentication session has occurred between them. Following this, it is noted that low-cost RFID tags are not tamper-resistant. The interested reader is referred to appendix A.1.5 for further information. A third and last notion of untraceability results, namely **forward untraceability**, whose definition reflects that fact [12]. An RFID system is said to provide forward untraceability when we cannot correlate the responses from a certain tag before the last valid authentication session, given that the tag has subsequently been compromised.

Finally, **desynchronisation resistance** is defined and its relevance shown. Some protocols assume synchronisation between tags and backend server database, based on the maintenance of consistent values at both sides. Tag and backend server are said to have been desynchronised when an attacker succeeds in disrupting that consistency. A formal definition for a desynchronisation is also provided in accordance to [13]. By way of example, it is also pointed out that the consequences of a successful desynchronisation can include that the backend server becomes unable to identify the tag any longer, which effectively amounts to a denial-of-service attack.

In the next section, an overview of a representative sample of non-protocol proposals to address RFID security and privacy is offered.

**2.3 Non-protocol Proposals**

This section is intended to provide an overview of a representative sample of non-protocol proposals to enhance the security and privacy of RFID systems. The material corresponding to this section on non-protocol proposals is studied in appendix A.3. Even though this appendix can be skipped without loss of continuity, it is our view that the reader would enrich their understanding of the subject area if they read it. Once more, for the reader’s convenience, the material in the appendix is summarised below.
2.4 Summary

It is emphasised that the list of proposals offered is not comprehensive, but, to the best of our knowledge, provides a fairly representative overview of existing proposals to date. The interested reader is referred to [3] for further information.

In the first place, tag killing is described. The reader sends a PIN-protected kill command to the tag, which becomes definitively inoperable after it has been received. It is an effective measure to provide privacy after the killing command. Nevertheless, there are some drawbacks, including the fact that some RFID applications require tags to have a life span beyond that of the objects they are attached to. Consequently, alternative approaches must be considered.

Putting the tags to sleep is an alternative to tag killing, and this is also described in the appendix. The idea would be to turn the tag temporarily inactive, so that it is possible to awake it later, and enjoy the benefits of the technology. For instance, once the customer has returned home after buying a product. Unfortunately, this approach requires some means of putting the tags to sleep and then awake them. PINs are one such possibility, but they might not be easy for the user to manage.

Then the blocking approach is addressed. It consists of using some cheap passive RFID device, the blocker tag, which would be carried by the tag holder and prevent readers from successfully identifying any tag held by the individual. Once more, the user has to take an action to make use of this technique: that is, they must remember to have the blocker tag with them, and, once in an environment perceived as free from security and privacy threats, separate it from ordinary tags.

Next the proxy approach to RFID security and privacy is described. It is noted that a personal device such as an RFID-enabled mobile phone could be used as a proxy. The essential idea would be that the proxy would mediate between readers and tags attached to objects the tag holder owns. The aim would be to take charge of a set of tags specified by the owner and enforce a series of policies in order to protect the privacy of the tag holder. It is also observed that, as it is the case in all the aforementioned non-protocol proposals, the user is actively involved. For instance, to specify the policies to be taken.

Finally, the last approach is described: namely, policy measures. It is pointed out that some organisations strongly oppose the widespread deployment of RFID, e.g. [52]. In view of this debate, authorities have issued recommendations to incorporate security and privacy functionality into RFID systems, e.g. [53, 54]. In addition, sets of principles have also been laid out by technologists and organisations, e.g. [55]: for instance, the right to know when, where and why tags are being read.

The following section summarises and concludes the chapter.

2.4 Summary

This chapter has attempted to provide the reader with some essential background on RFID security and privacy, so that the rest of this work can be approached on the basis of a solid foundation. It has been subdivided into three sections.

Section 2.1 offers some preliminary concepts. We describe the different architectural models for RFID, and the components of the traditional one, which is the one we work with. Then, we briefly approach standards, and the EPC C1G2 in particular. The section goes on to overview a variety of applications of
the technology and the different factors hindering its widespread deployment, if adequate measures are not taken. Finally, we draw attention to the fact that RFID privacy is a multilayered issue.

Then, section 2.2 addresses the different security properties we are concerned with in this work. We provide definitions and descriptions of identification, unilateral and mutual authentication, anonymity, universal and existential untraceability, forward untraceability and desynchronisation resistance.

Finally, section 2.3 provides an overview of non-protocol proposals to RFID security and privacy. Some representative ones are described, including tag killing and putting the tags to sleep, blocking, proxying, and policy measures.

It is our view that it is worthwhile to devote the whole of the following chapter, chapter 3, to providing a literature review of protocol proposals. The reason is that they will be the focus of our work from then on.
Chapter 3

Literature Review on RFID Security Protocols

Last chapter is aimed at providing sufficient background on RFID security and privacy. Some preliminary concepts are introduced, the different security properties considered in this are work specified and an overview of non-protocol proposals is also presented. The aim of this chapter is to provide a review of protocol proposals, which are going to be central for us in this work.

It is not possible to do justice in this work to the wealth of protocol proposals that attempt to provide security and privacy to RFID. An online repository is maintained at [17]. Nonetheless, it is our aim in this chapter to provide, to the best of our knowledge, a literature review of the different trends in the field to date.

We have considered it appropriate to subdivide protocol proposals into three different categories: proposals using public key cryptography, proposals using non-standard cryptography and proposals using symmetric key cryptography. The next three sections consist of a review of these three categories of proposals.

As was the case in last chapter, most of the material prepared for inclusion in this chapter is supplied as appendices. Once more, the main reason for this are space restrictions, but also because it is background information. It should also be possible to understand this work without reading the material in the appendices, but the reader is encouraged to devote some time to reading it to broaden their understanding of the subject area. The chapter will guide the interested reader so that maximum advantage can be obtained. It is also emphasised that this work has been written in such a way that no in-depth knowledge of the topic is required. Nonetheless, some familiarity with information security is assumed.

3.1 Proposals Using Public Key Cryptography

Most protocol proposals have used symmetric key cryptography [38]. Nevertheless, some have been based on public key cryptography (PKC for short). In this section, it is observed that PKC is very appealing in terms of scalability. However, it is also noted that the cost required to implement it on low-cost RFID tags has been considered expensive. As supplementary material, the feasibility of three particular cryptosystems for low-cost RFID tags is examined in appendices. Finally, three protocol proposals based on PKC are also identified. Further discussion on these proposals is offered as supplementary information in appendices as well.

PKC is very appealing with regards to scalability:
3. LITERATURE REVIEW ON RFID SECURITY PROTOCOLS

1. When symmetric key cryptography is used, tags and system share secret keys. For privacy reasons, the tag sends a pseudonym to the system, and not the real identifier. Subsequently, the system has to conduct a database search to obtain the real identifier. The complexity of that search is linear in the number of tags, i.e. $O(N)$ cryptographic operations, unless some technique is used to reduce that cost to sublinear, i.e. under $O(N)$. Unfortunately, when such techniques are used, there is always a price to be paid, such as degradation of privacy to some extent \[38\].

2. However, should a PKC scheme be used, each tag would store its secret key, associated with a public key, and would just have to prove to the system that it knows that secret by using a PKC-based identification protocol. Therefore, as we would have constant-time identification on the server’s side, the scalability problem could be solved.

However, we cannot obviate the cost of implementing PKC on chip and the power consumption and time required by its computations, e.g. point multiplication for Elliptic Curve Cryptography (ECC, for short) \[56, 57\]. Therefore, the main reason PKC has received much less attention from the community has certainly been that it has been considered expensive \[58\]. The interested reader is referred to appendix B.1.1 where we elaborate on this point, and cost implementations for three PKC cryptosystems are reviewed.

Three schemes which have been found representative of the work in the area of protocol proposals based on PKC are: the original Schnorr \[59\], randomised Schnorr \[18\] and Okamoto \[60\] schemes. In appendix B.1.2, the first one is described in detail, and the main differentiating characteristics of the other two are overviewed, with pointers provided for further information to the interested reader.

In the next subsection, protocol proposals based on non-standard cryptography are reviewed.

3.2 Proposals Using Non-Standard Cryptography

In this subsection, a second category of protocol proposals will be reviewed, namely, those using non-standard cryptography.

The main motivation underlying this line of work is the design of security protocol proposals that can be supported on current low-cost tags. To this end, lightweight non-standard cryptography is used. Such constructions would require smaller area requirements and thus would be cheaper than standard cryptographic components.

In this review, we categorise them in the following way:

1. Ultralightweight authentication protocols, in Chien’s classification \[19\]. They use very efficient and simple operations such as XOR ($\oplus$), AND ($\wedge$), OR ($\lor$), modular addition, or data-dependent rotations.

2. Lightweight authentication protocols, again, in Chien’s classification. In this class of protocols, tags provide support for Pseudo Random Number Generators (PRNG) and Cyclic Redundancy Check (CRC), over the previous classification. EPC Class-1 Gen-2 \[34\] compliant tags, for instance, provide both PRNG and CRC (the interested reader is referred to appendix A.1.3 for more information).

3. Proposals based on the Learning Parity with Noise (LPN, for short) problem. Embodied by the HB family, the motivation is the same once more, the design of lightweight alternatives suited for low-cost RFID tags. The security of these proposals is based on the difficulty of the LPN problem.
3.2 Proposals Using Non-Standard Cryptography

4. Minimalist cryptography. The concept of minimalist cryptography is introduced by Juels in [61], and is yet another attempt to improve the security of low-cost tags without resorting to costly standard cryptographic mechanisms. We base our exposition in the mutual authentication scheme presented in that same paper and termed pseudonym throttling.

In the next subsections these categories of proposals using non-standard cryptography are reviewed.

3.2.1 Ultralightweight Authentication Protocols

In this section, representative protocol proposals under the ultralightweight authentication line of work are reviewed. This family of protocols are mainly aimed at achieving low area and low power requirements on tag, and thus low cost. Unfortunately, all proposals under this category show miscellaneous security weaknesses [38]. However, it should be observed that they indeed represent a low-cost alternative which might be appropriate in some applications accepting a less strong adversary model.

These protocols use simple operations, such as XOR (⊕), AND (∧), OR (∨), modular addition, or data-dependent rotations. They are further characterised by the fact that they are stateful, and update their state after every successful authentication. This means they do not provide existential untraceability, but at most universal untraceability. The reader is referred to appendix A.2.4 for additional information on these security properties.

Under this family, these protocols are reviewed in appendix B.2.1: LMAP [62], M2AP [63], EMAP [64], SASI [19], and Eghdamian and Samsudin [65].

In the next subsection, lightweight authentication protocols are reviewed and characterised.

3.2.2 Lightweight Authentication Protocols

In this section, another category of protocol proposals using non-standard cryptography is reviewed. Their main aim is, again, to propose small area and low power designs, and thus feasible for implementation in current low-cost tags. Lightweight protocols assume that tags provide support for the operations used by ultralightweight ones, and, in addition, for Pseudo Random Number Generators (PRNGs) and Cyclic Redundancy Check (CRC).

Tags compliant with the EPC Class-1 Gen-2 standard [34] are indeed lightweight. The interested reader is referred to appendix A.1.3 for more information. However, it is also pointed out in that same appendix that the security offered under the standard is very weak. As a result, many of the proposals under this category aim at compliance with the standard, and, at the same time, to improve its security.

The protocols reviewed under this category in appendix B.2.2 are: the protocol proposal by Chien in [20], Mitra’s scheme [66], the Qingling et al mutual authentication protocol [36], and the mutual authentication protocol by Chen and Deng [37].

In the next subsection, yet another category of protocol proposals featuring non-standard cryptography is reviewed: namely, those based on the LPN problem.
3. LITERATURE REVIEW ON RFID SECURITY PROTOCOLS

3.2.3 Proposals Based on the LPN Problem

The motivation underlying the HB family is the same as the one in ultralightweight and lightweight proposals. Standard cryptographic mechanisms are beyond the capabilities of current low-cost RFID tags. As a result, lightweight alternatives are needed.

The protocols reviewed under this category in appendix B.2.3 are: HB, presented in [67] and adopted for the low-cost RFID field in [68]; HB+, also in [68]; HB++ [69]; Random-HB# and HB# [70]; Trusted-HB [71]; HB-MP [21]; and HB-MP+ [22]. Once more, the reader is encouraged to take the time to read the appendix, where all the aforementioned protocols are examined.

In the next section, the concept of minimalist cryptography is discussed.

3.2.4 Minimalist Cryptography

In this section, the concept of minimalist cryptography, introduced by Juels in [61] is reviewed. The main aim is once more to improve on the security of low-cost tags without resorting to costly standard cryptographic mechanisms. The mutual authentication scheme presented is termed pseudonym throttling. For further details, the reader is referred to appendix B.2.4, where a discussion on the scheme is provided.

In the next section, a literature review of protocol proposals using standard symmetric key cryptography is conducted.

3.3 Proposals Using Symmetric Key Cryptography

In this subsection, we are concerned with the literature review of protocol proposals using standard symmetric-key cryptography mechanisms. Most of the work in the area of security protocols for low-cost RFID tags has been devoted to this category [38]. As we observed in the previous sections, the main reasons for this are that public-key cryptography has been considered expensive for current low-cost tags (see section 3.1), and that proposals using non-standard cryptography to offer lightweight alternatives, even though promising, have various security and privacy issues (see section 3.2).

Once we have chosen to design proposals using standard symmetric key cryptography, a key factor to take into account is the complexity of the search procedure on the server’s side [38]. In a symmetric key solution, each tag shares a secret with the system. If this secret is different for each tag, and the tag does not send its secret in clear to the system on interrogation, the identification procedure on the server’s side takes $O(N)$ cryptographic operations, unless a solution designed to reduce that complexity is used. In other words, the system has to perform an exhaustive search throughout its database. The problem, known as scalability in the literature, is important when we consider large systems containing many millions of tags.

This is the reason why proposals in this category have offered different approaches to deal with this issue, so that the complexity of identification is reduced. This fact is noted in our exposition.

It has been decided to subdivide symmetric-key-based proposals into the following categories. These are not meant to be mutually exclusive, but to make the literature review clearer by grouping related protocols:
3.3 Proposals Using Symmetric Key Cryptography

1. **Protocols providing linear complexity in the identification procedure**: Under this category, protocols which offer linear complexity in the identification procedure on the server side are reviewed. As it has just been observed, those protocols are not scalable to systems comprising many millions of tags. The reason for this is that the time taken to identify a single tag might be unacceptably high in many applications. An example might involve the average waiting time at the checkout of a large store, where customers queue up. In addition, the time taken to identify the whole set of tags managed by the system might also be unacceptably high in many applications. An example might be the time taken by the search for a misplaced book in a library in almost real time.

2. **Protocols involving the reader in the refreshment**: We believe that it is important to devote a second subcategory to those protocols characterised by involving the reader in the refreshment of tags’ secrets. An important issue when approaching this type of proposal is that tags are always traceable between two legitimate authentications, unless the reply from the tag is randomised. However, the randomisation of the reply from the tag requires the server to carry out a more costly procedure to locate the tag’s entry in the server’s database.

   For instance, the $C^2$ scheme [13] adapts the tag’s reply in the second proposal in Dimitriou [24] so that the tag becomes untraceable between two successful authentication sessions. With this aim, it removes the component of the reply that was static between legitimate authentications. The reply becomes then randomised, and thus looks random to an attacker, who cannot trace the tag based on it. Unfortunately, the removed component allowed the server to locate the tag in its database in minimal time. After the removal, the time required to identify the tag has linear complexity in the number of tags.

3. **Protocols providing sublinear complexity in the identification procedure**: In the literature a wealth of protocols designed to reduce complexity can also be found, and they are reviewed under this category. To this end, the following subdivision has been made:

   (a) **Hash-chain-based**. The distinguishing characteristic of the protocols in this group is the use of the hash-chain technique to update the tag’s secrets. Notably, this technique provides forward untraceability, as an attacker compromising a tag is not able to link past sessions of the tag.

   (b) **Efficient database structures, such as trees, groups or grids**. Under this category, several attempts have been made to reduce the complexity of the identification procedure by means of the use of efficient database structures, such as:

   i. **Trees**: The idea is essentially to arrange the database’s secrets in a tree structure. Unfortunately, even though complexity is indeed successfully reduced, privacy is degraded. The Molnar and Wagner’s protocol [26] uses this structure.

   ii. **Groups**: Tags are divided into equally-sized groups, and each group is assigned a secret group key. In addition, each tag possesses its own secret key. As is the case with tree-based structures, compromise of at least one tag leads to degradation of privacy as well. Nevertheless, the group-based structure provides better privacy and complexity levels than the tree-based one. The Avoine, Buttyan, Holczer, and Vajda’s protocol [72] uses this structure.
3. LITERATURE REVIEW ON RFID SECURITY PROTOCOLS

iii. Grids: In this case, the structure is a grid, instead of a tree or a group. Once more, compromise of tags leaks information on other tags. The Cheon, Hong, and Tsudik’s protocol [73] uses this structure.

(c) The YA-TRAP family. This last group embodies an interesting approach at offering sublinear identification complexity. The central idea is that the server maintains a hash table, and pre-computes it each time it increments a monotonically increasing counter. In this way, when an identification has to be conducted, its complexity is constant-time. The interested reader can find further information in section 3.3.3, where protocol proposals using symmetric key cryptography are reviewed, and the YA-TRAP family in particular. In that section, the reader will also be referred to appendix B.3.3.3 for supplementary information.

A similar classification for sublinear protocols to the one presented here can be found in works such as [38].

In the next subsections, these categories of proposals using symmetric-key-based cryptography are reviewed.

### 3.3.1 Protocols Providing Linear Complexity

In this subsection, three representative protocols offering linear complexity in the identification procedure on the server’s side are succinctly introduced. Then as supplementary material, they are described in more detail in appendices. These three protocols are: the Randomised Hash Lock Scheme [74], the Rhee et al protocol [23], and the PEPS protocol [75]. The Hash Lock Scheme [74] has also been included for the sake of clarity only, as the Randomised Hash Lock Scheme is strongly related to it.

A succinct introduction to the selected protocols is provided below:

1. **The Hash Lock Scheme.** [74] As already mentioned, it has been included in this subsection for the sake of clarity. In fact, in terms of security, the Hash Lock Scheme is of historical interest only, as it only provides efficient identification, but no other security property.

2. **The Randomised Hash Lock Scheme** in [74]. It intended to provide untraceability to the Hash Lock Scheme, but at the cost of offering linear complexity and thus making the scheme non-scalable. It is noteworthy that its security profile is quite low.

3. **The Rhee et al protocol:** [23] This scheme is an example of a mutual authentication protocol based on a hash function and a PRNG. Its security profile is quite good, with the caveat that it does not provide forward privacy. Unfortunately, it offers linear complexity, which makes it non-scalable.

4. **The PEPS protocol:** [75] The last scheme that is addressed under this category is the PEPS protocol, standing for Private and Efficient Protocol based on a Stream cipher. Its security and privacy profile is very good. Nonetheless, its non-scalability is indeed a serious drawback in fundamental scenarios.

These protocols are described in more detail in appendix B.3.1. The reader is encouraged to take the time to read that appendix.

In the next subsection, schemes where the reader is involved in the refreshment of the secrets are reviewed.
3.3 Proposals Using Symmetric Key Cryptography

3.3.2 Protocols Involving the Reader in the Refreshment

In this subsection, three representative protocols involving the reader in the refreshment of the tag’s secrets are succinctly introduced. Then, as it was the case in the previous section, they are described in more detail as supplementary material in appendices. These three protocols are: **the second proposal in Dimitriou** [24], the **$C^2$ scheme** [13], and the **Henrici and Muller’s protocol** [11]. The **first proposal in Dimitriou** [24] has also been included for the sake of clarity only, as the second proposal in Dimitriou is strongly related to it.

A succinct introduction to the selected protocols is provided below:

1. **The first proposal in Dimitriou**: [24] As already mentioned, it has been included in this subsection for the sake of clarity. This first proposal provides constant-time identification together with excellent privacy properties, i.e. anonymity, and existential and forward untraceability. Unfortunately, it is easily desynchronised, leading to linear complexity.

2. **The second proposal in Dimitriou**: [24] This proposal slightly changes the previous one. It provides mutual authentication and the tag refreshes its identifier when the reader is legitimate only. Therefore, this is indeed a proper protocol within this subcategory.

3. **The $C^2$ scheme**: [13] Inspired by the second proposal by Dimitriou in [24] just listed above, it introduces some design variations which lead to further privacy but at the cost of a less efficient identification procedure.

4. **The Henrici and Muller’s protocol**: [11]. This protocol is yet another proper example of the protocols in this category. In addition, it is also an interesting case study, as it suffers from several serious weaknesses, which were successfully exploited afterwards. It has been found interesting to examine it in depth as a case study in section 4.2, where significant improvements are suggested which increase its privacy profile substantially, whilst keeping its optimum identification efficiency.

A more detailed examination of these protocols can be found in appendix B.3.2. The reader is, once more, encouraged to devote some time to studying that appendix to gain further understanding of these schemes.

In the next subsection, those protocols designed to reduce the complexity of the identification procedure on the server’s side are reviewed.

3.3.3 Protocols Providing Sublinear Complexity

In this subsection, yet another category of protocol proposals using symmetric key cryptography is reviewed, namely those providing sublinear complexity. This group includes those protocols which were designed to be scalable, either originally or in a subsequent version. This category is in contrast to the one where linear complexity is provided, addressed in subsection 3.3.1.

In this subsection, protocols providing sublinear complexity are subdivided into:

1. **Hash-Chain-based**. Two representative protocols under this subcategory are the **Ohkubo, Suzuki, and Kinoshita’s protocol** (OSK, for short) [25] and the **Avoine et al improvement to OSK** [58]. A detailed description can be found in appendix B.3.3.1.
2. **Efficient Database Structures, such as trees, groups or grids.** Four representative protocols under this subcategory are: the Molnar and Wagner’s protocol [26], the Avoine, Buttyan, Holczer, and Vajda’s protocol [72], the Cheon, Hong, and Tsudik’s protocol in [73], and the authentication extension to the latter presented in the same paper. A review of them can be found in appendix B.3.3.2.

3. **The YA-TRAP family.** Five representative protocols, reviewed in appendix B.3.3.3, are:
   1. The **YA-TRAP protocol**, proposed by Tsudik in 2006 [76], re-named YA-TRIP by Tsudik himself in 2007 [27] and RIP by Burmester et al. in 2009 [77].
   2. The **YA-TRIP protocol** proposed by Tsudik in 2007 [27], and re-named RIP+ by Burmester et al. in 2009 [77].
   4. An extension to the **YA-TRAP** protocol in [27], in an attempt to add forward untraceability to YA-TRAP*.
   5. The **O-TRAP** by Burmester et al., proposed in [78]. In [77] it was re-named O-RAP by the same authors.

The reader is encouraged, once more, to take the time to read the three aforementioned appendices, namely B.3.3.1, B.3.3.2, and B.3.3.3, where all of the protocols listed above in this subsection are examined in detail.

The following section summarises and concludes the chapter.

### 3.4 Summary

This chapter sets out to offer a literature review of protocol proposals that aim to provide security and privacy for low-cost RFID tags. As was noted in the introduction to the chapter, it would be impossible to do justice in this work to the wealth of proposals that have already been made in the field. Nevertheless, the interested reader can find an online repository at [17]. Instead, we focus on the different trends.

Three main categories have been identified: proposals using public-key cryptography, proposals using non-standard cryptography, and proposals using symmetric-key cryptography.

In the first place, proposals using **public-key cryptography** have been addressed. It has been observed that they are very appealing with regard to scalability. Unfortunately, its implementation on low-cost RFID tags has been considered expensive. As supplementary material, the reader has been provided with two appendices. The first one is concerned with the feasibility for low-cost RFID tags of three public-key cryptosystems. The second one addresses the study of three schemes which, to the best of our knowledge, represent the work that has been done in the area of public-key-based protocol proposals up to the present time.

Then proposals using **non-standard cryptography** are reviewed. It is pointed out that the main motivation underlying this line of work is the design of solutions that can be supported on current low-cost tags. With this aim, lightweight non-standard cryptography is used. This category has been further subdivided into:
3.4 Summary

1. Ultralightweight authentication protocols.
2. Lightweight authentication protocols.

Finally, protocol proposals based on standard symmetric-key cryptography mechanisms, where most of the work in the area is to be found [38], are addressed. Under this category, the importance of scalability has been emphasised. Scalability is a key property to ensure proposals can still have an efficient identification procedure on the server’s side in large systems. Once more, to make the literature review clearer, related protocols have been grouped:

1. Protocols providing linear complexity in the identification procedure.
2. Protocols involving the reader in the refreshment of the tag’s secrets.
3. Protocols providing sublinear complexity in the identification procedure: This category has been further subdivided into:
   (a) Hash-chain-based.
   (b) Efficient database structures, such as trees, groups or grids.
   (c) The YA-TRAP family.

This summary concludes the chapter. In the next chapter, three protocols are selected as case studies and examined in detail. The case studies are firstly described and then analysed with regards to a number of security properties identified in section 2.2. Moreover, suggestions for their improvement/fixes are also attempted, leading to an improved version for each one of them.
Chapter 4

Analysis of RFID Security Protocols

In this chapter, three case studies are selected and examined in detail. In section 4.1, the selection criteria are presented and justified.

Following that, in sections 4.2, 4.3, and 4.4, the case studies are firstly described and then analysed with regard to the security properties identified in section 2.2. Furthermore, suggestions for their improvement/fixes are also made, leading to an improved version in all three cases.

Finally, section 4.5 summarises and concludes the chapter.

4.1 Criteria for the Selection of Security Protocols

The following criteria have been applied to select the protocol proposals that we use as case studies:

1. Proposals involving public key cryptography have been discarded. It is true they offer potential advantages, such as providing minimal identification complexity on the server side, without degrading privacy. Nevertheless, we have not considered them because they are viewed as expensive for current low-cost tags, as discussed in section 3.1.

2. Proposals using non-standard cryptographic mechanisms have not been considered either. The reason is that security and privacy issues are usually found when these mechanisms are analysed, as we noted when we conducted the corresponding literature review in section 3.2.

3. Our case studies have been selected so that individually they show prominent techniques for achieving a certain security property, and collectively, their comparison provides helpful insights.

4. Finally, we have tried to include protocols suffering from weaknesses that can be helpful to identify pitfalls to be avoided.

The chosen protocols, meeting the above criteria are:

1. Henrici and Muller’s Protocol: It was proposed in [11] by Henrici and Muller, and a cryptographic hash function is used. We believe it is an interesting case study because it intends to prevent replay attacks using the transaction number. It also attempts to mitigate desynchronisation by maintaining two entries per tag in the system’s database, current and old. However, it also shows certain serious weaknesses.

2. Alomair et al. Protocol: Proposed in [12] by Alomair et al, it is also found to be an interesting case study. The authors attempted to offer constant-time identification complexity, together with a high degree of privacy. It can be classified as a protocol using an efficient database structure. However, it
is important to observe that the approach adopted is different from the ones in the literature review on protocols using efficient database structures (the interested reader is referred to appendix B.3.3.2). In the said appendix, tree-based [26], group-based [72] and grid-based [73] solutions were reviewed.

Unfortunately, the same problem that was found in these solutions is also encountered in the Alomair et al protocol, namely, compromise of tags leads to degradation of privacy. The Alomair et al protocol is examined in detail in section 4.3.

3. The $C^2$ scheme: It was proposed in [13] by Canard and Coisel. We succinctly considered it in section 3.3.2 as part of the literature review. We have found it an interesting case study because its security and privacy properties are excellent. Nevertheless, this is achieved at the cost of making the protocol non-scalable, which makes it a good example of the security/privacy - performance tradeoff. It requires a cryptographic hash function and a pseudorandom number generator on tag. In addition, it also shows a different approach to dealing with desynchronisation, as we also advanced in section 3.3.2. The database does not store current and previous identifier, as our two previous case studies did. It stores only the current one, and, if needed, it calculates the next one on-the-fly.

The next three sections are the central ones of this chapter. They deal with one case study each, in the form of a protocol proposal. The proposal is described and the way it meets, or fails to meet each studied security property identified. The properties considered are those described in section 2.2. Finally, suggestions for improvement/fixes are made, which lead to an improved version in all three cases.

4.2 First Case Study: The Henrici and Muller’s Protocol

In this section we study the Henrici and Muller’s protocol [11].

4.2.1 Protocol Description

We describe the protocol excluding information irrelevant to our discussion. The protocol description is as follows (also see figure 4.1 on page 23):

1. Notation:

   (a) **Data stored at the system per entry:** It must be noted that there are two entries per tag: namely current and old. This technique aims at providing desynchronisation resistance. If the last message from system to tag is delivered unsuccessfully and the tag does not refresh its secrets as a result, the database should still be able to identify the tag by the old entry in the next authentication session.
   1. $h(ID)$: Hash of the current identifier ID.
   2. ID: Current identifier.
   3. TID: Current transaction number at the system (corresponding to this entry).
   4. LST: Last successful transaction number.

   (b) **Data stored at the tag:**
   1. ID: Current identifier.
   2. TID: Current transaction number at the tag.
3. LST: Last successful transaction number.

(c) **Other notation:**
1. X ← Y: X is assigned the value of Y.
2. X ⊕ Y: X is XORed with Y.
3. h(X): Hash value of X, where h is a cryptographic hash function.
4. ΔTID: Difference between current transaction number TID at the tag and last successful transaction number LST, i.e. ΔTID equals TID - LST. ΔTID is used at the system to recover the value of the current transaction number TID used at the tag.
5. TID*: Calculated at the system, its purpose is to contain the current transaction number TID of the tag. It is calculated by adding the last successful transaction number, LST, to the received ΔTID just described above, i.e. TID* ← ΔTID + LST.
6. r: Pseudo-random number generated at the system.

2. **Protocol operation:**
1. The system sends a request message to the tag.
2. On reception, the tag adds 1 to TID and sends the message h(ID), h(TID ⊕ ID), ΔTID to the system, where ΔTID ← TID - LST.
3. Then, the system locates the database entry corresponding to ID by h(ID), it computes a temporal value TID* ← ΔTID + LST, and then checks whether the received value h(TID ⊕ ID) is correct. It also checks TID* > TID, to prevent a replay attack. If everything is correct, it updates TID ← TID* and LST ← TID, generates a random number r, sends r, h(r ⊕ TID ⊕ ID) to the tag, and then updates ID ← ID ⊕ r.
4. Finally, the tag checks h(r ⊕ TID ⊕ ID). If it is ok, the tag updates LST ← TID and its identifier ID ← ID ⊕ r.

In the next section, we address the security analysis of the protocol.

### 4.2.2 Security Analysis

In this section, the way the protocol meets, or fails to meet, the security properties objective of our study (see section 2.2) is analysed:

1. **Identification:** Tags are correctly identified by the database using h(ID) transmitted by the tag.
2. **Tag to reader authentication:** The property is satisfied, because the system has the guarantee that only the tag has been able to construct the received message, as it includes the secret ID. Furthermore, the message must be fresh, because of the use of the transaction counter, which further prevents a replay attack.
3. **Reader to tag authentication:** Similarly, this property is also satisfied. The third message includes the secret ID and the TID just sent from the tag in the second message.
4. **Anonymity:** The property is satisfied, as the ID of the tag is not disclosed to third parties. It is always transmitted within hashes.
4.2 First Case Study: The Henrici and Muller’s Protocol

Figure 4.1: The Henrici and Muller Protocol.

5. **Universal untraceability:** Strictly considered, the property does not hold. The reason for that is that it is possible for an attacker to mount a desynchronisation attack, proposed in [79], which causes no further successful authentications, and thus no more refreshments of the ID, to take place. From that moment onwards, the tag replies with the same h(ID), and thus is definitively traceable. For further details on the attack, see the analysis of the desynchronisation resistance property of the protocol, later in this section.

It is noteworthy, though, that if that desynchronisation attack was not possible, it would not be possible for an attacker to trace a tag by its responses before and after a successful authentication. The reason for this is that the response from the tag has three components \(<h(ID), h(TID \oplus ID), \Delta TID>\):

1. The first and the second are \(h(ID)\) and \(h(TID \oplus ID)\) respectively, which change after each successful authentication, because the ID is updated. In addition, the TID is also updated.

2. The last one is \(\Delta TID\), which is implicitly set to 1 after each successful authentication, as the tag updates \(LST\) to \(TID\).

Therefore, the tag could not be traced by its responses before and after a successful authentication if the desynchronisation attack was not possible.

6. **Existential untraceability:** This property does not hold either, being stronger than the previous one.

In addition to the aforementioned desynchronisation attack, it must also be noted that an adversary...
4. ANALYSIS OF RFID SECURITY PROTOCOLS

can also conduct the following attack between two successful authentications, as noted in [79]. They can query the tag as many times as they wish without completing the authentication successfully. As a result, in this protocol, the value $\Delta TID$ is incremented in such a way that the tag can be distinguished from others by the abnormally high value. After a successful authentication, though, $\Delta TID$ is implicitly set to 1, as the tag updates LST to TID.

Furthermore, even if those two attacks were not possible, the tag would still be traceable between two valid authentications, as it always replies with the same $h(ID)$ because the ID is not refreshed in between.

7. **Forward untraceability**: This property does not hold either. If a tag is compromised, its ID, TID and LST are obtained by the adversary. The attacker is also assumed to have all messages from previous sessions at their disposal. From the ID, it is easy for them to compute previous ID values, as they are calculated by $ID \leftarrow ID \oplus r$. It is also direct to obtain a previous TID by the immediately following LST, and a previous LST by the $\Delta TID$ in the immediately previous message. Consequently, all previous messages can be reconstructed, and thus the responses from the tag before the last valid authentication session can be correlated, i.e. forward untraceability is not provided.

8. **Desynchronisation resistance**: This property does not hold either. In [79], an attack against the protocol is proposed which desynchronises tag and database. The attack relies on the manipulation of the last message from the system. The adversary takes advantage of the use of the exclusive-or operation and manipulates $r$ so that $h(r \oplus TID \oplus ID) = h(TID \oplus ID)$, which is available from the second message. Then the attacker sends the manipulated $r$ and $h(TID \oplus ID)$ as the third message. The effect is that the tag accepts the message and updates its ID XORing it with the manipulated $r$, and thus tag and database are desynchronised from then on. Furthermore, the tag does not update its identifier any more, as there are no more successful authentications, and thus becomes definitely traceable.

9. **Cost of the identification**: The identification complexity is constant-time. The database can be indexed by $h(ID)$ which means no cryptographic operations are needed to find the tag entry. The protocol is, therefore, scalable.

10. **Requirements on tag**: The protocol requires the tag to provide support for a hash function.

11. **Computations on tag**: The tag must perform three hash operations. Two of them to generate the message from tag to system, and the third one to check the hash received from the system in the last message.

In the next section, suggestions for improvement/fixes are attempted.

### 4.2.3 Suggestions for Improvement/Fixes

In this subsection, we point out suggestions for the possible improvement of the protocol. See the resulting modified version in figure 4.2 on page 27 as well.

The notation used by the improved version is the same as the one used by the original version of the protocol, and can be found in section 4.2.1. The only difference is that concatenation is also used in the improved version, and it is denoted by commas, i.e. $X,Y$ denotes $X$ and $Y$ are concatenated.
Our recommendations are structured into three points. In the first one, it is justified that it is not suggested that the transmission of $\Delta TID$ be removed. In the second one it is proposed that the way that the identifier is updated be changed. Finally, in the third one, a recommendation is made in order to address the attacks described in [79] leading to the desynchronisation of tag and system. Our recommendations are discussed below:

1. It is not suggested that the transmission of $\Delta TID$ be removed: In the first place, we want to stress the fact that it would be possible to propose removing the transmission of $\Delta TID$ so that the traceability attack described in section 4.2.2 be rendered impossible. The attack showed that the transmission of $\Delta TID$ allowed an attacker to trace the tag between two valid authentications, i.e. existential untraceability was not satisfied. Nonetheless, we do not propose that measure, because:

(a) Even if we removed the transmission of $\Delta TID$, the tag would still be traceable between two valid authentications. The reason for this is that $h(ID)$ is also sent in the reply from the tag, and it does not change within that period, as the ID is only updated after a successful authentication.
(b) It could then be possible to consider removing both $h(ID)$ and $\Delta TID$ from the tag reply. However, the presence of $h(ID)$ allows the system to locate the tag entry in its database in minimal time, because $h(ID)$ is found as a column in the database. In other words, the scheme is scalable with $h(ID)$. If $h(ID)$ was removed, the system would need to locate the tag entry by the remaining component in the reply from the tag, i.e. $h(TID \oplus ID)$. Recall that both $h(ID)$ and $\Delta TID$ would have been removed.

In order to check $h(TID \oplus ID)$, the system would have to try each tag entry in the database. In normal operation, that would imply one hash computation per entry, because the removed $\Delta TID$ would have been 1. That alone would make the complexity of the search procedure linear in the number of tags, i.e. $O(N)$, and thus, the scheme would become non-scalable.

In fact, the situation would be even worse than that. In the presence of an adversary, the tag could have been queried as many times as the attacker wanted, resulting in a much higher TID value than the one corresponding to the last successful authentication. In that case, the system would be forced to perform a great number of hashes to identify the tag. For each tag entry, all values for TID from the last successful TID plus one to the received TID should be tried. That could amount to a form of a DoS attack against the system.

To address the DoS attack, a security parameter $C$ could be defined as the greatest number of allowed continuous unsuccessful authentication sessions between two legitimate ones. Once the limit was surpassed, the system would alert of a possible DoS attack and abort the authentication, so that out-of-band measures could be taken.

Unfortunately, although the establishment of the security parameter would solve the DoS attack against the system, it would transform it into a DoS attack against the tag, which could not be identified by the system any longer, unless out-of-band action were taken to turn its TID to the correct value.
4. ANALYSIS OF RFID SECURITY PROTOCOLS

To sum up our first point, it is true that the removal of $\Delta TID$ would thwart the traceability attack described in section 4.2.2, but the protocol would remain traceable between two successful authentications anyway. The reason for this is that an attacker can still trace the tag by its $h(ID)$, which remains static within that period.

The additional removal of $h(ID)$ would indeed provide existential untraceability, if the desynchronisation attack also described in section 4.2.2 was solved as well. However, it would create serious new problems. Firstly, it would turn the protocol non-scalable, and secondly, a DoS attack would be made possible, either against the system or the tag.

For these reasons, we make no attempt to remove $\Delta TID$ from the protocol and accept that the scheme does not provide untraceability between two successful authentications. In fact, it is not provided in the original version either, as the ID is not refreshed within that period, and $h(ID)$ is transmitted in the tag's reply. Nevertheless, it is our aim to significantly improve the privacy profile of the protocol, together with the provision of desynchronisation resilience. Via the next two suggestions, forward and universal untraceability and desynchronisation resistance are offered by our modified version, whereas none of them was satisfied in the original Henrici and Muller protocol.

2. **It is proposed that the way that the identifier is updated be changed:** As a first improvement, we identify changing the updating of the identifier from $ID \leftarrow ID \oplus r$ to $ID \leftarrow h(ID)$ at both system and tag. This would provide forward privacy, as the adversary, having compromised the tag and obtained its ID, would not be able to invert the hash function and thus recover previous IDs. As a result, previous messages could not be reconstructed and, therefore, could not be linked to the tag.

We find no issue with this improvement, apart from increasing the computations needed at the tag by one additional hash operation. There is a reduction of one XOR operation at the tag as well, but this is not significant, as XOR is a very efficient and simple operation. On the system side the performance effect of replacing one XOR operation with one hash operation per authentication session should be negligible.

3. **A recommendation to address the attacks described in [79] leading to the desynchronisation of tag and system:** A second improvement addresses the attacks described in [79] leading to the desynchronisation of tag and system. This measure would provide universal untraceability and desynchronisation resistance to our modified version of the protocol, as these attacks were the only ones preventing these properties from being satisfied.

The attacks take advantage of two characteristics of the protocol. Firstly, the properties of XOR (cancellation and neutral element in particular). Secondly, the fact that $h(TID \oplus ID)$ in the second message can be re-used in the third message by carefully manipulating $r$.

We suggest replacing:

(a) $h(TID \oplus ID)$ with $h(0, TID, ID)$ in the second message, where commas denote concatenation. Another symbol frequently used to denote concatenation is $||$.

(b) $<r, h(TID \oplus ID)>$ with $h(1, TID, ID)$ in the third message.

These changes make no difference to neither tag in creation nor database in checking, but they thwart the attack. Firstly, the attacker cannot re-use any component of the second message in the third
4.2 First Case Study: The Henrici and Muller’s Protocol

message. Secondly, we have removed the use of the XOR operator, which also enabled the attack in the original version.

We see no issue with performing this suggestion, and the attacks in [79] are prevented.

![Diagram of the modified Henrici and Muller Protocol]

To sum up, we believe the modified version is indeed an improvement because:

1. It continues to provide efficient constant-time identification, i.e. O(1), mutual authentication, and anonymity.
2. Based on our analysis above and, to the best of our knowledge, on the prevention of known attacks, it additionally provides:
   (a) Universal untraceability. The tag cannot now be traced, except between two valid authentication sessions.
   (b) Forward untraceability. If a tag is compromised, the attacker cannot link past transactions to it.
   (c) Desynchronisation resistance. An attacker cannot apply the attacks in [79]. Consequently, desynchronisation is successfully addressed by maintaining current and old tag entries in the database.

This section has concluded the examination of our first case study, the Henrici and Muller protocol [11]. In the next section, our next case study, the Alomair et al protocol [12], is studied in detail.
4. ANALYSIS OF RFID SECURITY PROTOCOLS

4.3 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol

In this section we study the Alomair et al protocol [12].

4.3.1 Protocol Description

Once more, the protocol is described excluding information irrelevant to our discussion. See figure 4.3 on page 30 as well:

1. **Notation:**
   
   (a) **Data stored at the tag:**
   
   1. \( c \): Cyclic counter which is incremented modulo \( C \) every time the tag is interrogated, i.e. a request message is received by the tag. This counter is initially set to zero at system initialisation. Consequently, \( C-1 \) is the maximum counter value, and the counter can take up to \( C \) values.
   
   2. \( k \): Current shared secret key corresponding to this tag.
   
   3. \( \psi \): Current secret pseudonym corresponding to this tag.

   (b) **Data stored at the system:**

   1. All \( h(\psi_i,c) \) values: At system initialisation, a pool of \( \psi_i \) distinct private pseudonyms is chosen at random, where \( i \in [1...N] \), and \( N \) is the total number of pseudonyms in the system. \( N \) is greater than the number of tags in the system. Then, each tag is assigned a counter, a shared secret key, and a unique pseudonym. The counter is initially set to zero. See data stored at the tag above in this notation description. After that, all \( h(\psi_i,c) \) values are pre-computed, where \( i \) ranges from 1 to \( N \), and \( c \) ranges from 0 to \( C-1 \). These values can be represented as a matrix of \( N \) rows and \( C \) columns. The memory address containing the information for the tag holding pseudonym \( \psi_i \) would be pointed by the \( i^{th} \) row in that matrix.
   
   The particular way the aforementioned matrix is physically instantiated in practice is implementation-dependent. Nevertheless, the authors of the paper [12], which the reader is referred to for further details, make a database structure proposal to manage all the pre-computed data efficiently.

   (c) **Other notation:**

   1. \( r \): Pseudo-random number generated at the system.
   
   2. \( C \): As already mentioned, the counter \( c \) is incremented by 1 modulo \( C \) at each interrogation. As a result, \( C-1 \) is the maximum counter value, i.e. the counter can take \( C \) values.

   3. \( X\leftarrow Y \): \( X \) is assigned the value of \( Y \).
   
   4. \( X\oplus Y \): \( X \) is XORed with \( Y \).
   
   5. \( X,Y \): \( X \) and \( Y \) are concatenated.
   
   6. \( h(X) \): Hash value of \( X \), where \( h \) is a cryptographic hash function.
4.3 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol

The authors themselves mention in the paper [12] that if the last message from system to tag is delivered unsuccessfully, then the tag does not refresh its secrets, but the database does. In order to deal with this desynchronisation issue, they propose a type of technique we already examined in the Henrici and Muller protocol [11], our previous case study. The database must store two pseudonyms per tag: current and old. If desynchronisation has occurred, the database is still able to locate the correct tag entry by the old pseudonym in the next authentication session.

2. Protocol operation:

(a) The system sends a random nonce \( r \) to the tag.
(b) On reception, the tag adds 1 modulo \( C \) to its counter \( c \). Then, it computes \(<h(\psi,c),r'>\), where \( r' = h(0,\psi,c,k,r) \), and sends it to the system.
(c) Then the system:
   1. Locates the database entry corresponding to the tag by the received \( h(\psi,c) \). As all \( h(\psi,c) \) values are pre-computed, the identification complexity is minimal.
   2. Checks correctness of \( r' \) to authenticate the tag.
   3. Randomly chooses a new pseudonym \( \psi' \) from the pool.
   4. Computes \(<h(1,\psi,k,r'),h(2,\psi,k,r')\oplus\psi',h(3,\psi',k,r')\>) \), and sends it to the tag.
   5. The secret key \( k \) of the tag is updated to \( h(k) \) in the corresponding entry.
(d) Finally, the tag:
   1. Checks the received \( h(1,\psi,k,r') \) to authenticate the system.
   2. Obtains the new pseudonym \( \psi' \) from \( h(2,\psi,k,r')\oplus\psi' \).
   3. Checks \( h(3,\psi',k,r') \) for integrity of the received new pseudonym \( \psi' \).
   4. Renews its pseudonym from \( \psi \) to \( \psi' \).
   5. Updates its secret key \( k \) to \( h(k) \).

In the next section, we address the security analysis of the protocol.

4.3.2 Security Analysis

In this section, an analysis is offered of the way in which the protocol meets, or fails to meet, the security properties object of our study, see section 2.2:

1. Identification: Tags are correctly identified by the database using \( h(\psi,c) \) transmitted by the tag. As was already noted in the previous section, when the protocol was described, when the system is initialised, all values \( h(\psi_i,c) \) are pre-computed.
2. Tag to reader authentication: The property is satisfied, because the system has the guarantee that just the tag has been able to construct the received \( r' = h(0,\psi,c,k,r) \), as it includes the secret key \( k \). Furthermore, the message must be fresh, because the nonce \( r \), which was sent by the system in the first message, is also included.
3. Reader to tag authentication: By the same token, this property is also satisfied. The received \( h(1,\psi,k,r') \) includes the secret \( k \) and the value \( r' \) just sent from the tag in the second message.
4. ANALYSIS OF RFID SECURITY PROTOCOLS

4. Anonymity: The property is satisfied, as the identifier of the tag is not disclosed to third parties. Furthermore, only pseudonyms are transmitted, and never in clear.

5. Universal untraceability: To examine whether this property is satisfied or not, two situations must be distinguished:

1. The attacker cannot tamper with tags: In this situation, the property is satisfied. When an authentication is completed successfully, the secret key and the private pseudonym on the tag are updated in an unpredictable way. The new key is assigned the hash of the current key, and a new pseudonym is provided by the system in encrypted form. Responses from the tag \(<h(ψ, c), h(0, ψ, c, k, r)>\), include in the hashes either the pseudonym, or both the key and the pseudonym. As a result, tag’s responses following a successful authentication look random to the attacker and thus cannot be linked.

2. The attacker can tamper with tags: In this work, this is the realistic assumption in a solution for low-cost RFID tags, as tamper-resistance on tag cannot be afforded. In this situation, the property does not hold, because a probabilistic attack can be launched to breach it.

The vulnerability was already noted by the authors themselves in [12]. Basically, if an attacker compromises a tag, they obtain both its secret key and its current private pseudonym. Unfortu-
4.3 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol

nately, when that occurs, the attacker can execute the protocol as many times as they need and obtain further pseudonyms from the system. As we already observed, the scheme is based on the re-use of the pseudonyms.

The attacker can then build a table of $h(\psi_i, c)$ values for the collected pseudonyms, where $c$ ranges from 1 to $C$, and $C$ is the maximum counter value. Consequently, the attacker is able, probabilistically, to distinguish between two tags. For instance, if two tags are shown to the attacker, and the $h(\psi_i, c)$ component of their responses is in the table, their counters $c$ are obtained. There is some probability that an attacker can distinguish between both of them if the value of their counters is noticeably different. If both tags are seen again, the attacker will be able to tell them apart by the value of their respective counters.

We refer the interested reader to the authors’ paper [12] for an introduction to the attack, and to [38] for a refined presentation.

6. **Existential untraceability:** Unfortunately, regardless of whether the attacker has the ability to tamper with tags or not, this property does not hold. The weakness was already noted by the authors themselves in [12]. An adversary can query the tag $C$ times and store all of its responses. As the counter $c$ is cyclic modulo $C$, the tag continues to give one of the responses stored by the attacker until the next successful authentication session with a valid reader. Consequently, the tag can be traced between two successful authentication sessions.

7. **Forward untraceability:** This property does not hold either. The reason for this is the aforementioned probabilistic attack we noted above, when discussing universal untraceability in the presence of an attacker who is able to tamper with tags. Once a tag is compromised, the attacker knows the current counter value $c$ of the compromised tag. If past responses from the same tag are located in the table, their counter values are revealed, and thus probabilistically linked to the tag.

8. **Desynchronisation resistance:** The property holds. As we observed when we described the protocol in the previous section, the database stores two pseudonyms per tag: current and old. If desynchronisation occurs, the database is still able to locate the correct tag entry by the old pseudonym in the next authentication session.

9. **Cost of the identification:** The identification complexity is constant-time. As we have already mentioned, when the system is initialised, all values $h(\psi_i, c)$ are pre-computed. The tag sends $h(\psi, c)$ in its response, and this value is used by the system to identify the tag with minimal complexity, i.e. $O(1)$.

10. **Requirements on tag:** The protocol requires the tag to provide support for a hash function.

11. **Computations on tag:** The tag must perform five hash operations. Two of them to generate the message from tag to system, and three more to check the message received from the system in the last message.

In the next section, suggestions for improvement/fixes are attempted.

4.3.3 Suggestions for Improvement/Fixes

In this subsection, suggestions for the possible improvement of the protocol are pointed out. Firstly, the two main weaknesses leading to the degradation of privacy are identified. Secondly, two suggestions for improve-
4. ANALYSIS OF RFID SECURITY PROTOCOLS

ment are presented which preserve the main structure of the protocol. There follow two recommendations involving more substantial changes to the protocol.

It is important to note the protocol offers constant-time identification, which implies that it is scalable. It also provides mutual authentication, and it is desynchronisation-resistant. Finally, anonymity is satisfied as well. Unfortunately, it does not provide any kind of untraceability: universal, existential or forward. As a result, possible suggestions should focus on the possibility of improving the level of privacy provided.

As already mentioned in the previous subsection, where the security and privacy of the protocol were analysed, the main weaknesses leading to the degradation of privacy are due to the following two attacks [12, 38]:

1. The cyclic counter $c$ on tag, incremented modulo $C$. The attacker can query the tag $C$ times, store its responses, and then have the ability to track the tag up until the next legitimate authentication session. The reason for this is that the tag continues to give one of the responses stored by the attacker until then.

2. The possibility of an attacker compromising a tag, and then running the protocol as many times as needed to obtain further pseudonyms from the system. These pseudonyms, as has been already observed, are re-usable and they are sent to the tag in encrypted form. An attacker in possession of a number of pseudonyms can build a table of $h(\psi, c)$ values. Subsequently, tags using pseudonyms in the table can be distinguished probabilistically by the value of their respective counters, even after successful authentication sessions. Between valid authentication sessions, tracking is even easier, as tags use the same pseudonym.

Some suggestions for the improvement of the level of privacy of the protocol while preserving its main structure are presented below:

1. In the first place, we propose incrementing the counter without using modular addition. This prevents an attacker from exploiting the first of the two vulnerabilities listed above. The tag would not generate its responses in a cyclical way between legitimate authentications and thus existential untraceability could not be breached in this way.

   Nonetheless, a DoS attack is then made possible. An adversary querying the tag more than $C$ times, where $C$ is the maximum counter value, is lost by the system. The reason for that is the system is not be able to find it in its database any longer. Of course, the tag could be re-incorporated into the system by out-of-band action. A similar issue occurs in the OSK protocol if the tag is queried by an illegitimate reader more than $m$ times, where $m$ is the maximum length of the hash chain (see appendix B.3.3.1, where the protocol is reviewed, for further information). Again, the tradeoff between the level of untraceability and DoS resistance should be studied with regard to the target application.

2. As a second improvement, we propose initialising the counter value on tag randomly at system setup, instead of setting it to zero. That would improve the level of privacy with regards to the second attack above. Indeed, it was noted that an attacker can distinguish two tags probabilistically if both of them are found in the table of $h(\psi, c)$ values created from captured pseudonyms $\psi$.

   The adversary can indeed obtain the current counter values of both tags if they are on the table. It could be that one of the two tags was issued long ago, but has not been queried more than $C$ times.
yet, i.e. its counter value is high. The other tag, though, might have been issued a short time ago and thus have a low counter value. That leads to a traceability issue by counter value. We remind the interested reader that an introduction to the attack can be found in the authors’ paper [12], and a refined presentation in [38].

Our suggestion removes the correlation between counter value and the amount of time the tag has been in the field, and thus improves privacy. To the best of our knowledge, our suggestion is novel.

The suggestions for improvement that have been proposed so far have been made so that the scheme preserves its main structure. However, it would also be possible to make further suggestions if that restriction was relaxed. To this end, two recommendations which involve more substantial changes to the protocol are made below. In the first place, notation is introduced, then the suggestions are presented, and finally they are justified. See the improved version resulting from these two suggestions in figure 4.4 on page 35 as well.

_in the first place, notation is introduced_: The notation used by the improved version is the same as the one used by the original version of the protocol, and can be found in section 4.3.1. The only differences are:

1. Data stored at the system:
   1. \(c, k, \psi\): The system’s database stores only one entry per tag, or two entries (current and old) if desynchronisation resistance is required. The tag entry includes the same data that can also be found on the tag, i.e. the current shared secret key \(k\), and the current secret pseudonym \(\psi\).

2. Other notation:
   1. \(r_{tag}\): Pseudo-random number generated at the tag.
   2. \(r_{sys}\): Pseudo-random number generated at the system.

_in the second place, our two suggestions are presented_: They are meant to be applied together.

1. We propose replacing \(h(\psi, c)\) with \(\psi\) in the first message. Furthermore, the usage of the counter \(c\) is removed altogether. The main aim of the counter was to provide untraceability between legitimate authentication sessions. However, due to the first weakness listed at the beginning of this subsection this was not achieved.

2. We suggest the addition of a tag nonce \(r_{tag}\) in the first message from the tag to the system, both in clear and as an additional parameter of the hash \(r'\). The reason for this is that the nonce is needed to ensure freshness of the third message. One of the aims of the third message is the authentication of the system to the tag.

_in the final place, our suggestions are justified:_

In the original version, the protocol provides neither universal, existential nor forward untraceability, as it has already been justified in this section. Implementing our suggestions, existential untraceability is not provided either, but both universal and forward are. This is the justification for this statement:
1. **Universal untraceability is provided.** In our improved version, the tag response consists of three components, and all of them are unpredictable to an attacker immediately after a successful authentication:

   (a) The first component is the current pseudonym $\psi$: Even though the tag sends its pseudonym in clear, immediately after a successful authentication, its value is unpredictable to an attacker because:
      
      i. The attacker does not know its value as it has just been sent in encrypted form by the system.
      ii. There is no correlation between this pseudonym and previous pseudonyms used by the tag.
      The reason for this is that pseudonyms are chosen by the system at random from the pool of pseudonyms.

   (b) The second component is a nonce ($r_{tag}$) just generated by the tag, and thus it is unpredictable to an attacker.

   (c) The third component is the hash $h(0,\psi,r_{tag},k,r_{sys})$, which includes $r_{tag}$. As a result, it is also unpredictable to an attacker.

2. **Existential untraceability is not provided.** Similarly to the case of our modified Henrici and Muller protocol, we emphasise that tags are still traceable between two valid authentication sessions in our modified Alomair et al version. The reason for this is secrets are refreshed at the tag when the last message authenticating the system is successful. If this message is not delivered successfully, the tag can be tracked by $\psi$ in the tag’s reply, where $\psi$ does not change within that period.

3. **Forward untraceability is provided.** The justification for this is that, if a tag is compromised, and thus its current pseudonym $\psi$ and shared secret key $k$ are captured by an attacker:

   (a) knowledge of the current shared secret key $k$ does not allow an attacker to recover past values of it, because it is updated using a one-way function, namely a cryptographic hash function.

   (b) knowledge of the current pseudonym $\psi$ does not allow an attacker to recover past pseudonyms either, because, as we have already observed, they are not correlated amongst them.

As a result, previous messages sent from the tag cannot be reconstructed.

It is important to emphasise that, as was also the case in our modified Henrici and Muller protocol with $h(ID)$, and for similar reasons (see section 4.2.3), we could always remove $\psi$ from the message from tag to system over our modified version. That would make the protocol existentially untraceable, i.e. untraceable even between two valid authentications. However, the complexity of the identification would turn linear in the number of tags, i.e. $O(N)$, instead of remaining constant-time, i.e. $O(1)$. The reason for that is $r' = h(0,\psi,r_{tag},k,r_{sys})$, should be used by the system to locate the tag’s entry in its database. Note $r'$ is the other component of the message which can potentially identify the tag, once the pseudonym $\psi$ has been removed. Avoidance of that degradation of the complexity of the identification procedure on the server’s side makes us decide to preserve the pseudonym $\psi$ in the response from the tag. In other words, we decide to preserve scalability at the cost of not providing untraceability between two legitimate authentications. The tradeoff, once more, should be considered depending on the target application.

To sum up, it has been shown the modified version is indeed an improvement because:
4.4 Third Case Study: The $C^2$ Scheme

1. It continues to provide efficient constant-time identification, mutual authentication, anonymity, and desynchronisation resistance.

2. Based on our analysis above and, to the best of our knowledge, on the prevention of known attacks, it additionally provides:
   1. Universal untraceability. The tag cannot be traced, except between two valid authentication sessions.
   2. Forward untraceability. If a tag is compromised, the attacker cannot link past transactions to it.

In this section, we have built on the Alomair et al protocol, and we have done so independently. Nevertheless, we wish to acknowledge that we are aware of the existence of the Asadpour et al protocol [80]. Their protocol bears some resemblance to our improved version of Alomair et al. The interested reader is referred to appendix C.1.1.1 where the main characteristics of the Asadpour et al protocol are described, removing information irrelevant to our discussion, and emphasising similarities to our improved version of Alomair et al where appropriate.

This section has concluded the examination of our second case study, the Alomair et al protocol [12]. In the next section, our next case study, the $C^2$ scheme [13], is addressed.

4.4 Third Case Study: The $C^2$ Scheme

In this section we study the $C^2$ scheme [13].
4. ANALYSIS OF RFID SECURITY PROTOCOLS

4.4.1 Protocol Description

We describe our third case study, the \( C^2 \) scheme. The tag and the system share a secret key \( k_{\text{tag}} = k_{\text{sys}} \). As we noted in section 4.1 when we introduced the chosen case studies, the database does not store the previous identifier to deal with desynchronisation. It stores the current one only, and, if needed, it calculates the next one on-the-fly. As a result, just one entry per tag is required. Finally, we point out that three cryptographic hash functions \( h, g, \) and \( i \) are used.

The protocol description is as follows, see figure 4.5 on page 37 as well:

1. **Notation:**
   
   (a) **Data stored at the system per tag entry:**
   1. \( k_{\text{sys}} \): Shared secret key.
   2. \( \text{ID} \): Tag identifier.

   (b) **Data stored at the tag:**
   1. \( k_{\text{tag}} \): Shared secret key.

   (c) **Other notation:**
   1. \( X \leftarrow Y \): \( X \) is assigned the value of \( Y \).
   2. \( h, g, i \): \( h, g \) and \( i \) are cryptographic hash functions, e.g. \( h(X) \) equals the hash value of \( X \) using function \( h \).
   3. \( r_{\text{tag}} \): Pseudo-random number generated at the tag.
   4. \( r_{\text{sys}} \): Pseudo-random number generated at the system.

2. **Protocol operation:**
   
   (a) The system sends a nonce \( r_{\text{sys}} \) to the tag.

   (b) On reception, the tag generates a nonce \( r_{\text{tag}} \). Then, it computes \( <r_{\text{tag}}, r> \), where \( r = h(k_{\text{tag}}, r_{\text{sys}}, r_{\text{tag}}) \), and sends it to the system.

   (c) Once received, the system:
   
   i. Tries to locate the database entry corresponding to the tag:
   
      1. In the first place, the system performs an exhaustive search. For each entry in the database, it computes a temporary value \( r' = h(k_{\text{sys}}, r_{\text{sys}}, r_{\text{tag}}) \) and compares it with the received \( r \). If the entry is found, the search procedure is over, and \( k_{\text{sys}} \) is not updated yet.

      2. If the entry has not been found, the tag might be desynchronised. The system then performs an additional exhaustive search. For each entry in the database, it computes a temporary value \( r' = h(g(k_{\text{sys}}), r_{\text{sys}}, r_{\text{tag}}) \) and compares it with the received \( r \). If the entry is found, the search procedure is over, and \( k_{\text{sys}} \) is updated to \( g(k_{\text{sys}}) \), so that \( k_{\text{sys}} \) equals \( k_{\text{tag}} \). Otherwise, if the entry is not found, the tag is rejected.

   The correctness of the received \( r \) authenticates the tag.

   ii. Computes \( r' = h(g(k_{\text{sys}}), r_{\text{sys}}, r_{\text{tag}}) \) and sends it to the tag.

   (d) When the tag receives the message from the system:
4.4 Third Case Study: The $C^2$ Scheme

1. Computes a temporary value $r'' = h(g(k_{tag}), r_{sys}, r_{tag})$ and checks it equals the received $r'$. If it does, the system is authenticated.
2. Updates its secret key $k_{tag}$ to $g(k_{tag})$.
3. Computes $i(k_{tag})$ and sends it to the system.

(e) Finally, once the system has received the last message from the tag, it checks its correctness. If it is correct, the system updates its secret key $k_{sys}$ to $g(k_{sys})$.

Figure 4.5: The $C^2$ Scheme.

In the next section, we address the security analysis of the protocol.

4.4.2 Security Analysis

In this section, the way the protocol meets, or fails to meet, the security properties object of our study, see section 2.2, is analysed:

1. Identification: Tags are correctly identified by the system using $r = h(k_{tag}, r_{sys}, r_{tag})$ transmitted by the tag. An exhaustive search through the system’s database is needed, as we noted in the previous section, when we described the protocol. Therefore, the complexity of the identification procedure at the server’s side is linear in the number of tags, i.e. $O(N)$.
2. Tag to reader authentication: The property is satisfied, because the system has the guarantee that only the tag has been able to construct the received message, as it includes the secret $k_{tag}$. Furthermore, the message must be fresh, because it also includes the nonce $r_{sys}$, which was transmitted by the system to the tag in the first message.
3. Reader to tag authentication: By the same token, this property is also satisfied. The third message includes the secret $k_{sys}$ and the nonce $r_{tag}$. The latter was just sent from the tag in the second message.
4. **ANALYSIS OF RFID SECURITY PROTOCOLS**

4. **Anonymity:** The property is satisfied, as the identifier of the tag is not transmitted, and thus it is not disclosed to third parties. Not even the secret $k_{tag}$ is disclosed, as it is never transmitted in clear.

5. **Existential untraceability:** This property is satisfied. The tag sends two messages to the system:
   1. In the first message from the tag, the response from the tag is composed of two parts both of which are randomised. The first is a nonce just generated by the tag, and the second is a hash computation which includes that nonce.
   2. In the second message from the tag, the response is $i(k_{tag})$, where $i$ is a cryptographic hash function. This response looks random to an attacker as well, as it is computed just after the tag has updated its secret $k_{tag}$ to $g(k_{tag})$, where $g$ is another cryptographic hash function, and $k_{tag}$ is unknown to the attacker.

6. **Universal untraceability:** This property holds as well, being weaker than the previous one.

7. **Forward untraceability:** This property is also satisfied. If a tag is compromised, its $k_{tag}$ is obtained by the adversary. The attacker is also assumed to have all messages from previous sessions at their disposal. However, $k_{tag}$ is updated to $g(k_{tag})$ after each successful authentication. That implies it is computationally unfeasible for an attacker to recover previous values of $k_{tag}$, as $g$ is a one-way function. Without previous $k_{tag}$, the attacker cannot reconstruct previous responses from the tag before the last valid authentication session. Consequently, they cannot be correlated.

8. **Desynchronisation resistance:** This property holds as well. As we already noted in the previous section, the protocol addresses desynchronisation. The tag updates its secret only after it authenticates the system. And the system updates its secret when it receives confirmation of the tag’s update in the last message of the protocol. That means the only possible desynchronisation occurs when the last message is not successfully delivered. Then, the tag updates its secret, and the system does not. However, a legitimate reader resynchronises its secret at the next authentication session on reception of the second message of the protocol. This occurs once the search procedure reveals the tag is not using the current secret but the next one.

9. **Cost of the identification:** The identification complexity is linear in the number of tags, i.e. $O(N)$. Unfortunately, that makes the protocol non-scalable. The system must perform an exhaustive search through its database to find the tag’s entry from the received $<r_{tag},r>$, where $r = h(k_{tag}r_{sys}r_{tag})$. If the tag is desynchronised, a further exhaustive search is needed, as we noted in the previous section when we described the protocol.

10. **Requirements on tag:** The protocol requires the tag to provide support for a hash function and a pseudo-random number generator (PRNG, for short).

11. **Computations on tag:** The tag must perform four hash and one PRNG operations. One hash and one PRNG operation to generate the hash and the nonce respectively in the first message. There then follow two further hash computations to check the third message of the protocol, and compute the new value of the secret. Finally, a last hash to compute the last message of the protocol.

In the next section, suggestions for improvement/fixes are attempted.
4.4 Third Case Study: The C\textsuperscript{2} Scheme

4.4.3 Suggestions for Improvement/Fixes

In this subsection, suggestions are attempted for the possible improvement of the protocol. It is important to note that the protocol offers excellent privacy and security levels, including mutual authentication, anonymity, existential untraceability, forward untraceability and desynchronisation resistance. Nonetheless, this comes at the cost of a linear identification complexity, i.e. O(N), which makes the protocol non-scalable. This fact makes this case study an excellent example of the security-performance tradeoff. Our suggestions, therefore, inevitably focus on providing better performance, while trying to preserve the high degree of security and privacy as much as possible.

The presentation of our suggestions in this subsection has been structured as follows:

1. **Presentation of our three recommendations.** These are oriented to the:
   (a) **Reduction of the gate count on tag.** To this end, it is recommended that instead of three different hash functions, only one hash function, with a one-bit parameter added to it be used.
   (b) **Reduction of communication cost and computation time on tag.** It is proposed that the desynchronisation resistance technique of the original scheme be replaced with the one used in several other schemes, e.g. [11, 12, 80].
   (c) **Improvement of the identification complexity at the server’s side.** To achieve this objective, it is suggested that \textit{h(k\textsubscript{tag})} be added to the first message.

2. **Presentation of the improved version of the protocol** resulting from the implementation of our recommendations.

3. ** Provision of some final notes with regards to the improved version.**

Our three recommendations are presented below:

1. **In the first place,** it is recommended that instead of three different hash functions, only one hash function with a one-bit parameter added to it be used. This suggestion is intended to reduce the gate count on tag. The price of low cost tags must continue to drop to the 5-cent figure if it is to achieve wide-spread deployment [31, 32, 33]. The interested reader is referred to appendices B.1.1 and appendix A.1.5 for further information. In addition, it has also been noted that the cost of adding a security mechanism can be estimated as a function of the number of gates needed to implement its digital components on chip [81]. Consequently, in particular, we must try to reduce the gate count on a low-cost tag as much as possible.

   As a result, the following improvement is proposed. The \textit{C\textsuperscript{2}} protocol uses three different cryptographic hash functions, namely \textit{h, g, and i}. We suggest using only one hash function with a one-bit parameter added to it. That would clearly reduce the gate count on tag. This technique is used in our second case study, the Alomair et al protocol [12]. In addition, it was proposed by Avoine in [82].

   As an example application, in the \textit{C\textsuperscript{2}} protocol, we could use function \textit{h} only, instead of the three functions \textit{h, g} and \textit{i}. We would replace \textit{h(k\textsubscript{tag}, r\textsubscript{sys}, r\textsubscript{tag})} in the second message with \textit{h(1,k\textsubscript{tag}, r\textsubscript{sys}, r\textsubscript{tag})}. By the same token, we would replace function \textit{g} with function \textit{h} and parameter 2. Finally, we would replace function \textit{i} with function \textit{h} and parameter 3.
2. Our second suggestion is replacing the desynchronisation resilience technique used by the C² scheme with the one used in several other schemes, e.g. [11, 12, 80]. This suggestion aims at reducing communication costs and computation time on tag.

As we have already noted in our exposition of this case study, the desynchronisation resistance technique used by the C² scheme is based on storing only the current identifier, and, if needed, calculating the next one on-the-fly. The technique used by several other schemes, e.g. [11, 12, 80], is characterised by the storage of both current and old identifiers. As a result, the technique used by the C² scheme uses only around half the space required by those other schemes. Nevertheless, this saving in space comes at the cost of extra communication and computation costs for the tag. In particular, one additional message from the tag and one additional hash computation on tag are needed. These are the fourth message, and its contents i(k_{\text{tag}}) respectively.

Our suggestion considers that the total computation time on tag should be as low as possible in a security protocol for low-cost RFID tags. This also applies to communication costs. At the same time, it takes into account the fact that server’s database can afford a space capacity well in excess of the needs of a relatively large RFID system. We justify our statements below:

(a) Firstly, we study the significance of the space capacity in the server database. A large RFID system such as a big library can be taken to serve as an example. The Hamburg library, for instance, has already tagged more than two million books, CDs and DVDs [83]. We use $2^{21}$ (a power of two), which equals two million approximately, to simplify our calculation. Even if we store several kilobytes of information per tag entry, say 16 KB, which equals $2^{14}$ bytes, this amounts to a $2^{21} \cdot 2^{14} = 2^{35} = 32$ GB requirement. Currently, we can find desktop hard disks with capacities of up to 4 TB, such as the ones by Seagate [84].

(b) Secondly, we argue that communication costs must be as low as possible in an RFID scenario. The reason for this is that an RFID reader can read several hundreds of tags per second. As an example, ThingMagic Inc. released the Mercury6e, a UHF EPC Gen-2 RFID embedded reader module, claiming read rates of up to 400 tags per second [85]. That leads to the need for efficient collision avoidance protocols, which manage multi-access without interference [1]. Therefore, as noted in [12], an increase in communication costs can only make anti-collision more complex.

(c) Finally, the need to keep computation costs as low as possible is justified. As we have just observed, an RFID reader can read hundreds of tags per second. If all tags have to be read within one second, the time available for encryption operations per tag is limited. This strongly suggests, we believe, that it is good practice to design RFID security protocols so that the time taken by the cryptographic computations is minimised. In particular, we believe it is advisable to try to minimise the number of computations performed on tag. However, we still have to provide evidence to support our argument. We do so by referencing the work of Feldhofer et al in [86]. The authors propose an AES architecture to meet the strict requirements of low-cost RFID tags. They also note that compatibility with existing standards should be ensured. In particular, they work on the ISO/IEC 18000-3 in its March 2003 version [87]. They observe that (the reader is referred to the authors’ paper [86] for further details):
4.4 Third Case Study: The $C^2$ Scheme

1. The internal clock frequency of the tag must be reduced under 100 kHz to meet the low-power requirement of a low-cost tag.

2. The standard requires a response is sent by the tag 320 $\mu$s after a request. The tag should not operate otherwise.

3. This leads to a time of 32 cycles at 100 kHz, which is not enough time to perform one encryption with the AES.

4. Then, they propose modifying the challenge-response protocol so that challenges and responses are interleaved. Using this technique, up to 50 tags can be authenticated per second, and each tag has at least 1800 cycles at 100 kHz to perform the cryptographic computations.

5. This time, as their AES proposal encrypts in around 1000 cycles, it is possible to perform one encryption at a frequency under 100 kHz, which meets the requirements of low-cost tags.

The interleaving technique provides 1800 cycles (at 100 kHz) to each tag to perform one encryption, in a scenario where 50 tags are authenticated per second. The AES proposal by the authors, which encrypts in around 1000 cycles, meets the requirement. It is also true some other works have further reduced the number of cycles required for one encryption. For instance, in [88] the authors present a Tiny Encryption Algorithm (TEA, for short) parallel core claiming a 64 cycles figure.

Nevertheless, we have already observed many more tags can be authenticated per second. The Mercury6e from ThingMagic Inc., claiming read rates of up to 400 tags per second [85] is one such example. In addition, security protocols usually feature several cryptographic operations, and not just one. For instance, the $C^2$ scheme features four hash operations and one PRNG operation [13].

In conclusion, we believe there is enough evidence to support the view that minimising the number of cryptographic operations per tag is a desirable property for a security protocol for low-cost RFID tags.

The arguments provided above have attempted to support the point that the implementation of our suggestion to replace the desynchronisation resilience technique used by the $C^2$ scheme is worthwhile. The reason for this is the $C^2$ scheme’s desynchronisation resilience technique reduces server’s database space capacity at the cost of increasing computation time on tag and communication costs. We have argued that the opposite is desirable in an RFID environment for low-cost tags.

3. **As a third recommendation, it is suggested that $h(k_{tag})$ be added to the first message.**

The objective of our third recommendation is the improvement of the identification complexity at the server’s side. The $C^2$ scheme is non-scalable, because its identification complexity is linear in the number of tags, i.e. $O(N)$. Therefore, it cannot be used in those applications where the RFID system contains a large number of tags, and the time to identify one tag is important. We further expand on this statement below. Therefore, we believe it is worthwhile the suggestion of a proposal which improves identification complexity, even if it is at the cost of losing some degree of privacy.

We believe this proposal must be optional, because the relative importance of scalability and privacy depends on the target application. For instance, scalability is essential in major retail stores. The
number of tags is large, and customers queue at the checkout and are not willing to wait an unacceptable amount of time until the identification procedure is completed. On the other hand, in an RFID system that manages a relatively small number of tags, we might prefer the higher degree of privacy offered by the \( C^2 \) scheme: for example, in an access control application for a small to medium-sized company. Especially for those applications where scalability is important, there are several reasons why it is important that the identification of the tag at the server side does not take too long, including:

1. One reader must be able to read hundreds of tags per second. In our previous suggestion for improvement, we referenced the Mercury6e from ThingMagic Inc. as an example. It claims read rates of up to 400 tags per second [85].
2. The previous point is reinforced by the fact that the RFID system’s database might have to support concurrent access from many readers. For instance, the Hamburg library we referenced in our previous suggestion, features 40 readers [83].
3. Finally, the system might also have to identify all tags it manages in (almost) real-time to conduct some operations. For example, to search for a misplaced book in a library.

Having exposed the argument supporting our optional proposal, we present it: \textit{we suggest adding} \( h(k_{tag}) \) \textit{to the first message.} That means the first message becomes \(<r_{tag}, h(k_{tag}), r>\), where \( r = h(k_{tag}, r_{sys}, r_{tag}) \). That change implies:

1. \textit{The identification procedure becomes constant-time.} The system locates the tag entry by \( h(k_{tag}) \), which is found as a column in the database, ideally as a primary key. The modified \( C^2 \) scheme is thus, scalable and offers a minimal identification time, unlike the original version.
2. \textit{Unfortunately, scalability is achieved at the cost of losing existential untraceability,} which holds in the original version. The tag replies with the same \( h(k_{tag}) \) within the period between two legitimate authentication sessions, because \( k_{tag} \) is not updated at the tag in between. If we tried to randomise \( h(k_{tag}) \), our modified version of the protocol would become non-scalable, like the original one. The reason for this is that the system would need to perform an exhaustive search to locate the tag’s entry.
3. \textit{All other security and privacy properties are satisfied} in both our modified version and the original one, namely all of them except for existential untraceability.
4. \textit{An additional hash computation is required on tag} after applying our suggestion. However, if it is combined with our second recommendation, i.e. replacing the desynchronisation resilience technique, the number of computations on tag is the same for the original \( C^2 \) scheme and our modified version.

We finally observe that the implementation of this suggestion can be found in protocols such as both proposals in [24] by Dimitriou, or our first case study, the Henrici and Muller protocol [11].

For the sake of clarity, \textit{the presentation of the improved version of the protocol is provided below.} It incorporates the implementation of our three suggestions on the original \( C^2 \) scheme. See figure 4.6 on page 45 as well:
4.4 Third Case Study: The C² Scheme

1. **Notation:** The notation used by the improved version is the same as that used by the original version of the protocol, and can be found in section 4.4.1. The only difference is related to the data stored at the system:

- **Data stored at the system:** In the modified version, the current and previous values of \( h(1, k_{\text{sys}}) \), where \( k_{\text{tag}} = k_{\text{sys}} \) if tag and system are synchronised, are stored as columns in a table in the system’s database. These values are denoted as \( \text{HK}_{\text{curr}} \) and \( \text{HK}_{\text{prev}} \) respectively, and are used by the system to efficiently locate the tag’s entry in the system’s database. It is noted that \( h(1, k_{\text{tag}}) \) is sent to the system by the tag in the second message of the protocol.

In addition, it is emphasised that storing current and old values is a technique to prevent desynchronisation. In this protocol, if the last message from system to tag is not delivered successfully and the tag does not refresh its secret key \( k_{\text{tag}} \) as a result, the system should still be able to efficiently identify the tag by the previous value \( \text{HK}_{\text{prev}} \) in the next authentication session. For similar reasons, the current and previous values of \( k_{\text{sys}} \) are stored in the tag’s entry. If desynchronisation occurs, the system should still be able to authenticate the tag in the next session.

Consequently, the values stored on the system’s side per tag entry are:

(a) \( \text{HK}_{\text{curr}} \): The current value of \( h(1, k_{\text{sys}}) \).
(b) \( \text{HK}_{\text{prev}} \): The previous value of \( h(1, k_{\text{sys}}) \).
(c) \( k_{\text{sys}} \): The shared secret key between tag and system, at the system.
(d) \( k_{\text{sysprev}} \): The previous value of \( k_{\text{sys}} \).
(e) \( \text{ID} \): Tag identifier.

2. **Protocol operation:** It is emphasised that both system and tag share a secret key, initially set to the same value \( k_{\text{tag}} = k_{\text{sys}} \).

(a) The system sends a nonce \( r_{\text{sys}} \) to the tag.
(b) On reception, the tag generates a nonce \( r_{\text{tag}} \). Then, it computes \( < h(1, k_{\text{tag}}), r_{\text{tag}}, r_{\text{sys}} > \), where \( r = h(1, k_{\text{tag}}, r_{\text{sys}}, r_{\text{tag}}) \), and sends it to the system.
(c) Once received, the system:
   i. Tries to locate the database entry corresponding to the tag by the received \( h(1, k_{\text{tag}}) \). The system performs the following three actions sequentially:
      1. Firstly, it tries to find an entry in the database containing the received \( h(1, k_{\text{tag}}) \). The column corresponding to the current value \( \text{HK}_{\text{curr}} \) is used. If the entry is found, the tag is identified.
      2. If the entry is not found, the tag might be desynchronised, in which case, the column corresponding to the previous value \( \text{HK}_{\text{prev}} \) is used. If the entry is found, the tag is identified. Furthermore, it becomes resynchronised.
      3. If the entry is not found, the tag is rejected.
   ii. If the tag was identified by a match in the column corresponding to the previous value \( \text{HK}_{\text{prev}} \), the system checks whether \( h(1, k_{\text{sysprev}}, r_{\text{sys}}, r_{\text{tag}}) \) equals the received \( r \). If it does, the tag is authenticated.
4. ANALYSIS OF RFID SECURITY PROTOCOLS

iii. If the tag was identified by a match in the column corresponding to the current value HK\textsubscript{curr}, the system:
A. Checks whether h(1, k\textsubscript{sys}, r\textsubscript{sys}, r\textsubscript{tag}) equals the received r. If it does, the tag is authenticated.
B. Updates HK\textsubscript{prev} to the value of HK\textsubscript{curr} in the tag’s entry.
C. Updates HK\textsubscript{curr} to h(1,h(2,k\textsubscript{sys})).
D. Updates k\textsubscript{sysprev} to k\textsubscript{sys}.
E. Updates the shared secret key k\textsubscript{sys} to h(2,k\textsubscript{sys}).

iv. Computes r’ = h(1,k\textsubscript{sys},r\textsubscript{sys},r\textsubscript{tag}) and sends it to the tag.

(d) When the tag receives the message from the system:
1. Computes a temporary value r” = h(1,h(2,k\textsubscript{tag}),r\textsubscript{sys},r\textsubscript{tag}) and checks it equals the received r’. If it does, the system is authenticated.
2. Updates its secret key k\textsubscript{tag} to h(2,k\textsubscript{tag}).

Finally, some final notes are provided with regards to the improved version:
Summarising, we believe the modified version is an improvement because:

1. It continues to provide mutual authentication, anonymity, universal and forward untraceability and desynchronisation resistance.
2. Based on our analysis above, it additionally provides:
   1. Efficient constant-time identification, i.e. O(1), which makes the protocol scalable. In the original version, identification has complexity linear in the number of tags, i.e. O(N), making the protocol non-scalable.
   2. The gate count on tag is decreased. The modified version uses only one hash function with an additional parameter, whereas the original version uses three different hash functions.
   3. Communication costs are also reduced. The last message of the original protocol is not included in the modified version.

Unfortunately, as we noted, constant-time identification, and thus scalability, are achieved at the cost of losing existential untraceability, which is satisfied in the original version. In the modified version, tags are traceable between two legitimate authentications. Nonetheless, if the protocol is to be used in a large system where the identification time is relevant, it must be scalable. In other words, the C\textsuperscript{2} scheme can only be used in systems where the number of tags managed is relatively small. This justifies, we believe, the implementation of our third proposal to make the protocol scalable. Of course, small systems can take advantage of the original scheme and its higher degree of privacy.

We also noted that in the original version the database space needed is just around half the space required in the modified version. This saving in database space in the original version, though, is obtained at the cost of increasing the total computation time on tag, and the communication cost. We discussed the opposite tradeoff is in fact desirable in an RFID system. Consequently, our second recommendation doubles the database space required, which we have shown to be perfectly affordable, in order to reduce computation on
This section has concluded the examination of our third and last case study, the $C^2$ scheme [13]. In the next section, we summarise the exposition of our three case studies.

4.5 Summary

This section is intended as a summary of the chapter.

In section 4.1, the chapter starts with the establishment of selection criteria to choose three protocols as case studies. These protocols are examined in-depth in sections 4.2, 4.3, and 4.4. For each protocol, a description is provided. Then, a security analysis is conducted. Finally, suggestions for its improvement/fixes are proposed. In particular, one improved version, which we claim is an improvement on the original one, is presented for each protocol.

The three case studies selected have been:

1. The Henrici and Muller’s Protocol [11]. After describing the protocol, there follows an analysis, the conclusion of which is that this protocol provides efficient constant-time identification, mutual authentication and anonymity. Unfortunately, it is also shown not to provide any kind of untraceability, and to be desynchronisable. Finally, it requires a hash function implemented on tag, and the tag must perform three hash computations. Then, several improvements/fixes are suggested, which eventually lead to an improved version of the

![Diagram of Modified $C^2$ Scheme](image-url)
4. ANALYSIS OF RFID SECURITY PROTOCOLS

protocol. We recommend several changes to the original protocol, which substantially improve its privacy profile. The improved version maintains all the properties that the original protocol satisfies and it further offers universal and forward untraceability, and desynchronisation resistance.

2. Alomair et al. Protocol [12]. Once the protocol is described, its security analysis is conducted. It is pointed out that efficient constant-time identification is provided, mutual authentication, anonymity and desynchronisation resistance. Unfortunately, it is also shown that the protocol does not provide any kind of untraceability, except for universal and forward untraceability when the attacker cannot tamper with tags. However, if we take into account the fact that low-cost tags are not tamper-resistant, this is not a realistic assumption. In addition, it is further noted that the protocol requires the implementation of a hash function on tag, and the tag must perform five hash computations. Subsequently, several improvements/fixes are proposed. The focus is on addressing weaknesses of the protocol that lead to privacy degradation. Our modified proposal improves the privacy profile of the original Alomair et al protocol substantially. The modified version continues to provide mutual authentication, anonymity, desynchronisation resistance and an efficient constant-time identification. Even though tags continue to be traceable between two legitimate authentications, the modified version additionally provides universal and forward untraceability. Finally, it is noted that the improved version requires the additional implementation of a PRNG on tag.

3. The $C^2$ scheme [13]. As was the case with our two previous case studies, after the description of the protocol, a security analysis is conducted. Excellent security and privacy are provided. All of mutual authentication, anonymity, universal, existential and forward untraceability, and desynchronisation resistance are provided. Unfortunately, the identification complexity is linear in the number of tags, which makes the protocol non-scalable. It is further highlighted that three different hash functions and one PRNG on tag are needed, and four hash and one PRNG computations are required on the tag’s side. Having addressed the security analysis, a number of improvements/fixes are proposed. The original protocol already offers excellent security and privacy. Consequently, our suggestions focus on improving performance, gate count on tag and communication cost while maintaining the security and privacy degree as high as possible. In fact, after implementing our suggestions, all properties satisfied in the original scheme are preserved, except for existential untraceability. Nonetheless, our modified version additionally offers constant-time identification, instead of linear. Therefore it is scalable. Furthermore, the gate count on tag is decreased, and communication costs reduced.

This section has summarised and finished the chapter. In the next one, our case studies, their respective modified versions, and a number of other protocols selected from the literature review in chapter 3, are classified. The classification is made according to the studied security properties the protocols meet, or fail to meet, and a number of performance parameters. The objective is to conduct further analysis from their comparison.
Chapter 5

Classification of RFID Security Protocols

In this chapter, a classification of a number of security protocols is made. The protocols included are:

1. The three case studies in chapter 4, namely the Henrici and Muller protocol [11], the Alomair et al protocol [12], and the $C^2$ scheme [13].
2. Our modified versions of the three case studies above, which we claim to be an improvement on them.
3. A number of other protocols selected from the literature review in chapter 3. These protocols have been chosen according to the criteria presented in section 4.1. Consequently, they are proposals using symmetric key cryptography, see section 3.3. Their selection is intended to provide further analysis from their comparison.

The classification of the aforementioned protocols is made in accordance to:

1. Three criteria to categorise them. These criteria have been chosen so that coherence is maintained between the classification in this chapter and the classification in section 3.3, where the literature review on proposals using symmetric key cryptography is made. The criteria selected are:
   (a) **The complexity of the identification at the server’s side**, namely whether it is **linear**, i.e. $O(N)$, or **sublinear**, i.e. under $O(N)$, in the number of tags.
   (b) **The type of refreshment of the tag’s secrets**. In this work, the possible values are, and are defined as:
      i. **No refreshment**: Tag’s secrets are not refreshed.
      ii. **Reader-triggered refreshment**: Tag’s secrets are refreshed only after the tag authenticates the reader.
      iii. **Self-refreshment**: The tag refreshes its secrets after each interrogation.
   (c) **The family of the protocol**. In this work, the possible values are, in accordance with the literature review in section 3.3: **historical interest**, **linear complexity**, **reader-triggered refreshment**, **hash-chain**, **efficient database structures**, and **YA-TRAP family**.

We emphasise that the different families are not meant to be mutually exclusive. The category assigned reflects the characteristic of the protocol that has been considered most salient, despite the fact that more than one classification could have been applied.
5. CLASSIFICATION OF RFID SECURITY PROTOCOLS

2. The security and privacy properties described in section 2.2. Those properties are: identification, entity authentication (tag to reader and reader to tag), anonymity, universal untraceability, existential untraceability, forward untraceability and desynchronisation resistance.

3. Performance and implementation requirements: the selected characteristics have been:
   
   (a) The complexity of the identification procedure on the server side.
   
   (b) The relevant implementation requirements on tag, such as the need for a hash function and/or a PRNG.
   
   (c) The relevant computations required on tag, such as the number of hash computations performed.
   
   (d) The communication costs. Under this characteristic, the number of exchanged messages is shown.

The objective of the chapter is to derive further analysis from the comparison of the selected protocols. In section 5.1, the next one, three tables are presented with a view to providing a succinct and clear view of the classified protocols by the aforementioned characteristics. Then, in section 5.2, an analysis of the classification is made, which is intended to provide additional insights into the design of security protocols for low-cost RFID tags. Finally, section 5.3 summarises the chapter.

5.1 Presentation of the Classification

In this section we present three tables in order to classify a number of security protocols for low-cost RFID tags:

1. The first one includes the case studies we examined in detail in chapter 4, together with our improved/fixed versions. See table 5.1 on page 49.

2. The second one includes a representative sample of protocols that were reviewed in section 3.3.1 on protocols providing linear complexity and 3.3.2 on protocols involving the reader in the refreshment. See table 5.2 on page 50.

3. Finally, the third one includes a representative sample of protocols from section 3.3.3, on protocols providing sublinear complexity. See table 5.3 on page 51.

As noted in the introduction to this chapter, the protocols in the tables have been categorised according to three criteria, namely its identification complexity, its type of refreshment and the family of the protocol. They have also been further classified with regard to several security and privacy properties. Furthermore, performance and implementation requirements on tag have also been considered, together with communication costs. Finally, it is observed that footnotes have been added below the tables when the need to elaborate on the contents of a cell has been identified.

In the next section, the classification depicted in the tables is analysed in order to derive further insights on security protocol design for low-cost RFID tags.
### 5.1 Presentation of the Classification

Table 5.1: Protocol Comparison Table - Part 1 - Case Studies and Our Improved Versions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Refreshment</td>
<td>Reader-triggered</td>
<td>Reader-triggered</td>
<td>Reader-triggered²</td>
<td>Reader-triggered²</td>
<td>Reader-triggered</td>
<td>Reader-triggered</td>
</tr>
<tr>
<td>Family</td>
<td>Reader-triggered refreshment</td>
<td>Reader-triggered refreshment</td>
<td>Efficient database structures</td>
<td>Efficient database structures</td>
<td>Reader-triggered refreshment</td>
<td>Reader-triggered refreshment</td>
</tr>
</tbody>
</table>

| Identification             | Yes                      | Yes                                           | Yes               | Yes                                    | Yes           | Yes                                |
| Tag to reader authentication | Yes                     | Yes                                           | Yes               | Yes                                    | Yes           | Yes                                |
| Reader to tag authentication | Yes                     | Yes                                           | Yes               | Yes                                    | Yes           | Yes                                |
| Anonymity                 | Yes                      | Yes                                           | Yes               | Yes                                    | Yes           | Yes                                |
| Universal untraceability  | No                       | Yes                                           | No³               | Yes                                    | Yes           | Yes                                |
| Existential untraceability | No                       | No                                            | No⁴               | No                                     | Yes           | No                                 |
| Forward untraceability    | No⁷                      | Yes                                           | No³               | Yes                                    | Yes           | Yes                                |
| Desynchronisation resistance | No                     | Yes                                           | Yes               | Yes                                    | Yes           | Yes⁵                               |
| Cost of the identification | O(1)                    | O(1)                                          | O(1)              | O(1)                                   | O(N)          | O(1)                               |
| Requirements on tag       | Hash                     | Hash                                          | Hash              | Hash, PRNG                             | Hash⁶ PRNG    | Hash, PRNG                         |
| Computations on tag       | 3 hash                   | 4 hash                                        | 5 hash            | 4 hash + 1 PRNG                        | 4 hash + 1 PRNG | 4 hash + 1 PRNG                    |
| Communication cost         | Three-pass               | Three-pass                                    | Three-pass        | Three-pass                            | Four-pass     | Three-pass                         |

1 In the number of tags.
2 It must be noted that the tag refreshes its secret key when it checks the reader is legitimate. In addition, the next pseudonym is provided by the system in encrypted form.
3 Provided an attacker can tamper with tags.
4 Provided an attacker can tamper with tags. Otherwise, if an attacker can query the tag C times, it is not satisfied either. C is the maximum value of the counter c stored on tag.
5 The desynchronisation resistance technique used is different from the one used in schemes such as Henrici and Muller, Alomair et al or Asadpour et al [11, 12, 80], where both current and old identifiers are stored. The C² scheme [13] stores the current identifier only, and, if needed, calculates the next one on-the-fly.
6 Three different hash functions are used, whereas only one hash function with an additional one-bit parameter is used in our modified version.
7 Forward untraceability is not satisfied. Our improved version changes the updating of the identifier from ID ← ID ⊕ r to ID ← h(ID) to provide it.
### Table 5.2: Protocol Comparison Table - Part 2 - Representative Sample of Protocols from Sections 3.3.1 and 3.3.2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Refreshment</td>
<td>No refreshment</td>
<td>Reader-triggered</td>
<td>Self-refreshment</td>
<td>Reader-triggered</td>
</tr>
<tr>
<td>Family</td>
<td>Linear complexity</td>
<td>Linear complexity</td>
<td>Historical interest</td>
<td>Reader-triggered refreshment</td>
</tr>
<tr>
<td>Identification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Tag to reader authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reader to tag authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anonymity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Universal untraceability</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Existential untraceability</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Forward untraceability</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Desynchronisation resistance</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cost of the identification</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(1) - O(N)²</td>
<td>O(1)</td>
</tr>
<tr>
<td>Requirements on tag</td>
<td>Hash, PRNG</td>
<td>Encryption, PRNG</td>
<td>Hash, PRNG</td>
<td>Hash, PRNG</td>
</tr>
<tr>
<td>Computations on tag</td>
<td>2 hash + 1 PRNG</td>
<td>Stream cipher encryption³ + 1 PRNG</td>
<td>3 hash + 1 PRNG</td>
<td>4 hash + 1 PRNG</td>
</tr>
<tr>
<td>Communication cost</td>
<td>Three-pass</td>
<td>Three-pass</td>
<td>Two-pass</td>
<td>Three-pass</td>
</tr>
</tbody>
</table>

1. In the number of tags.
2. O(1) if not desynchronised, otherwise O(N).
3. The amount of keystream produced at each protocol run equals 2l + k, where l is the length of an authentication response, and k is the length of the key input to the stream cipher.
4. Even though it is not explicitly stated by the author in his paper [24], in this work we assume current and old tag entries are stored in the system’s database. Consequently, desynchronisation resistance is addressed.
### 5.1 Presentation of the Classification

Table 5.3: Protocol Comparison Table - Part 3 - Representative Sample of Protocols from Section 3.3.3.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Refreshment</td>
<td>Self-refreshment</td>
<td>No refreshment</td>
<td>No refreshment²</td>
<td>No refreshment²</td>
<td>No refreshment²</td>
</tr>
<tr>
<td>Family</td>
<td>Hash-chain</td>
<td>Efficient database structures</td>
<td>YA-TRAP family</td>
<td>YA-TRAP family</td>
<td>YA-TRAP family</td>
</tr>
<tr>
<td>Identification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Tag to reader authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Reader to tag authentication</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Anonymity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Universal untraceability</td>
<td>Yes</td>
<td>No¹²</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Existential untraceability</td>
<td>Yes</td>
<td>No¹²</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Forward untraceability</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Desynchronisation resistance</td>
<td>No³</td>
<td>Yes¹⁰</td>
<td>No</td>
<td>No</td>
<td>Yes⁴</td>
</tr>
<tr>
<td>Cost of the identification</td>
<td>From O(1) to O(N)⁵</td>
<td>O(logN)</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1) - O(N)⁶</td>
</tr>
<tr>
<td>Requirements on tag</td>
<td>Hash</td>
<td>Hash, PRNG</td>
<td>Hash, PRNG</td>
<td>Hash, PRNG</td>
<td>Hash</td>
</tr>
<tr>
<td>Computations on tag</td>
<td>3 hash¹¹</td>
<td>(2 Hash + 1 PRNG)⁷</td>
<td>1 Hash or 1 PRNG¹²</td>
<td>(2 Hash and 1 PRNG) or 2 PRNG¹⁴</td>
<td>1 Hash⁹</td>
</tr>
<tr>
<td>Communication cost</td>
<td>Two-pass</td>
<td>(3-d)-pass⁹</td>
<td>Two-pass</td>
<td>Two-pass</td>
<td>Two-pass</td>
</tr>
</tbody>
</table>

¹ In the number of tags.
² The counter value is refreshed, but the secret keys are static.
³ An attacker can desynchronise a tag from the database if it queries it more than a parameterised number of times. The attack, though, is detected. In addition, the tag holder can resynchronise the tag by out-of-band action.
⁴ It is easy to desynchronise tag and database. The attacker only needs to interrogate the tag once. However, synchronisation is re-gained in the next successful protocol run. In that regard, we say it is desynchronisation resistant.
⁵ Depending on the time-memory trade-off used.
⁶ In normal operation O(1). If desynchronised, O(N).
⁷ d is the depth of the tree structure. The basic challenge-response protocol requires 2 hash and 1 PRNG computations. This is repeated as many times as levels has the tree structure.
⁸ Two hashes were used in the original version O-TRAP [78], one for r and another one for h. In the version in [77] both r and h are obtained in the same hash.
⁹ d is the depth of the tree structure. The basic challenge-responses protocol requires three messages. This is repeated as many times as levels has the tree structure.
¹⁰ Secrets are not refreshed. Therefore, no synchronisation is maintained between tag and system. In that regard, we say it is desynchronisation resistant.
¹¹ In order to apply the time-memory trade-off in [89], three hashes must be computed on tag, as noted in [82]. The first to update the tag’s secret, the second to allow the system to identify the tag, and the third to allow tag’s authentication.
¹² Provided an attacker can tamper with tags. Otherwise, it is satisfied.
¹³ As a result of a conditional step, some responses from the tag might involve 1 hash computation, and some others 1 PRNG computation.
¹⁴ As a result of a conditional step, some responses from the tag might involve 2 hash + 1 PRNG computations, and some others 2 PRNG computations.
5.2 Analysis of the Classification

In this section, an analysis of the classification presented in the previous section is conducted.

This analysis has been structured around a number of characteristic patterns that have been observed from the classification depicted in the tables. It must be emphasised that each pattern involves most of the protocols in the tables, but exceptions to the general rule might exist.

Below, a list of nine observed patterns is presented and discussed. Due to space limitations, the discussion on each one of these points is an extract of the one that can be found in appendix D.1.1. Even though it is not necessary to read the appendix to understand the chapter, the interested reader is encouraged to take the time to do so. We point out that for the ninth point on desynchronisation resistance, all the discussion in the appendix has been included. The reason for this is that amongst the protocols in the classification there are a variety of different techniques used to achieve that security property. As a result, in our view, useful insights can be obtained from its detailed analysis. These are the nine patterns that have been identified:

1. In general, protocols that have linear complexity in the identification procedure on the server’s side, i.e. non-scalable, have excellent security and privacy properties. Therefore, the protocol designer has to tradeoff identification complexity, i.e. scalability, against security and privacy. The PEPS protocol [75] and the $C^2$ scheme [13] are two examples of linear complexity protocols under this pattern. They satisfy all security and privacy properties in the tables.

2. Protocols that do not refresh tag’s secrets do not offer forward untraceability. This observation must be taken into account when forward privacy is desired in a security protocol. This is the reason why the Rhee et al protocol [23] does not offer forward untraceability. The same applies to the Molnar and Wagner protocol [26], and the three protocols from the YA-TRAP family we have included, namely RIP [77], RIP+ [77] and O-RAP [77].

3. In general, protocols that refresh tag’s secrets using a one-way function provide forward untraceability. This pattern complements the previous one. Refreshing tag secrets is not a sufficient condition to provide forward untraceability. In addition, the refreshment must be performed using a one-way function, so that an attacker cannot reverse the computation and thus obtain previous values of the secrets from the captured ones.

   Protocols in the tables that agree with the general pattern include: the already mentioned modified versions of the Henrici and Muller protocol and the Alomair et al protocol; the $C^2$ scheme [13] and its modified version in this thesis; the PEPS protocol [75]; the two proposals by Dimitriou [24]; and the Avoine et al improvement to OSK [58].

   The Henrici and Muller protocol [11] is an example of a protocol that refreshes its secret, but does not use an irreversible computation. Instead, the secret is updated to the result of XORing it and a nonce which has just been received.

4. Sublinear protocols with reader-triggered refreshment do not offer existential untraceability. In other words, tags are traceable between two legitimate authentications in sublinear protocols which refresh tag’s secrets only after the tag has authenticated the reader. We note some authors have made observations along the same lines. Most notably Avoine in [82], but also Dimitriou in [90]. This
5.2 Analysis of the Classification

is another pattern that can be observed in the tables presented. If existential untraceability is required by the target application, this category of protocols should then be avoided. Nevertheless, these protocols could offer existential untraceability if the reply from the tag was randomised. However, that would lead to a more costly procedure to locate the tag’s entry in the server’s database. See section 3.3, where proposals using symmetric key cryptography are addressed, for additional discussion, or appendix D.1.1, where we elaborate further on the results of this analysis.

5. **All protocols in the tables provide identification.** This must be considered a minimum requirement for any RFID security protocol. The system must be able to identify the tags it manages, otherwise the RFID system serves no purpose.

6. **All protocols in the tables provide anonymity.** This privacy property is satisfied by all protocols in the tables. That suggests that it is a desirable and important property for any general RFID security protocol. The consequences of the lack of anonymity include the leakage of sensitive information about an individual holding a tagged product. The interested reader is referred to appendix A.2.3 for further information.

A much used mechanism to provide anonymity amongst RFID security protocols is the transmission of a cryptographic hash of the identifier, instead of transmitting it in clear. Examples include the two proposals by Dimitriou [24], and the Henrici and Muller protocol [11] and our corresponding modified version.

7. **Most protocols in the tables provide mutual authentication.** Mutual authentication is identified in the tables as a usual design objective. Nevertheless, whether it is required or not will depend on the target application. Some applications might not need security beyond simple identification, e.g. identification of lost pets. However, many others require some form of authentication (unilateral or mutual), as in anti-counterfeiting applications.

8. **Most protocols in the tables provide universal untraceability.** In the first place, it is important to note that universal untraceability is weaker than existential untraceability. In order to provide universal untraceability, protocols do not need to prevent the tracking of tags between two successful authentications, whereas in order to provide existential untraceability protocols must prevent it within that period as well.

Having made that observation, it is noted that universal untraceability is provided by most protocols in the tables, whereas existential untraceability is provided by only around half of them. Furthermore, the group of protocols that provide existential untraceability is, inherently, a subset of those providing universal untraceability. That would suggest that universal untraceability must be, in general, the minimum level of untraceability a protocol designer aims to achieve. For instance, the designers of the $C^2$ scheme [13], as we have already noted, recognise that they had universal untraceability guaranteed, but they decided to further add existential untraceability at the cost of making the protocol non-scalable.

9. **Most protocols in the tables provide desynchronisation resistance.** The importance of this property for any RFID security protocol is supported by this observation. The consequences of a successful desynchronisation attack might include the system no longer being able to identify the tag
5. CLASSIFICATION OF RFID SECURITY PROTOCOLS

involved. See section 4.2.2, where the security analysis of the Henrici and Muller protocol is conducted, for an example of this.

The protocols in the tables can be subdivided into the following groups, according to the way desynchronisation is addressed:

(a) The database stores two pseudonyms per tag, current and old. If desynchronisation occurs, the database can locate the correct tag entry by the old pseudonym in the next authentication session. As examples of protocols using this technique successfully, the Alomair et al protocol [12] and our modified version can be named.

(b) The database stores the current pseudonym only, and if needed, the next one is calculated on-the-fly. In this case, if desynchronisation occurs, in the next authentication session the database can locate the correct tag entry by calculating the next pseudonym to the current stored one. In this work, the $C^2$ scheme [13] has been identified as the only protocol using this technique.

(c) The database stores a hash chain for each tag. In protocols such as the Avoine et al improvement to OSK [58], the database stores a hash chain for each tag. This is inherent to the protocol family. The tag replies at each interrogation with the next element of the chain, and the system explores each chain up until a pre-set maximum. That implies desynchronisation resistance is limited to that maximum. Once that limit is reached, the tag becomes definitively desynchronised. Unfortunately, that limit can also be maliciously exhausted by an attacker. For further details, the interested reader is referred to appendix B.3.3.1, where the scheme is reviewed.

(d) Protocols where secrets are not refreshed. In general, these protocols are desynchronisation resistant because there is no need to maintain any synchronisation for the secrets, as they are static. Examples include the Rhee et al protocol [23] and the Molnar and Wagner protocol [26]. It is important to note that secrets might not be the only elements that a given protocol intends to maintain synchronised, as can be seen in the following point.

(e) Desynchronisation in the YA-TRAP family. Protocols such as RIP [77] and RIP+ [77] are easily desynchronised. Secrets are not refreshed, but the counters used are. For more information, the reader is referred to appendix B.3.3.3, where the YA-TRAP family is examined. A desynchronisation attack applicable to both protocols is succinctly described below:

i. It is relevant for the description of the attack that we observe that the tag receives a monotonically increasing counter from the system in the first message of the protocol. The tag expects the value of the received counter to be no greater than a pre-set maximum, but greater than the last valid received one, which is stored in the tag. Consequently, a given counter value can be used once only.

ii. In the first place, the attacker sends the tag a counter value which equals the maximum one.

iii. The tag accepts the value and updates its current counter value to the received one, i.e. to the pre-set maximum.

iv. As a result, the tag no longer accepts any other value from the system as valid, and thus becomes definitively desynchronised.
5.3 Summary

O-RAP \cite{77} does not behave in the same way as RIP and RIP+. The protocol can also be easily desynchronised, as an attacker just needs to interrogate the tag once. However, the tag re-synchronises at the next successful authentication. Therefore, in this work, it is assumed that desynchronisation resilience is provided. The interested reader is referred to appendix B.3.3.3, where the protocol is reviewed, for further details.

(f) Finally, some exceptions exist amongst the surveyed protocols. For instance, the Henrici and Muller protocol \cite{11} uses the first technique above in this ninth point on desynchronisation resistance, i.e. the database stores two pseudonyms per tag, current and old. However, it does not provide desynchronisation resistance. The reason for this is a weakness in the protocol, which makes it possible for an attacker to mount a desynchronisation attack. The attack is based on the manipulation of the last message from the system \cite{79}. The reader is referred to section 4.2.2, where a security analysis of the protocol is conducted, for further details and a description of the attack.

Additionally, for the reader’s convenience, a generic example has been prepared which features a possible application of the results of the analysis of the classification. Due to space constraints, the reader can find it as appendix D.1.2. The example concerns a major retail store. The organisation provides the requirements of the target application, and takes into account the results obtained in this section. They expect to obtain the main characteristics of the security protocol that they need. As the chapter can be understood without consulting that appendix, it can be considered supplementary material, and the reader can decide to skip it without loss of continuity.

The next section summarises and concludes the chapter.

5.3 Summary

The main aim of the chapter has been the classification of a number of security protocols in order to provide additional insights into the design of security protocols for low-cost RFID tags. The protocols selected and the properties they are classified against are specified in the introduction:

1. The protocols included have been selected in accordance with the criteria that is established in section 4.1. Our three case studies and their respective improved versions have been included. In addition, a number of other protocols from the literature review in chapter 3 have also been incorporated. According to our selection criteria, all classified protocols belong to section 3.3 on proposals using symmetric key cryptography.

2. Several classification properties have been chosen to characterise the protocols, including: some criteria to categorise them, security and privacy properties, and performance and implementation requirements on tag, together with communication costs.

For the sake of clarity, the classification of the protocols is presented in three tables in section 5.1. In section 5.2, the analysis of the classification is conducted. The results of the analysis are presented and discussion on them is carried out. Due to space restrictions, the analysis in the section is an extract
of the one that can be found in appendix D.1.1. Even though it is not necessary to read the appendix to understand the chapter, the interested reader is encouraged to take the time to do so.

Finally, in section 5.2 as well, an example showing the relevance of the conclusions drawn from the analysis of the classification is offered as appendix D.1.2. The example shows how the results of our analysis can be used, for instance, to derive the main traits of a protocol that is suitable for the set of requirements of a given scenario. At the same time, the example provides an environment where the results of the analysis can be further examined. This appendix, as already noted in section 5.2 where it is offered to the reader, can be understood as supplementary material and thus skipped without loss of continuity.

This section has summarised and finished the chapter. In the next one, an overview of a selection of tools for the automated formal verification of security protocols is given. In addition, an attempt is also made to provide suggestions for the improvement of those tools to better capture the requirements of security protocols for low-cost RFID tags.
Chapter 6

Tools for the Automated Formal Verification of Security Protocols

This chapter sets out to provide a number of suggestions for the improvement of the tools for the automated formal verification of security protocols. These suggestions are intended to improve the ability of these tools to better meet the requirements of security protocols for low-cost RFID tags.

In the first place, the need for the tools is explained. Then, some background on formal verification of security protocols is provided. These tools are: AVISPA/AVANTSSAR [6, 41], Proverif [42], Scyther [43], and Casper/FDR [44, 45, 46]. Once the necessary background has been obtained, suggestions are attempted for the improvement of the tools to better capture the requirements of security protocols for low-cost RFID tags.

It is important to highlight that most of the background material prepared for inclusion in this chapter is supplied as appendices. Space constraints have been the main reason for this, but also the fact that it is background information. Even though it should be possible to understand the chapter without resorting to the referenced appendices, the interested reader is encouraged to take the time to read this material. The main reason for this is that, in our view, it would enrich the understanding of the subsequent suggestions for the improvement of the tools.

In the next section, the need for the formal verification of security protocols is motivated.

6.1 Motivation for the Formal Verification of Security Protocols

In this section, the relevance of formal verification of security protocols is justified.

The material in this section is supplied as appendix E.1. As noted in the introduction to the chapter, there are two main reasons for this. The first one is that it background information, and the second one is space limitations. Nonetheless, the interested reader is encouraged to devote some time to reading this information. For the reader’s convenience, a short summary of the material in the appendix is provided below.

In the first place, the appendix discusses the current need for security protocols. It is noted that there is an increasing use of network infrastructures and associated network protocols. In addition, these network protocols often must guarantee that a set of required security properties be satisfied.

Then, the error-prone nature of the design of security protocols is pointed out, and two well-known examples are given of protocols which have been shown to be flawed after their publication. The first one is the Needham-Schroeder Public-Key Protocol [39], broken and fixed by Lowe using the tool Casper/FDR.
The second one is the Single Sign-On (SSO) protocol used in the SAML-based Single Sign On for Google Applications, shown to be flawed by Armando et al using the tool SATMC [91].

Following on from that, it is observed that the potential flaws in the design of security protocols might lead to undesirable consequences, which might include financial and public-relations damage to the companies involved.

As a result, it is concluded that formal verification techniques are needed. As was noted for the first time in the introduction to this project, in chapter 1, a protocol whose security has been formally proven, to some extent, offers a much greater degree of confidence, which is essential in applications characterised by security sensitivity.

The next section provides some background on formal verification of security protocols.

6.2 Background on Formal Verification of Security Protocols

In this section, some background is provided on formal verification of security protocols.

As was the case in the previous section, due to space restrictions and the fact that it is background information, the material in this section is supplied as an appendix, namely E.2. Once more, the interested reader is nevertheless encouraged to devote some time to reading this information. For the reader’s convenience, a short summary of the material in the appendix is also provided below.

In the first place, it is emphasised that the problem of protocol security is undecidable, in general, which poses a challenge to automated tools. It is also noted that to address this issue, restrictions can be applied. For instance, in [92], it is shown that protocol insecurity is NP-complete with a bounded number of sessions, regardless of the size of the messages.

Then formal methods are subdivided into three main groups, namely methods based on logics, model checking, and theorem proving.

To address the first group, namely methods based on logics, the most well-known and representative logic within it is studied, namely the BAN logic, named after their authors Burrows-Abadi-Needham [93]. Firstly, a description of its main characteristics is provided. Then both the fact that it can provide useful insights to the protocol designer and its limitations are briefly addressed.

With regard to the second group, model checking, it is observed that the main idea is the analysis of protocols through a state space model, where, typically, transitions represent actions of honest agents or the intruder, and states represent the knowledge of the intruder and the local state of honest agents. Ideally, the model checker would check that the property to be analysed holds in all states to ensure correctness. If the property did not hold in all states, a trace leading to the attack state would be produced. Nevertheless, the appendix then notes the fact that the state space can be very large, or even infinite, and a brief discussion follows on corresponding implications.

Finally, the third group is examined, namely, theorem proving. In order to describe theorem proving, a salient example of an interactive theorem prover is used, namely Isabelle [94]. In the first place, it is observed that protocols are modelled as a set of traces, and target security properties are modelled as theorems, which are proved by induction on traces. Then interactivity in theorem provers is briefly discussed. The fact that theorem provers in general, and Isabelle in particular, can address an unbounded number of sessions and
agents is also noticed. Finally, it is pointed out that the method focuses on proving correctness rather than finding attacks.

The next section offers an overview of four prominent tools for the automated formal verification of security protocols.

### 6.3 Overview of Surveyed Tools

In this section four prominent tools for the automated formal verification of security protocols are reviewed. The tools are AVISPA/AVANTSSAR [6, 41], Proverif [42], Scyther [43], and Casper/FDR [44, 45, 46].

Due to space limitations, the material in this section is supplied as appendix E.3. As noted in the introductory section to this chapter, it should not be necessary to read the material in the appendix to understand the suggestions for the improvement of the tools provided in the next section. However, it is our opinion that the reader would benefit from this supplementary information.

For the reader’s convenience, a short summary of the material in appendix E.3 is provided below:

1. **The AVISPA Tool** is studied in appendix E.3.1.1. Its architecture is examined, including a description of its four backends: namely, the On-the-Fly Model-Checker (OFMC, for short), the Constraint-Logic-based Attack Searcher (CL-AtSe, for short), the SAT-based Model-Checker (SATMC, for short) and the Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP, for short). The operation of its web interface is also explained. In addition, an introduction to HLPSL, its language to model security protocols, is provided.

2. **The AVANTSSAR Platform**, the successor to the AVISPA Tool, is reviewed in appendix E.3.1.2. Its three backends are identified as new versions of the corresponding ones already found in AVISPA. In addition, four different application-level languages, available to model security protocols, are also noted. All of them are translated into the common ASLan language, which is input into the backends. Then, the architecture of the platform is described. Finally, the appendix provides a short explanation of the two forms in which the platform is offered, namely as a SOA (short for service-oriented architecture) and as a web interface.

3. **Proverif** is examined in appendix E.3.2. The essential characteristics of the tool are reviewed. In addition, its architecture is also described. Then, the type of properties that the tool verifies is addressed, namely reachability properties, correspondence assertions and observational equivalence. Finally, it is noted that the tool is offered as a web interface, and as a package which can be downloaded and installed on the user’s machine.

4. **Scyther** is reviewed in appendix E.3.3. In the first place, the main characteristics of the tool are surveyed. For instance, it is noted that secrecy and authentication can be analysed, direct support for equational theories is not offered, and support for adversarial models other than the Dolev-Yao is provided, to name but a few. Finally, the graphical user interface of the tool is also described.

5. **Casper/FDR** is examined in appendix E.3.4. Firstly, the subsection elaborates on the FDR tool, the CSP process algebra and the Casper compiler. Once that preliminary description has been made, the input file to Casper is described in some detail. Finally, some other features of Casper/FDR are
6. TOOLS FOR THE AUTOMATED FORMAL VERIFICATION OF SECURITY PROTOCOLS

identified and explained. For example, it is noted that support for flow control is limited to test-lines, which allows the modelling of the possible tests conducted by the agents participating in the protocol.

In the next section, suggestions for the improvement of the tools to better meet the requirements of security protocols for low-cost RFID tags are provided.

6.4 Suggestions for the Improvement of the Tools for the RFID Environment

In this section, suggestions for the improvement of these tools are provided, in order to better capture the requirements of security protocols for low-cost RFID tags. Each suggestion is structured in two parts. The first part evaluates the need for its inclusion, and the second examines its current coverage in each of the surveyed tools. Our suggestions are listed below. Also see table 6.1 on page 65:

1. **First suggestion: Many RFID security protocols would benefit from exclusive-or support:**

   (a) *The need for the inclusion of the suggestion:* In chapter 3, where the literature review on RFID security protocols is addressed, it is shown that exclusive-or is a widely used operator amongst this type of protocols, e.g. [11, 12, 26, 58, 73].

   (b) *Current coverage of the suggestion in each one of the surveyed tools:* Some of the available tools do not provide support for exclusive-or. The SATMC tool, one of the AVISPA Tool/AVANTSSAR Platform backends is one example. Even though the SAT-based approach is not suitable for algebraic properties, further investigation might solve this issue [95]. As another example, Scyther does not support exclusive-or either [96]. The interested reader is referred to appendix E.3.3 on Scyther for more details. However, its support is considered future work by the author of the tool in [97]. Nevertheless, the current situation is different for most other surveyed tools:

   i. OFMC and CL-AtSe, two of the backends of the AVISPA Tool/AVANTSSAR Platform [96] provide support.

   ii. TA4SP, one of the backends of AVISPA, provides support from a new version announced on the web page of the author [98].

   iii. Proverif provides support after the work in [99], which reduces protocols using XOR to their XOR-free equivalent.

   iv. Casper considers exclusive-or for Vernam encryption [100, 101, 102].

2. **Second suggestion: Even though it is not a pressing issue at the moment, many RFID security protocols might benefit from support for Diffie-Hellman exponentiation:**

   (a) *The need for the inclusion of the suggestion:* As discussed in section 3.1, where protocol proposals based on public key cryptography are reviewed, public key cryptography has been considered expensive for implementation on low-cost RFID tags [58]. Nevertheless, work is ongoing to reduce that cost [103]. For that reason, support for DH exponentiation might prove advantageous, but it is not a pressing issue at the moment.
6.4 Suggestions for the Improvement of the Tools for the RFID Environment

(b) Current coverage of the suggestion in each one of the surveyed tools: Amongst the surveyed tools, it has been observed that:

i. OFMC and CL-AtSe, two of the backends of the AVISPA Tool/AVANTSSAR Platform provide support [96].

ii. SATMC, one of the backends of AVISPA Tool/AVANTSSAR Platform does not provide support. As was observed above when discussing the support for exclusive-or, the SAT-based approach is not suitable for expressing algebraic properties, but further investigation might solve the issue [95].

iii. TA4SP, one of the backends of AVISPA Tool, provides support from a new version announced on the web page of the author [98].

iv. Proverif provides support after the work in [99], which reduces protocols using DH exponentiation to their DH-free equivalent.

v. Scyther does not directly support the modelling of Diffie-Hellman exponentiation [8]. In [97], the author of the tool mentions that support is considered as future work. The interested reader is referred to appendix E.3.3 on Scyther for more details.

vi. To the best of our knowledge, the current distribution of Casper does not support exponentiation [96, 101]. See appendix E.3.4 on Casper/FDR for more details.

3. Third suggestion: It would be advantageous to provide support for communication channels/adversary models other than the Dolev-Yao model [104]:

(a) The need for the inclusion of the suggestion: The reader is reminded that an intruder under the Dolev-Yao model is assumed to be in full control of the network and is only limited by the perfect cryptography assumption. Restriction on or addition to the capabilities of the attacker can prove useful when considering security protocols for low-cost RFID tags. As noted in [82], in the RFID security context, the intruder is modelled differently depending on the publications. For instance:

i. To serve as an example, minimalist cryptography can be considered [61]. See section 3.2.4, where it is reviewed, for further details. The main proposal assumes a weak adversarial model. The intruder, for instance, is active, but the number of times that he can interact with a tag between two successful interactions with a legitimate verifier is limited. Furthermore, some simpler variants of the main proposal are also presented. They require fewer resources on tag, and aim to protect only against passive eavesdropping.

ii. As another example, some authors present attacks which can be performed even by a passive adversary. For instance, the ultralightweight protocol presented by Eghdamian and Samsudin [65] was successfully attacked in [105]. It was shown that the attack could be carried out by a passive adversary. The Eghdamian and Samsudin protocol was reviewed in section 3.2.1.

iii. Finally, a third example involves an important characteristic of low-cost RFID tags, namely that they cannot afford tamper-resistance. That fact is noted in appendices A.1.5 and A.2.4, which the interested reader is referred to for more information. As a result of the lack of tamper-resistance, it is important to consider the possibility of an attacker compromising the contents of a tag, which include secret data. That adversarial capability goes beyond the
6. TOOLS FOR THE AUTOMATED FORMAL VERIFICATION OF SECURITY PROTOCOLS

Dolev-Yao model. Therefore, for those tools which do not currently allow the user to model that capability, an extension would be required in order to better meet the requirements of security protocols for low-cost RFID tags.

(b) Current coverage of the suggestion in each one of the surveyed tools: Amongst the surveyed tools, some steps have been taken in the direction of providing the user with the possibility of modelling different adversarial capabilities. These include:

i. The AVANTSSAR Platform allows the specification of channels other than the Dolev-Yao. Indeed, it is possible to specify confidential and authentic channels \[6\]. This fact is also noted in appendix E.3.1.2, where the Platform is reviewed.

ii. However, AVISPA supports Dolev-Yao-type channels only \[106\].

iii. With regard to Proverif, apart from public channels where the attacker can read and write, private channels where the adversary can neither read nor write can also be specified. In addition, it is also possible to model asymmetric channels, where the adversary can either read or write, but not both \[107\].

iv. Scyther provides support for adversary models other than the Dolev-Yao \[108\]. By way of example, the adversary’s ability to learn session keys can be modelled, as well as the adversary’s ability to learn the long-term keys of a given agent \[108\]. In Scyther, however, authenticated channels are not directly supported. Nevertheless, they can be approximated \[109\]. This fact is also noted in appendix E.3.3, where the tool is reviewed.

v. Casper makes it possible for the user to model channels other than the Dolev-Yao, such as confidential, for example. In addition, it is also possible to specify that the intruder cannot fake messages on the channel, for instance. These properties of the channel can be declared at message-level \[101\]. Furthermore, it is also possible to specify that all keys of a given type will be compromised \[101\]. The reader is referred to appendix E.3.4 on Casper/FDR for further details.

4. Fourth suggestion: Support for untraceability analysis is important for RFID security protocols:

(a) The need for the inclusion of the suggestion: One of the main factors hindering widespread deployment of RFID technology is the threat to privacy. This fact is also noted in appendix A.1.5, where these factors are addressed. In particular, the possibility of the tag holder being tracked in time and space is a main issue. In this appendix, it is also noted that the Big Brother concern provokes movements against RFID technology \[52\]. Furthermore, authorities recommend the incorporation of security and privacy into RFID systems \[53, 54\]. An illustrative example of the feasibility of traceability attacks can be found in \[110\], where the authors implement a system to track people wearing the Nike+iPod sport kit. Consequently, support for the automated formal verification of untraceability in security protocols for low-cost RFID tags is highly desirable.

(b) Current coverage of the suggestion in each one of the surveyed tools: Amongst the surveyed tools, it is observed that:
6.4 Suggestions for the Improvement of the Tools for the RFID Environment

i. The AVANTSSAR Platform does not offer support. Indeed, properties over multiple execution traces are not supported [111]. For instance, untraceability, which is related to the ability of an attacker to differentiate pairs of traces [112].

ii. The AVISPA Tool does not offer support either. In fact, previous versions of the same backends of AVANTSSAR are used. It is true AVISPA additionally uses the TA4SL backend, but it does not offer this capability either [113].

iii. Proverif does provide support. In order to model untraceability in the applied pi calculus, observational equivalence is used in [114], and Proverif can prove observational equivalences [107].

iv. Even though Scyther can analyse secrecy and authentication [8], it does not analyse observational equivalence [113].

v. Only secrecy and authentication properties are supported by Casper [101].

5. Fifth suggestion: An expressive language that supports flow control is required to model some RFID security protocols:

(a) \textit{The need for the inclusion of the suggestion:} Flow control includes such features as if-then-else or looping. Some protocols do not require them, but some others do. In chapter 3, where there is a literature review on RFID security protocols, some of the RFID security protocols examined do not require flow control constructs, e.g. [23, 58, 73]. However, some others do. RIP [77] can serve as an example. The interested reader is referred to appendix B.3.3.3, where the YA-TRAP family is examined, for further details. The tag receives a counter \(t_{sys}\) from the system. Then, it checks whether the counter satisfies \(t_{tag} < t_{sys} < t_{max}\). If it does, the tag updates \(t_{tag}\) to \(t_{sys}\) and \(H_{k_{tag}}(t_{sys})\) is sent to the system. Otherwise, the tag generates and sends a random value to the system. Consequently, in order to model the protocol, the flow control if-then-else construct is needed.

(b) \textit{Current coverage of the suggestion in each one of the surveyed tools:} Amongst the tools surveyed, it has been observed that:

i. The AVANTSSAR Platform directly supports the modelling of such constructs as loops and branches. For instance, they can be found amongst ASLan++ statements [111].

ii. The AVISPA Tool provides limited support to flow control. It is possible to deal with some protocols which have conditional steps [106]. For example, in the Needham-Schroeder Public Key - Key Server protocol, NSPK-KS for short [39], participants need to ask the server for the public key of their counterpart if they do not already know it. In order to model that conditional step, two transitions can be defined as follows: one of them is executed when the participant does not know the public key of their counterpart, and the other one when it does.

iii. In Proverif, support for flow control is limited. It is possible to model conditionals as there is direct support for an if-then-else statement [107].

iv. Protocols that feature branching or loops cannot be modelled in Scyther, as support is not provided for these features [97, 113, 115]. Consequently, no support for flow control is offered.
v. Support for flow control in Casper is limited. The only mechanism provided is the test-line that allows the modelling of the possible tests conducted by the agents. This fact is also noted in appendix E.3.4, where the tool is reviewed and which the reader is referred to for further details.

In this section, we have provided suggestions for the improvement of these tools in order to better capture the requirements of security protocols for low-cost RFID tags. The next section summarises and concludes the chapter.

6.5 Summary

The main aim of the chapter is the provision of a series of suggestions for the improvement of the tools for the automated formal verification of security protocols. These improvements are intended to better capture the requirements of security protocols for low-cost RFID tags.

The chapter starts in section 6.1, where the need for formal verification techniques is justified. Then, section 6.2 provides some background on formal verification of security protocols. After that, section 6.3 reviews a selection of four salient tools. These tools are: AVISPA/AVANTSSAR [6, 41], Proverif [42], Scyther [43], and Casper/FDR [44, 45, 46].

Once the review of the tools has been conducted, the aforementioned suggestions are made in section 6.4. Each suggestion consists of two main components: firstly, an evaluation of the need for its inclusion in the tools, and secondly, an examination of the current coverage in each one of the surveyed tools.

This section summarises and concludes the chapter. In the next chapter, the conclusions of the project are drawn.
Table 6.1: Suggested Features for the Improvement of the Tools - Current Coverage - Comparison Table

<table>
<thead>
<tr>
<th>Feature</th>
<th>AVISPA</th>
<th>AVANTSSAR</th>
<th>Proverif</th>
<th>Scyther</th>
<th>Casper/FDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for XOR</td>
<td>Yes1,2,3</td>
<td>Yes1,2</td>
<td>Yes4</td>
<td>No11</td>
<td>For Vernam encryption2</td>
</tr>
<tr>
<td>Support for exponentiation</td>
<td>Yes1,2,3</td>
<td>Yes1,2</td>
<td>Yes4</td>
<td>No11</td>
<td>No19</td>
</tr>
<tr>
<td>Adversary models supported</td>
<td>DY only5</td>
<td>DY and others6</td>
<td>DY and others4</td>
<td>DY and others21</td>
<td></td>
</tr>
<tr>
<td>Support for untraceability</td>
<td>Yes7</td>
<td>Yes7</td>
<td>Yes9</td>
<td>No10</td>
<td>No17</td>
</tr>
<tr>
<td>Support for flow control</td>
<td>Limited11</td>
<td>Yes14</td>
<td>Limited15</td>
<td>No16</td>
<td>Limited20</td>
</tr>
</tbody>
</table>

1 Both the OFMC and CL-Atse backends of the AVISPA Tool/AVANTSSAR Platform support the verification of protocols using the algebraic properties of XOR and exponentiation [96].
2 The SATMC backend of the AVISPA Tool/AVANTSSAR Platform does not offer support for XOR, and it does not for exponentiation either. The reason for this is that the SAT based approach is not suitable for expressing algebraic properties. This aspect requires further investigation [95].
3 The TA4SP backend of the AVISPA Tool offers support for XOR and exponentiation from a new version announced on the web page of the author [98].
4 Support provided after the work in [99, 116], which reduces protocols using XOR/DH to their XOR/DH-free equivalent.
5 The Dolev-Yao intruder [104] is the only one supported by the AVISPA Tool [106].
6 The Platform allows the protocol designer to specify communication channels other than Dolev-Yao [6].
7 The AVANTSSAR Platform does not offer support. Indeed, properties over multiple execution traces are not supported [111]. For instance, untraceability, which is related to the ability of an attacker to differentiate pairs of traces [112]. The AVISPA Tool does not support properties over multiple execution traces either. In fact, it uses previous versions of the same backends of AVANTSSAR. In addition, AVISPA also uses TA4SL, but it does not offer this capability either [113].
8 Apart from public channels, it is also possible to model private channels and asymmetric channels [107].
9 Proverif does provide support. In order to model untraceability in the applied pi calculus, observational equivalence is used in [114], and Proverif can prove observational equivalences [107].
10 Even though Scyther can analyse secrecy and authentication [8], it does not analyse observational equivalence [113].
11 Scyther does not directly support equational theories [8]. These would allow the direct modelling of Diffie-Hellman exponentiation, for example. Furthermore, exclusive-or is not supported either [96], by the same token. In [97], the author of the tool mentions that support for both of these algebraic operators is considered as future work.
12 Scyther provides support for adversary models other than the Dolev-Yao [108]. In Scyther, though, authenticated channels are not directly supported. Nevertheless, they can be approximated [109].
13 In the AVISPA Tool, it is possible to deal implicitly with some protocols which have conditional steps [106].
14 The AVANTSSAR Platform directly supports the modelling of such constructs as loops and branches. For instance, they can be found amongst ASLan++ statements [111].
15 Support for control flow is limited. It is possible to model conditionals as there is direct support for an if-then-else statement [107].
16 Protocols that feature branching or loops cannot be modelled in Scyther, as support is not provided for these features [97, 113, 115].
17 Only secrecy and authentication properties are supported by Casper [101].
18 Casper considers XOR for Vernam encryption [100, 101, 102].
19 To the best of our knowledge, the current distribution of Casper does not support exponentiation [96, 101].
20 Support for flow control in Casper is limited. The only mechanism provided is the test-line that allows the modelling of the possible tests conducted by the agents [101].
21 Casper makes it possible for the user to model channels other than the Dolev-Yao. In addition, it is also possible to specify that all keys of a given type will be compromised [101].
Chapter 7

Conclusions

This thesis has focused on the topic of security protocols for low-cost RFID tags and their automated formal verification. This section, contains the conclusions deriving from this project. Firstly, our contribution and main results are identified and succinctly exposed. Then, the objectives the project set out to achieve are listed, and the extent to which we believe they have been fulfilled shown. Finally, pointers are attempted to the possible evolution of this subject area.

Our contribution and main results have been:

1. *An improved version of the Henrici and Muller protocol* [11]. The new version preserves an efficient constant-time identification, i.e. $O(1)$, mutual authentication, and anonymity. Moreover, it additionally provides universal and forward untraceability and desynchronisation resistance. To obtain these results, two improvements have been applied. The reader is referred to section 4.2.3 for further details.

2. *An improved version of the Alomair et al protocol* [12]. The improved version adds universal and forward untraceability to the protocol. Three improvements are applied with this aim in view. In addition, two further suggestions for the improvement of the protocol are made. These are oriented to improving the privacy level of the protocol while preserving its main structure, and are not part of the improved version. The reader is referred to section 4.3.3 for further information.

3. *An improved version of the $C^2$ scheme* [13]. The original version of this protocol offers excellent security and privacy properties. Unfortunately, this is achieved at the cost of a linear identification complexity, i.e. $O(N)$, which makes the protocol non-scalable. This case study, therefore, is a prime example of the security-performance tradeoff. Consequently, our suggestions focus on the improvement of performance and cost, and, at the same time, the preservation, as far as possible, of the original high degree of security and privacy.

   Our first proposal leads to the reduction of the gate count on tag. The second proposal is intended to reduce computation time on tag and communication cost. Our third and final proposal improves the identification complexity on the server’s side, which makes the scheme scalable, albeit at the cost of losing a certain degree of privacy. The reader is referred to section 4.4.3 for additional details.

4. *The classification of fifteen security protocols*: These protocols have been chosen from the literature review, and include our three case studies and their respective improved versions as well. The classification has been made against a number of security and privacy properties, and various performance and implementation parameters.

   The classification of the protocols itself is understood as a contribution. The reason for this is that
critical analysis has been conducted from the comparison. This analysis has provided helpful insights into the design techniques used by the protocols involved, collectively considered. Furthermore, these protocols are representative ones among proposals using symmetric-key cryptography, see section 3.1. Therefore, the conclusions drawn have the potential to be extended to this type of proposals as well. These conclusions, in the form of characteristic behavioural patterns, are, inherently, of a generic nature, and as a result, might have been observed in existing literature as well.

To serve as an illustrative example, it has been observed from the comparison that protocols that do not refresh tag secrets do not offer forward untraceability. Furthermore, the refreshment of tag secrets is not a sufficient condition to provide forward untraceability, it must also be performed using a one-way function. The remaining characteristics derived can be found in section 5.2, where they are presented and discussed. Due to space restrictions, the analysis in the section is an extract of the one that can be found in appendix D.1.1. Even though it is not necessary to read the appendix to understand the results of the analysis of the classification, the interested reader is encouraged to take the time to read it. In addition, a case study providing specific context for further examination of the results can also be found in appendix D.1.2. This appendix can be understood as supplementary material and thus skipped without any loss of continuity.

5. The presentation of five suggestions for the improvement of the tools for the automated formal verification of security protocols to better meet the requirements of low-cost RFID tags: For each suggestion, evidence is provided of its relevance for security protocols for low-cost RFID tags. In addition, the current coverage of the recommendation amongst the tools surveyed is explored. Moreover, a table is included which provides a succinct and to-the-point picture of all the suggestions presented. By way of example, the need for an expressive protocol specification language that supports flow control has been proposed. For a complete list of the suggestions provided, the reader is referred to section 6.4.

This project set out with the aim of attaining a number of objectives. These are summarised below for the reader’s convenience. Moreover, it is also indicated to what extent we believe that they have been fulfilled.

- As a first step and objective, it was considered essential to gain an all-encompassing perspective of the work that has been done in the area to date. To satisfy this objective, a literature review has been conducted on RFID security and privacy. It can be found on chapters 2 and 3. It includes an introduction to the technology, its applications and its security and privacy related issues. In addition, an overview of non-protocol proposals to RFID security and privacy has also been offered. Most importantly, a literature review has been also provided of protocol proposals, which are the focus of this work.

  It is our view this objective has been met to a great extent. Due to space constraints, most of the background material prepared to that effect has been delivered as appendices. The interested reader is encouraged to take the time to read them.

- The second objective was the identification of security properties which are significant to the RFID field. These properties can be found in section 2.2, and we believe that they adequately cover the security and privacy requirements of the RFID field.
7. CONCLUSIONS

• Our third objective concerned the justified selection of three case studies; and our fourth objective involved their description, security analysis and provision of suggestions for their improvement/fixes. As already mentioned in this section on conclusions, three case studies have been selected, described, and analysed, and suggestions for their improvement/fixes made. This work can be found in chapter 4. In our view, the objectives have been met insofar as a new improved version has been proposed for each case study/protocol.

• The fifth objective was the classification of a number of security protocols for low-cost RFID tags by a series of security properties and performance parameters. This objective, in our opinion, has also been achieved to a reasonably high extent. In chapter 5, as mentioned above in this conclusions section, fifteen security protocols are classified and behavioural patterns are derived and critically analysed from their comparison. Nevertheless, it would have been possible to incorporate a higher number of protocols into the classification. It is our view, however, that fifteen representative ones from the literature review is a reasonable figure for scope of our project.

• Our sixth objective was the review of a selection of tools for the automated formal verification of security protocols. Four salient tools have been reviewed, namely AVISPA/AVANTSSAR [6, 41], Proverif [42], Scyther [43], and Casper/FDR [44, 45, 46]. Due to space limitations, this material has been provided as appendix E.3. Once more, even though it should be possible to understand this project report without reading the appendices, the reader is encouraged to devote some time to them. It is our view that this sixth objective has been met to a reasonable extent. It would have been possible to include some other tools in the analysis. However, we believe that those included are representative, and constitute a reasonable figure for the scope of this work.

• Our seventh and last objective was the provision of suggestions for the improvement of these tools to better capture the requirements of security protocols for low-cost RFID tags. This is our final objective, and it has been covered in section 6.4. Several suggestions have been made with this aim in view, (see section 6.4 for a complete list). As a result, it is our view this objective has also been fulfilled to a reasonable extent.

Finally, some pointers about the possible evolution of the field of security protocols for low-cost RFID tags and their automated formal verification are attempted:

1. Throughout our study, no protocol has been found that can be considered ideal. Research is ongoing to design protocols which offer an increasingly improved tradeoff between, on the one hand, security and privacy, and, on the other hand, performance parameters such as scalability and on tag requirements. In this project, a significant number of relevant and representative security protocols for low-cost RFID tags have been examined. Throughout the study, no protocol has been found that can be considered ideal. For instance, some protocols offer a high degree of security and privacy, but are not scalable, e.g. the $C^2$ scheme [13] or the PEPS protocol [75]. Some others are scalable, but their degree of security and privacy is not as high, e.g. the Avoine et al improvement to OSK [58] or the Molnar and Wagner protocol [26]. Research is ongoing to design protocols which offer an increasingly improved tradeoff between, on the one hand, security and privacy, and, on the other hand, performance parameters such
as scalability and requirements on tag.
In the meantime, there is a range of protocols at our disposal which might offer sound solutions for specific target applications. For instance, if scalability is not an issue, e.g. access control at a small to medium-sized organisation, the C² scheme [13] offers an excellent degree of security and privacy. Otherwise, if limited desynchronisation resistance is acceptable, the Avoine et al improvement to OSK [58], for example, offers a high degree of security and privacy and its scalability can be adjusted depending on the time-memory tradeoff used.

2. It is important to note that PKC would solve the scalability issue, but it has generally been considered expensive for low-cost tags [58]. Nevertheless, work is also in progress which might eventually reduce that cost [103].

3. With regard to the automated formal verification of security protocols, it is our view that there is sufficient evidence to anticipate the progressive improvement of these tools to better meet the requirements of security protocols for low-cost RFID tags. The justification for this statement is based on two main reasons.

The first one is that there is an interest from the authors of the tools to incorporate additional features into them, in order to provide support for a wider range of protocols and associated claimed security properties that can be modelled and analysed. For instance, the author of the Casper compiler explicitly states his interest to receive requests for further features [101]. Some other authors explicit as future work lists of features that they want to include, see [97] for Scyther, for example.

The second reason is that there is a need for confidence in the security of protocols for low-cost RFID tags, and a protocol whose security has, to some extent, been formally proven, offers a much greater degree of that confidence. Indeed, in the field of RFID technology in general, security and privacy are under the spotlight. This can be observed in movements against the technology [52], and in recommendations to incorporate security and privacy into RFID systems as well [53, 54]. In addition, security protocols are an important mechanism to address such concerns.

It is our view that the combination of both reasons strongly points to the adoption of such characteristics as those suggested in this work. Therefore, we anticipate the progressive improvement of these tools to better meet the requirements of security protocols for low-cost RFID tags.

4. Finally, as a last pointer to the future of the field with regards to the automated formal verification of security protocols, we have chosen to make a prediction, namely the possible introduction of a standardised protocol specification language. This language could be then translated into the language that any existing tool expected. The particular language of each tool could remain in its current form.

It has been observed in this work that each tool has its strengths and weaknesses. If the protocol designer desires to take advantage of the strengths of a set of tools, a different protocol specification language must be used for each of them, which is not an easy task. It is our view that a standardised high-level language would remove that entry barrier for the user. Once the specification was written in this language, any tool could analyse it, covering a wider range of protocol characteristics and security properties. Those tools which were not competent with regard to some functionality could return an inconclusive result for that part of the requested verification.
7. CONCLUSIONS

This section has been concerned with drawing the conclusions of this work. Our contribution and main results have been identified and summarised. The objectives that this project set out to achieve listed, and the degree to which we believe that they have been fulfilled assessed. Finally, we have attempted some considerations concerning the possible evolution of the subject area.
Bibliography


3, 5, 57, 59, 64, 68, 146, 150


75


76


BIBLIOGRAPHY


[136] EPCglobal. GS1 EPC Tag Data Standard 1.7. May 2013. 99


Appendices
Appendix A

Background on RFID Security and Privacy

A.1 Preliminary Concepts

A.1.1 Architectural Models for RFID

In this thesis, we will work under the assumption that a traditional RFID model is used, which will be composed of a set of readers, a much bigger set of tags, and a backend database server. The backend database server is assumed to be trusted and reliable, unless explicitly stated otherwise.

This model has captured most of the research work to date, see [17] for an online repository. Indeed, a large share of real-world applications are built using it, as we will see in subsection A.1.4. However, other models exist, and we briefly mention them here:

1. Some research has been made on scenarios where there is a need to identify tags offline. Essentially, proposals try to download sufficient data from database to readers so that these themselves can authenticate tags, while ensuring that a compromised reader will not allow adversaries to clone tags. We will not pursue this model any more in this report, but the interested reader is referred to [117] for additional treatment and a particular proposal.

2. There is also ongoing investigation on cloud computing architectures for RFID. The central idea is to take advantage of the emerging cloud-computing paradigm to address the scalability issues of RFID. Coupled with the advent of cloud-computing, new privacy and security issues arise, as cloud services providers cannot, in general, be assigned the same level of trust as the corporate in-house backend.

We will not be concerned with this line of work either, but we refer the interested reader to [118] for a proposed architecture framework. Security solutions in this research field are typically based on storing corporate information encrypted in the cloud server, performing searches over encrypted data (SoED), and adding some method to preserve privacy. See [119] for two secure and efficient SoED schemes. To preserve privacy, a well-known method in the field is the use of hash-chains, e.g. [89], and more recently the idea of the use of co-privacy [120] has been proposed in [121].
A. BACKGROUND ON RFID SECURITY AND PRIVACY

Having established the operational model we will work with, the next subsection will describe the components of the traditional RFID architectural model.

A.1.2 Components of the Traditional RFID Architectural Model

In section 1.1 the traditional RFID architectural model was succinctly presented. We now describe its components in some more detail in this subsection:

1. **A large set of resource-constraint tags.**
   For us, an RFID tag, in its simplest form, is a small and relatively cheap data-carrying device which has the capability to transfer its data by radio frequency to a nearby reader. It is made up of a tiny chip and an antenna. See figure A.1 on page 92 for an example tag anatomy and form factors.
   Tags can be grouped into either active or passive. Active tags have their own source of power, whereas passive tags extract their energy from the magnetic field created by the reader. This means a passive tag is inactive while outside the reader’s locality. In this work, from now on, we will be interested in passive tags only. We are concerned with low cost tags, and active tags are relatively expensive.
   The tag will receive data from the reader via the reader’s magnetic field, and will send responses back using that same field. The two typical techniques used for the tag to respond to the reader are load modulation for near-field coupling and back scattering for far-field coupling. Further explanation of the workings of these techniques is outside the scope of this work, but the interested reader is referred to [1] for an in-depth treatment.
   We note that passive RFID tags have no internal clock. They become inactive when a reader is not powering them. Therefore, they have no internal notion of time.
   Finally, it is worth mentioning the definition of an RFID tag has been very abused and some confusion exists about it [122]. We can safely state, though, that different RF devices can be sorted by their sophistication:
   (a) Towards the sophisticated end of the spectrum we find such RF devices as general contactless smart cards.
   (b) At the other end, we find 1-bit transponders (e.g., electronic anti-theft devices, EAS for short).

In this work we are interested in low-cost resource-constraint passive RFID tags providing some computational capability. Consequently, neither RF devices such as general contactless smart cards nor 1-bit transponders are the object of our study.

2. **A computationally powerful backend system.**
   There are many examples of such servers and include a PC or a robot control system [1]. The backend server will store information (both secret and non-secret) for all tags.

3. **A set of computationally powerful readers.**
   Readers can be stationary or mobile devices. Their essential function will be requesting information from a tag and forwarding the responses of the tag to the backend database server. In addition,
messages from the server will also be delivered to the tag via the reader. Readers are made up of three main functional blocks: control unit, which manages communications and computing capabilities; a transmitter and a receiver. In addition, a coupling element to the transponder is also found, and, in many cases, an interface to a backend system [1]. Finally, it must be noted a reader can read hundreds of tags per second [123].

4. A communication channel between backend server and readers.
   This channel, in the traditional RFID model that we are concerned with in this work, is considered secure, because both readers and backend servers, being powerful devices, are assumed to be able to afford the use of strong security mechanisms to protect data both at rest and in transit.

5. A communication channel between reader and tags.
   This wireless channel is assumed to be insecure. Any reader within reading range of a tag can interrogate it. In addition, in the most basic scenario, the tag replies with its static identifier. Moreover, it is also possible that the tag holder is unaware of clandestine scanning. Furthermore, we assume a Dolev and Yao adversary as described in [104]. They have full control of the network, both from a passive (eavesdropping) and an active perspective (including injection and blocking of messages). However they are limited by the perfect cryptography assumption, that is, they will not be able to break cryptography, unless given the corresponding keys. It is also important to note that readers transmit using much higher power than tags. As a result, it is easier to eavesdrop on the reader-to-tag channel than the tag-to-reader channel. However, we will not make such distinction in this work, and consider both of them equally vulnerable. The reason is any adversary equipped with a reader with a powerful antenna and/or high power output can gain access to the tag-to-reader channel from distances many times the nominal range, i.e. the maximum distance at which an ordinary reader operating according to a standard or product specification can scan a tag [3].

In the next subsection, we briefly mention RFID standards, and point out the main characteristics of one of them, namely, the EPCglobal Class-1 Gen-2 standard [34].

A.1.3 EPCglobal Class-1 Gen-2 standard

We will now turn our attention to standards. Several standards have been developed to try and ensure interoperability and baseline functionality for different RFID applications (such as animal identification and item management) and frequencies (low, high and ultra high). Further detail is outside the scope of this work. Nonetheless, we refer the interested reader to the websites of both the International Organization for Standardisation (ISO) [124] and EPCglobal [35] where they can be obtained.

We note that the type of tags we are interested in are well exemplified by the EPCglobal Class-1 Gen-2 standard (EPC-C1 G2, for short) [34]. It is worth pointing out the main characteristics of Class-1 Gen-2 tags. They:
A. BACKGROUND ON RFID SECURITY AND PRIVACY

Figure A.1: Example tag anatomy and form factors. Pictures taken from [2].
A.1 Preliminary Concepts

1. Are passive, i.e. they obtain power from the reader.
2. Contain the EPC, a unique identifier.
3. Provide Cyclic Redundancy Code (CRC) and Pseudo-Random Number Generator (PRNG) capabilities.
4. Offer security limited to certain sensitive operations only, such as kill and access (to disable the tag and to read/update memory, respectively). The following mechanism is used: the reader provides a password before being granted the right to carry out the operation.

Unfortunately, the security of Class-1 Gen-2 tags is very weak. An adversary can easily eavesdrop the password sent to the tag, as it just XORed with a key the tag previously sent to the reader in plaintext in the same authentication session. As we mentioned in subsection A.1.2 when introducing the components of the traditional RFID architectural model, and the channel between reader and tag in particular, the fact that the key was provided via the tag-to-reader channel does not prevent eavesdropping by an attacker with equipment not particularly difficult to obtain.

In the next subsection, salient characteristics of RFID will be identified, together with example applications where the technology has already been deployed.

A.1.4 Applications of RFID

In the first place, we will describe why RFID is functionally very attractive. Salient characteristics include:

1. The identifier associated to an object is unique. Tags are attached to objects, and the identifier of the object, which is stored in the tag, is unique. This is in contrast to barcodes, for instance, which identify the type of the object only.
2. It requires neither line-of-sight nor preestablished positioning, which again provides an advantage over other Auto-ID technologies.
3. A reader can read hundreds of tags concurrently. This can be advantageous to speed up identification in many applications, such as retail stores, where clients queue up at the checkout at peak times.
4. Tags have computing capability, even if constraint. This allows the possibility of the implementation of functional mechanisms, including security ones. In particular, read/write operations can be made.
5. It is also noteworthy that RFID tags are very small. For instance, the Hitachi Chemical Ultra Small Package UHF RFID tag is only 2.5 by 2.5 by 0.4 mm, including a built-in antenna. The product has also been reviewed by RFID Network.
6. We finally note RFID tags are potentially reusable. In those applications where there is a business case to have tags which are not disposable, such as libraries, there is an important implication for security. The tag can be a bit more expensive, and thus potentially include more complex security functionality.
Due to its undeniable advantages, **RFID systems have already been deployed in many real-world applications**, such as:

1. **Large retail stores** [14],
2. **Drug identification** [15],
3. **Pet identification** [16],
4. **Car immobilizer systems** [127], and
5. **Human beings** [128]. This case is a good illustration of the obvious privacy and ethical concerns generated by the technology. The reader is referred to [129], where a suggestion by VeriChip Corp. to implant tags into immigrants to track their every move is highlighted. The proposal is also emphasised on the Spychips.com website [52] as a press release [130].

The identification of factors hindering widespread deployment of RFID technology is the aim of the next subsection.

### A.1.5 Factors Hindering Widespread Deployment

Despite its unquestionable functional advantages, **RFID has not been widely deployed yet.** Several factors have contributed to this fact, including:

1. **The cost of the tags.** According to RFID Journal, a passive 96-bit EPC inlay (chip and antenna mounted on a substrate) costs from 7 to 15 U.S. cents [31]. EPCglobal’s aim is to see this price drop to the 5 cents figure in order to make massive deployment possible [32]. The interested reader is also referred to [33], where the five-cent target figure was also speculated as possible, through a combination of synergies in the system and state-of-the-art technologies in each and every component.

2. **Privacy threats:** RFID tags can be quite small and thus go unnoticed [125]. They are all-pervasive and respond to any reader that interrogates them. Furthermore, in the absence of proper security mechanisms, the response consists of a static identifier. This poses privacy threats, which should also be addressed before large scale deployment of the technology. See also figure A.2 on page 95. We distinguish:
   (a) **Information leakage**, where the tag discloses information about the object it is attached to, which can be potentially sensitive, such as a drug name that reveals an illness of the tag holder.
   (b) **Location tracking**, where the owner of the tag is traced in time and space by the tags they carry.
   (c) **Disclosure of information on past transactions**, where compromise of a tag, i.e. exposure of the contents of its memory, allows an adversary to obtain information on past transactions involving the tag. This can be an important threat in some applications, such as tagged books in a library. It should not be possible for an adversary to learn the list of people who borrowed the book
previously. Some books could point to an illness or the political stance of the people who read the book.

It must be noted at this point that low-cost RFID tags, being inexpensive devices, are not tamper resistant. A survey of possible physical attacks and countermeasures is outside the scope of this work. Nevertheless, we refer the interested reader to the wealth of work on this topic for smart cards, which also applies to low-cost RFID tags. [122, 131] are excellent resources.

There are some examples of movements against the use of RFID based on the resulting Big Brother concern. A case in point is the Spychips.com website [52], where a great deal of press releases and other material opposing the technology can be found.

Authorities have also reacted to this problem and have issued corresponding recommendations to incorporate security and privacy mechanisms into RFID systems, e.g. [53] in Europe or [54] the US.

3. Tag cloning: Tag cloning naturally leads to counterfeiting. This is also an important issue which should be dealt with if RFID is to gain widespread adoption. The negative consequences of counterfeited products entering the supply chain are varied and include risks to consumers (e.g. fake drugs), revenue losses for corporations, and reduction of tax income for governments. For further analysis of the effects of counterfeiting the interested reader is referred to [132].

One way to achieve tag cloning can be by taking advantage of the lack of tamper-resistance in low-cost tags. For instance, the adversary physically compromises a tag and then creates a clone which is attached to a counterfeited product. Another approach consists in scanning a tag lacking application
A. BACKGROUND ON RFID SECURITY AND PRIVACY

level security and copying its identifier into another tag which will from then on reply as if it were the genuine tag.

The next subsection briefly notes the important consideration that RFID privacy must take into account all layers of the RFID communication model.

A.1.6 RFID Privacy is a Multilayered Issue

In this subsection we highlight that privacy must be enforced at every layer of the RFID communication model. We introduce the different layers and provide examples of how privacy can be breached at each one of them.

The OSI model [133] established a multilayered structure for computer networks. The same approach can be applied to RFID communication. This model allows us to reason that RFID privacy must be enforced at every layer if it is to provide effective protection.

In the literature, there are different proposals to define the layers of the RFID communication model. Two well-known models are the one by Avoine et al [48] and the one by Deursen [49]. Both of them define three layers: physical, communication and application. However, they are not equivalent. A level by level comparison between Deursen’s model and Avoine et al’s model follows, see the table below as well:

1. The physical level in Deursen’s model includes both the physical and the communication levels in Avoine et al’s model.
2. The communication level in Deursen’s model is the application level in Avoine et al’s model.
3. The application level in Deursen’s model is not considered in Avoine et al’s model.

Table A.1: The Multilayered Nature of RFID Privacy - Comparison between Deursen and Avoine et al Models

<table>
<thead>
<tr>
<th>DEURSEN</th>
<th>AVOINE ET AL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Communication</td>
<td>Application</td>
</tr>
<tr>
<td>Application</td>
<td>Not considered</td>
</tr>
</tbody>
</table>

We briefly describe an example of how privacy can be breached at each and every layer of the model:

1. Physical layer, as defined in Avoine et al’s model. Here we find the physical air interface. To serve as an example, traceability can be breached by radio fingerprinting, which would distinguish tags from different manufacturers. As shown in [134] the distinction can be made even for tags from the same manufacturer and type with high probability, over 95%.

2. Communication layer, as defined in Avoine et al’s model. We find collision avoidance protocols at this layer. Their objective is to manage multi-access, from tags to reader, without interference. The multi-access to a reader occurs because when a reader broadcasts a request, all tags within the reader’s
A.1 Preliminary Concepts

locality reply at the same time [1]. Anti-collision protocols which involve static identifiers for tags, as noted in [49], allow an attacker to trace a tag. The attacker knows the same tag is present when they observe the same identifier.

3. Application layer, as defined in Avoine et al’s model. In this layer, we find application protocols, mostly cryptographic ones. It is essential that these protocols protect the information transmitted from the tag, typically an identifier. That identifier will allow the database to identify the tag and obtain related information. A wealth of application-layer protocol proposals have been made in the literature, see [17] for an online repository.

As an example of how privacy can be breached in this layer, we consider the Hash Lock Scheme [74]. The protocol consists in the tag replying the request of the reader by sending a hash of a static key. This hash is consequently static, which implies the tag can be traced by it.

To address the traceability problem of the Hash Lock Scheme, a randomised version is also offered in the same paper. The identifier returned is no longer static. However, the protocol is now less efficient, because it requires a linear complexity search at the backend system to identify a tag. Both the Hash Lock Scheme and its randomised version are examined in detail in section 3.3.1.

4. Application layer, as defined in Deursen’s model. Protocols at this layer implement functionality to recover, interpret and manage user-application information on tags. For instance, the current number of tickets on a public transportation system tag. It must be ensured those protocols do not leak any information which leads to a breach of privacy.

The RFID systems focus of this work, however, will be mainly concerned with identification of tags only. Therefore, they will not commonly implement application-layer protocols.

The interested reader is referred to [49] for a detailed description of an application-layer traceability attack on an electronic fare collection system, the e-go public transportation system. The attacker succeeds in isolating and then decoding transaction data stored on the tag. The data consists of the date and time of the last 5 transactions the owner made, which is a clear privacy breach.

As also noted in [49], and as an example concerning the type of RFID systems we will be interested in, it is also possible to combine information obtained at both the application and communication layers to breach privacy. The active attack against the HB+ protocol proposed in [135] is one such case. To perform it, information leakage from a legitimate verifier is used, namely, acceptance or rejection of the tag.

Finally, it must be highlighted even if a tag cannot be distinguished from any other of the same type at any layer, it can still be distinguished from tags of different types or manufacturers. Thus a tag holder can be traced by the set of tags it carries [48, 49]. Of course, the attack is probabilistic, as its success will depend on how common the set of tags held by the individual, or RFID profile, is.

An in-depth coverage of this topic is outside the scope of this work, and we refer the interested reader to the mentioned papers by Avoine et al [48] and Deursen [49] for more detailed treatment.
In this subsection we have made an important observation, namely that privacy protection must be enforced at every layer of the RFID communication model to be effective. From this point on, though, our work focuses on the application layer as defined in Avoine et al’s model, i.e. the communication layer in Deursen’s.

A.1.7 Summary

The main objective of this section A.1 has been to provide some essential preliminary background on RFID security and privacy so that the reader is well equipped to deal with the rest of our work.

We introduced the different architectural models for RFID, and highlighted we are concerned with the traditional one. Then, we described its components. A succinct mention to standards followed, and especially to the EPCglobal Class-1 Gen-2 standard, which exemplifies closely the type of tags we will be interested in. The section then offered a view of the range of applications of the technology and the different factors which, if not conveniently addressed, will hinder its forecasted widespread deployment. Finally, the important consideration of the multilayered nature of RFID privacy was pointed out.

A.2 Security Properties

A.2.1 Identification

Identification is a basic requirement for any RFID protocol. It will make it possible for the system to obtain the identity of a tag. However, in the absence of entity authentication, which we define in the following subsection, no corroboration of that identity is obtained.

The typical identification protocol consists of two messages. The first one is a request, from reader to tag. The second one is the identifier of the tag, from tag to reader.

Identification protocols are sufficient in many RFID applications which do not require further security assurances. One such example might be the identification of lost pets so that they can be returned to their owners. However, some other applications claim for stronger security, and we examine the corresponding properties in the following subsections.

A.2.2 Entity Authentication: Unilateral or Mutual

In this subsection, we define and describe both unilateral and mutual entity authentication.

Firstly, both definitions are given:

1. **Unilateral entity authentication** allows one of the parties in the protocol to have the assurance that the identity of the other party is as claimed. This assurance is only valid at the moment the protocol ends successfully.

2. **We say mutual entity authentication is satisfied when unilateral entity authentication holds in both directions**, i.e. both parties have assurance of each other’s identities at the moment the protocol ends successfully.
We understand that the definitions given above are sufficient for our purposes in this work. Nonetheless, the interested reader is referred to \cite{50} for excellent further treatment of the terminology and techniques involved in identification and entity authentication.

Entity authentication is essential in many RFID applications requiring proof of the authenticity of the objects the tags are attached to. A case in point are drugs, which can be the target of counterfeiting and thus cause serious health risks for consumers.

Having addressed identification and entity authentication, the following two subsections deal with the main aims of RFID privacy, namely the provision of anonymity and untraceability.

\subsection*{A.2.3 Anonymity}

Privacy receives, as we noted in the introduction to this section, a great deal of public attention when it comes to RFID technology. Within privacy, we describe anonymity and untraceability. The former is addressed in this subsection, and the latter in the next one.

Anonymity is sometimes referred to as the prevention of information leakage, in the RFID privacy context, e.g. \cite{48}. \cite{50} defines anonymity as an information security objective consisting on \textit{concealing the identity of an entity involved in some process}. For us, the aim, is effectively to make sure the tag does not leak any information which can lead to the revelation of some information related to its owner.

We will now give an example of the kind of threat anonymity is intended to protect against. Most RFID protocols are identification protocols only, as defined in subsection \ref{A.2.1}. Consequently, they reply with its static clear identifier to a reader’s request. In this scenario, the identifier might include information revealing the type of object it is attached to. For instance, the EPC general identifier format \cite{136}, apart from a serial number, contains some other fields, including an object class. It indicates the type of object the tag is attached to, which is assumed to be unique within a manufacturer. It can then reveal, for instance, that the object corresponds to a drug prescribed to treat a serious condition, which is sensitive information related to the tag owner.

One way to provide anonymity, i.e. prevent information leakage, in the RFID context, is to use identifiers which look random to the attacker. A widely used technique would be to use a cryptographic hash of the identifier, e.g. \cite{24}, so that it is computationally unfeasible for the attacker to derive the identifier from its hash, i.e. reverse the hashing operation.

In the next subsection, we deal with untraceability, which is the second main property to be considered when addressing RFID privacy.

\subsection*{A.2.4 Universal, Existential and Forward Untraceability}

In the previous subsection, we defined anonymity, and stated that it is one of the two main objectives of privacy, in the RFID context. Untraceability is the other one. Otherwise known as location privacy, untraceability is not a security property restricted to the RFID field, but can also be found in other areas such as mobile cellular systems. \cite{51} defines it as \textit{the ability to prevent other parties from learning one’s current or past location}.
We further said in the last subsection that one way to provide anonymity was that the tag answered with an identifier which looked random to the adversary, e.g. a hash of the real identifier. This solution, though, is not sufficient to provide untraceability. Indeed, regardless of whether the tag replies with the real identifier or a random version of it, e.g. hash or encryption, if the reply is static, an adversary will be able to easily link two observations of the same tag in time and space.

In this work, we will be interested in differentiating between two levels of untraceability. The terms universal and existential have been borrowed from [12]. We will say an RFID system provides:

1. **Universal untraceability:** when two responses from a given tag cannot be correlated if one of them occurs before a valid authentication session and the other one afterwards.

2. **Existential untraceability:** when two responses from a given tag cannot be correlated irrespective of whether a valid authentication session has occurred between them.

The fact that we have these two definitions at our disposal is advantageous. The reason is we can classify protocols as a function of the type of untraceability they meet, if any. For instance, those protocols where the identifier in the tag is updated as a function of some data provided by the reader, will be traceable between valid authentications and thus provide universal untraceability, at most, but not existential untraceability. For instance, in [137], the tag updates its identifier after it checks the last message of the reader only. As a result the tag can be traced between valid authentication sessions, i.e. it does not provide existential untraceability.

As noted in subsection A.1.5, low-cost RFID tags are not tamper-resistant, i.e. their contents, including secret data, can be obtained by physical means. The fact that tags can be compromised is well reflected in the definition of forward untraceability [12]. We say an RFID system provides forward untraceability when we cannot correlate the responses from a certain tag before the last valid authentication session, given that the tag has been compromised afterwards.

For instance, the Ohkubo, Suzuki, and Kinoshita’s protocol (OSK, for short) [25] provides forward untraceability. We will now describe the technique. Let us call ID the identifier on the tag at a certain time. At a given protocol run, two different cryptographic hash functions, H and G, are applied to ID, yielding H(ID) and G(ID). Then, the ID on the tag will be updated to H(ID), and G(ID) will be sent to the reader. In this way, if the tag is compromised and the adversary obtains ID, they are not able to infer the contents of any previous message from tag to reader. As a result, forward untraceability is achieved.

In this subsection, we have described untraceability, and in the next one we approach desynchronisation resistance. We will also examine the interrelationship between them.

### A.2.5 Desynchronisation Resistance

Some protocols assume synchronisation between tags and backend server database, based on the maintenance of consistent values on both sides, e.g. the last value of the tag identifier. We say tag and backend server have been desynchronised when an attacker succeeds in disrupting that consistency.

A formal definition for a desynchronisation is provided in [13]:

---

**Universal untraceability:** when two responses from a given tag cannot be correlated if one of them occurs before a valid authentication session and the other one afterwards.

**Existential untraceability:** when two responses from a given tag cannot be correlated irrespective of whether a valid authentication session has occurred between them.
A.2 Security Properties

1. Considering:

(a) In normal operation, the system and the tag maintain a key $K$ in synch, i.e. the key stored in the tag $TK$ equals the key stored in the system $SK$.

(b) We denote $K^i$ the value of $K$ after it has been updated $i$ times.

(c) In case tag and system are desynchronised, we have $TK = K^i$ and $SK = K^j$, with $i \neq j$.

2. We define a desynchronisation as an operation inducing an incrementation of the value $|i-j|$ by 1. The value $|i-j| \in \mathbb{N}$ is called number of desynchronisations.

For instance, a successful desynchronisation attack can lead to the backend server being unable to identify the tag any longer. This effectively amounts to a denial-of-service attack, as the availability of the functionality of the RFID system is affected [1].

An example of a desynchronisation attack is provided by the Heinrici and Muller’s protocol [11]. The protocol is examined in detail in section 3.3.2 as part of the literature review on RFID security protocols. It is also studied in section 4.2 as a case study. For our purposes in this section, it suffices to say it is a three-pass protocol: a request from system to tag, a second message from tag to system and a third message from system to tag.

After the second message is successfully checked, the system updates the three values it maintains in sync with the tag. The tag, in turn, updates two of those values after the third message is successfully checked. The update of the remaining value by the tag is irrelevant to our discussion. Unfortunately, an attacker can provoke that the tag updates one of the two values only if they maliciously modify the third message [82].

As a result, from the next protocol run onwards, the system:

1. will not identify the tag, which implies a successful denial-of-service attack,

2. will not send the third message to the tag, which leads to the tag not updating its identifier and sending an static one to any interrogating reader. That means the tag is now traceable.

We have just seen that the Henrici and Muller’s protocol [11] suffers from a desynchronisation attack by means of a carefully crafted third message to the tag. Nevertheless, if we obviate that attack, the protocol shows a general technique to provide desynchronisation resistance. The technique consists on maintaining two entries per tag in the system’s database: one for the current identifier, and another one for the previous identifier. In this way, if the database updates the tag’s identifier but the tag does not, e.g. the third message is not correctly delivered, the database entry corresponding to the previous identifier will be used in the next protocol run. Therefore, the tag will still be identified. As a final note, we point out that the tag is traceable between two valid identifications, as the identifier is not updated in the meantime.

The next subsection summarises and closes the section.
A. BACKGROUND ON RFID SECURITY AND PRIVACY

A.2.6 Summary

This section A.2 has provided a description of the different security properties we have found fundamental when approaching security protocols for low-cost RFID tags.

The section set out to describe identification, which is the most basic requirement one would expect from an RFID protocol. We stated identification makes possible for the system to identify a tag. Then, we defined unilateral authentication, which allows one of the parties in the protocol to have the assurance, only at the moment the protocol ends successfully, that the identity of the other party is as claimed. We further affirmed mutual entity authentication is satisfied when unilateral entity authentication holds in both directions.

The section then moved on to the presentation of the two main aims of privacy for security protocols for low-cost RFID tags. The first objective, anonymity, was defined, citing [50], as concealing the identity of an entity involved in some process. The second objective, untraceability, was also defined, this time citing [51], as the ability to prevent other parties from learning one’s current or past location. In addition, two further definitions were given to establish two levels of untraceability: universal and existential.

Then, we affirmed an RFID system provides forward untraceability when we cannot correlate the responses from a certain tag before the last valid authentication session, given that the tag has been compromised afterwards.

Finally, the section examined and defined desynchronisation resistance. We stated that the tag and backend server have been desynchronised when an attacker succeeds in disrupting the consistency of the values some protocols assume in synchronisation.

In the next section, an overview is offered of the main non-protocol proposals to provide security and privacy to RFID systems.

A.3 Non-protocol Proposals

A.3.1 Killing the Tags

We referenced tag killing in subsection A.1.3 when addressing the EPCglobal Class-1 Gen-2 standard [34]. Indeed, EPC tags support this approach. The reader sends a PIN-protected kill command to the tag, which becomes definitively inoperable after receiving it.

It is clear this measure is effective to enforce privacy. As stated in an expressive sentence in [3], dead tags tell no tales. However, the killing approach has numerous drawbacks, even though to some extent, it can be appropriately applied to supply chain applications, where the tags would be killed at the checkout.

Apart from technical obstacles, such as PIN management, there are several functional disadvantages for a variety of applications. For instance, if tags are killed at the checkout, they will not be operable for smart appliances at home.

As a result, alternative measures have been proposed by researchers, as we see in the following subsections.
A.3 Non-protocol Proposals

A.3.2 Put the Tags to Sleep

As an alternative to tag killing, tags can be put to sleep. The idea would be to turn the tag temporarily inactive. For instance, and taking the supply chain by way of example, the tag could be put to sleep at the checkout. Then, once in a place where the tag holder considers there are no security and privacy concerns, the tag could be awoken. For example, to provide the benefits of smart appliance functionality.

Nevertheless, one clear disadvantage of this approach is that there should be some means to both put the tag to sleep and awake it. One possibility would be to have a PIN-protected sleep/awake command, similarly to the case of the killing command. However, that would require the tag holder to bear PIN management, which would cause a significant nuisance to them.

Once more, more satisfactory approaches would be desirable, and again we examine some others in the following subsections.

A.3.3 Blocking

The blocking approach consists in the usage of some cheap passive RFID device, the blocker tag, which would be carried by the tag holder and prevent readers from successfully identifying any tag held by the individual. A well-known proposal instantiating this principle is presented in [28]. The authors present a selective scheme, where only those tags carried by the individual which have been categorised as private, by means of a privacy bit on the tag, will be protected from scanning. The main idea would be to selectively block the anti-collision protocol, which readers use to read tags simultaneously. The interested reader is referred to [28] for an example involving the tree-walking anti-collision protocol and further details.

Using our running example in this section, the supply chain case, a blocker tag could be given at the checkout to the customer, who would carry it with him. As a result, tags selected as private could not be scanned by readers on the person’s way home. Once at home, the blocker tag would be separated from ordinary tags, which could then be interrogated by, for instance, smart appliances, and thus provide their full functionality.

Once more, we have presented an approach where some action is required by the tag holder. They must be aware of the blocker tag, be willing to use it, and remember to separate it from ordinary tags when no security and privacy threats are perceived.

In the following subsection, we succinctly describe yet another alternative, namely proxying.

A.3.4 Proxying

In this subsection we are going to describe the proxy approach to RFID security and privacy.

The proxy could be any RFID-enabled personal device which can act as a reader, such as a mobile phone. We already have at our disposal USIM cards featuring RFID reader chips, which make it possible for mobile phones to read passive UHF tags [138].

Essentially, the proxy would mediate between readers and tags attached to objects the tag holder owns. The objective would be to take charge of a set of tags specified by the owner and enforce a series of privacy-protecting policies.
In the literature, we find some attempts at instantiating the proxying approach, one being the "RFID Guardian" [29] which, as a powerful device, can mediate between readers and the tags it manages and filter requests from unauthorised readers, according to specified privacy policies.

Once more, this approach requires active involvement of the user, who has to manage the proxy device, e.g. specifying appropriate policies. This might be perceived as inconvenient for some users such as those lacking the necessary skills, time or simply the willingness to do it.

In the next subsection we describe policy measures, which is another non-protocol approach to RFID security and privacy, and the last one we review.

A.3.5 Policy Measures

In subsection A.1.5, we already highlighted that several movements have shown strong opposition to the wide-spread deployment of RFID, being the Spychips.com website [52] a well-known example. Not only do they argue over the Big Brother concern but also over any other negative related issue, such as a possible link between implanted tags and lethal tumors in canines [139].

Authorities have not been unaware of the debate, and we also pointed out in subsection A.1.5 several recommendations have been issued by them, aimed at the provision of security and privacy functionality in RFID systems, examples being [53] in Europe and [54] in the US.

Apart from recommendations by authorities, technologists and organisations have issued sets of principles. It is worth citing [55], where a technician presents a set of principles which should help prevent misusing of the technology. For instance, the right to know when, where and why tags are being read, or the right to have them deactivated at the checkout.

We finish this subsection on policy measures by briefly referencing an important observation in [3]. Given their respective limitations, a combination of technology and policy, rather than just one of them only, is key to appropriately mitigating RFID privacy threats.

The next subsection concludes and summarises the discussion on non-protocol approaches.

A.3.6 Summary

This section A.3 has provided an overview of non-protocol proposals to RFID security and privacy.

The section set out to describe tag killing and putting the tags to sleep. The former is an effective measure to provide privacy after the killing command. However, we pointed out several drawbacks which call for alternative approaches to be considered. The latter made it possible to awake the tag afterwards, and enjoy the benefits of the technology, for instance, once at home after buying a product. Nevertheless, it requires some means to put the tags to sleep and then awake them. PINs are one such possibility, but their management might not be convenient for the user.

The section then turned to the description of blocking, which prevents readers from successfully interrogating those tags categorised as private. Once more, we noted the user has to take an action to use the technique, that is, they must remember to have the blocker tag with them, and, once in an environment perceived as free from security and privacy threats, separate it from ordinary tags.
A.3 Non-protocol Proposals

We then addressed proxying, and noted a personal device such as an RFID-enabled mobile phone could be used as a proxy, which would mediate between readers and managed tags. It would enforce policies to protect the privacy of the tag holder. As we had seen with all the previous non-protocol proposals, the user is actively involved, e.g. to specify the policies.

Then we reviewed the last proposal, policy measures. We highlighted some organisations strongly oppose widespread deployment of RFID. We also noticed authorities have already issued recommendations to include security and privacy mechanisms for RFID technology. Finally, we mentioned sets of principles have been stated by technologists and organisations.

This summary has attempted to provide the main ideas about this current section on non-protocol proposals.
Appendix B

Literature Review on RFID Security Protocols

B.1 Proposals Using Public Key Cryptography

B.1.1 Suitability for Low-Cost RFID Tags

In subsection A.1.5 we stated that the cost of the tag must be reduced to the five-cent figure if it is to become viable for wide-spread deployment [31, 32, 33]. It is also widely acknowledged that the cost of adding a security mechanism can be estimated as a function of the number of gates needed to implement its digital components on chip, being the current cost of a gate around one thousandth of a cent [33]. We should then take into account we have no more than around 2000 to 5000 gates at our disposal for security measures on low cost tags [81].

As a result, if we want to know whether a protocol proposal based on PKC is viable for low cost RFID tags, we must first investigate the cost of implementing both a PKC processor (to perform the costly PKC computations) and the corresponding algorithm. Low cost tags, apart from small footprint, have some other constraints as well, such as low power and short bandwidth. However, we focus on the area required, as our main aim is to estimate the cost of the additional security functionality.

We now review cost implementations for three PKC cryptosystems: NTRU [140], which we see is appropriately small footprint, but has significant security concerns; ECC, where most proposals are found; and Rabin’s scheme [141], based on the the difficulty of factoring large integers, as RSA [142], which has been shown unsuitable for implementation on low-cost tags.

1. In [143] it was concluded the NTRUEncrypt algorithm, based on the NTRU cryptosystem, is suitable for low cost tags, featuring area requirements around 3000 gates. It is worth noting this result is comparable to the one in [86] for symmetric key cryptography, which featured an area requirement of around 3600 gates for a low-power small die size version of the AES algorithm. However, the authors in [140] provide a security analysis of the NTRU cryptosystem where several attacks are pointed out, such as brute force, meet-in-the-middle and multiple transmission.

2. Most PKC proposals for RFID tags tend to focus on ECC. In [144], the authors investigate the viability of both the Okamoto’s identification protocol [60], and the original Schnorr’s identification protocol [59], which we review later in this section. They conclude the former can be realised from around 8500 gates. They also show the ECC version of the latter is found to be slightly less expensive than the
Okamoto’s scheme. Both are, though, well beyond the current silicon area at our disposal for security functionality, which can be situated, as we mentioned above in this same subsection, between around 2000 and 5000 gates. The area requirements of ECC proposals are, in general, situated around 10000 gates [103].

3. In [143], the feasibility of Rabin’s scheme for low cost tags was also examined, and it was concluded it was not a suitable possibility. For security equivalent to 60 bits, it required an area of around 17000 gate equivalences.

B.1.2 Protocol Proposals

The Schnorr’s scheme [59] is an efficient three-pass identification protocol providing resistance against passive attacks. It is secure under the discrete logarithm (DL, for short) assumption.

We now describe the Schnorr’s identification protocol, see also figure B.1 on page 107:

1. In the first message, the tag chooses a random number \( r \) within \([0, q - 1]\), where \( q \) is the prime order of the group of points \( E_p \) of an elliptic curve \( E \) defined over a field \( F_p \), with \( p \) a prime integer. It then calculates \( X \leftarrow r \cdot P \), where \( P \) is a point on the curve, and sends it to the system.

2. The system then choses a random number \( e \), the challenge, within \([0, q - 1]\), and sends it to the tag.

3. On reception of \( e \), the tag calculates \( y = s \cdot e + r \mod q \), where \( s \) is the secret key of the tag, and sends it to the system. Once \( y \) has been received by the system, the tag is successfully identified if there is some identity \( I \) in the list of the system \( L \), such that \( y \cdot P - X = e \cdot I \).

![Figure B.1: The Original Schnorr Scheme.](image-url)
We observe, though, that an adversary eavesdropping on the protocol, and thus capturing the messages exchanged, is also able to infer the identity of the tag. The randomised Schnorr scheme [18] addresses this weakness. Roughly speaking, it slightly differs from the Schnorr scheme in that:

1. It additionally uses a new key pair \((v,v \cdot P)\), where the secret key \(v\) is only known to the system.
2. It sends, in the first message, an extra computation, \(X_2 = \beta \cdot v \cdot P\), which guarantees just the system is able to perform the verification and derive the identity \(I\). \(\beta\) is a random number within \([0, q - 1]\).

We understand that it is appropriate to conclude this subsection by mentioning the Okamoto’s identification protocol [60]. Firstly, we describe the three-pass protocol, see also figure B.2 on page 108:

1. In the first message, the commitment, the tag chooses two random numbers \(r_1\) and \(r_2\) within \([0, n - 1]\), where \(n\) is the order of the curve. It then calculates \(X \leftarrow r_1 \cdot P_1 + r_2 \cdot P_2\), where \(P_1\) and \(P_2\) are points on the curve, and sends it to the system.
2. In the second message, the challenge, the system chooses a random number \(e\), within \([0, 2^t]\), and sends it to the tag.
3. On reception of \(e\), the tag calculates \(y_i = r_i + e \cdot s_i \mod n\) for \(i=1,2\) and sends \(<y_1, y_2>\) to the system. \((s_1, s_2)\) is the tag’s secret, satisfying \(Z = -s_1 \cdot P_1 - s_2 \cdot P_2\).
4. Once \(<y_1, y_2>\) has been received by the system, the tag is successfully identified if \(y_1 \cdot P_1 + y_2 \cdot P_2 + e \cdot Z\) equals \(X\).

![Figure B.2: The Okamoto Scheme.](image)

---

108
B.2 Proposals Using Non-Standard Cryptography

Unlike the aforementioned original Schnorr’s scheme, which offers resistance against passive attacks only, the Okamoto’s scheme can also be proven resistant against impersonation under active and concurrent attacks assuming the discrete logarithm problem is hard. That proof is possible because of the witness-indistinguishability \cite{145} and proof-of-knowledge \cite{146} properties of the scheme. We refer the interested reader to \cite{60} for a presentation of the scheme, and to \cite{144} for a discussion on the feasibility of its implementation on RFID tags.

B.2 Proposals Using Non-Standard Cryptography

B.2.1 Ultralightweight Authentication Protocols

The first three protocols we cite in this family were proposed by Lopez et al: LMAP \cite{62}, which stands for Lightweight Mutual Authentication Protocol, M2AP \cite{63}, which stands for Minimalist Mutual-Authentication Protocol, and EMAP \cite{64}, which stands for Efficient Mutual Authentication Protocol.

All three protocols use XOR (\(\oplus\)), AND (\(\wedge\)), OR (\(\vee\)), and modular addition only, and the PRNG is just on the system’s side. Tag and system share a public index pseudonym, IDS, which allows the system to locate the tag’s secrets in its database. The tag’s secrets are its identifier and key material. The protocols feature three steps:

1. In the first step the tag is identified. The system sends a ”hello” message to the tag, and the tag replies with its IDS, which effective identifies the tag, as it allows the system to locate the tag’s identifier and key material on the database.

2. In the second step, mutual authentication is accomplished. The system generates two nonces, and creates a message for the tag using the nonces and the key material of the tag. We emphasise only ultralightweight operations are used. On reception, the tag authenticates the system and extracts the nonces. Then, it creates a message for the system, which will allow the system to authenticate it.

3. In the third and last step, IDS and key material are updated.

Unfortunately, all three protocols suffer from a variety of attacks. We reference here the ones illustrated in \cite{147}. The authors first present a practical desynchronisation attack. It consists on changing some bits in the message sent from system to tag in the second step, and then, intercept and correct the reply from tag to system. The mutual authentication will still be successful, but the shared values are updated differently in the system and the tag, thus effectively desynchronising them. More importantly, the authors also present a full-disclosure attack which allows the adversary to obtain the identifier of the tag. This attack definitely invalidates the security claims of all three protocols. We refer the interested reader to \cite{147} for the details of the attacks.

The following protocol we reference in this category was proposed by Chien \cite{19} and called SASI, which stands for Strong Authentication and Strong Integrity. This mutual authentication protocol follows the characteristics of the ultra-lightweight family we stated at the beginning of this section. Most notably,
rotations were added over the operations already used in the three previous protocols described above. We highlight randomness is provided by the reader, so no pseudorandom number generator (PRNG) is required in the tag. The protocol consists of four messages, see [19] for detailed contents. The first message is a "hello" from reader to tag. The second message is a public index-pseudonym, IDS, from tag to reader. The reader uses it as an index to locate the tag secrets (identifier and keys) on the reader’s database. The third message is sent from reader to tag and involves some ultralightweight computations including nonces generated by the reader itself. Then, on reception of the third message, the tag can authenticate the reader. The fourth and last message is sent from tag to reader, and once the reader receives it, it can authenticate the tag. After successful authentication, secrets and IDS are updated at both system and tag.

Several attacks have been shown against SASI over the years. For instance, in [148], the authors point out two desynchronisation attacks, the first one can be avoided if two copies of the variables are stored in the reader’s database, however, even with that fix, the second attack still applies. We refer the reader to [148] for the details of the attacks. Another example can be found in [149] where the authors present a practical passive full-disclosure attack requiring to eavesdrop $2^{17}$ or $2^{19}$ runs depending on whether Hamming-weight rotation or modular rotation, respectively, is used (the type of the rotation was not defined clearly in [19]).

We finish our review of ultralightweight protocols referencing a mutual authentication one presented by Eghdamian and Samsudin [65]. This protocol features just $\oplus$, $\land$, $\lor$, modular addition and data-dependent rotations, which are, again, very simple operations, requiring few area and power requirements on chip. The protocol set out to achieve more security than their predecessors in the family. Unfortunately, further research, once more, showed otherwise. Similarly to the previous schemes, the protocol consists of three phases: identification, mutual authentication and updating of shared key material and tag’s identifier. The first message is a "hello" from system to tag. In the second one, the tag replies with its IDS, which identifies it to the system. The system then generates a random value and constructs a message for the tag involving the random value and key material, which allows the tag to authenticate the reader. The tag then replies creating another message involving random value and key material as well, which in turn allows the system to authenticate it. After successful authentication, secrets are updated at both system and tag.

In [105], though, a key-recovery attack was presented that allows an attacker to recover the key material shared between system and tag. If a 96 bit key material length is used, as recommended by the authors of the scheme in [65], the attack can be carried out by a passive adversary in around 20 authentication sessions. For greater key material lengths, the attack is still very efficient, as noted in [105], where the details of the attack can be found.

The authors in [105] also question, due to the difficulty to attain security and privacy under the constraints of ultra-lightweight protocol proposals, as we have seen in this subsection, whether this approach is really relevant.

### B.2.2 Lightweight Authentication Protocols

The first protocol proposal we review was presented by Chien [20] in 2007. It is a mutual authentication protocol claiming compliance with the EPC C1G2 standard, which aims at solving its security shortcomings. In Chien’s scheme, tags store three values: an authentication key, an access key, and the EPC. The first
two values change every time an authentication session is successful. The system has an entry for each tag containing current and previous authentication and access keys, the EPC and the data related to the object the tag is attached to. The protocol run is as follows:

1. The system sends a nonce to the tag.

2. On reception, the tag generates another nonce, and computes the CRC of the concatenation of its EPC and both nonces. Then, the tag xores it with its authentication key and sends the result to the system.

3. Once the system has received the message from the tag, it searches the entry of its database satisfying the calculation made by the tag. For each entry, it checks using both the current and the previous authentication key values. This is a technique intended to provide resilience against desynchronisation attacks.

If tag authentication is successful, the system updates its keys, current and previous values. In addition, it computes the CRC of the concatenation of the tag’s EPC and the nonce sent by the tag. Then it xores it with its access key. Depending on whether the current or previous authentication key was received from the tag, the access key used by the system is the current or the previous one, respectively, as well.

4. When the tag receives the message from the system, if the calculation received holds, it updates its keys.

Unfortunately, the scheme does not satisfy its security claims. As demonstrated in [150], it suffers from many vulnerabilities. The first issue is that EPCs allow $2^{88}$ possible values, and the CRC defined in the standard is a 16-bit one only. As a result, unequivocal identification of tagged items is not guaranteed.

The absence of unequivocal identification also leads to system auto-desynchronisation just in normal operation. The reason is more than one EPC ($2^{88}$ possible values) yields the same CRC computation ($2^{16}$ possible values), which is sent from tag to system. This computation is checked by the system to identify the tag. Therefore, the system might update a database entry corresponding to a tag which is different from the one being authenticated. The probability that this behaviour leads to desynchronisation is non-negligible [150].

Both weaknesses could be solved using larger CRCs, non-conforming to the standard. Nevertheless, the authors show other attacks which are much more difficult to address, as they are due to the linearity of CRC. A passive attacker eavesdropping just one protocol run is able to construct a valid message to impersonate the tag, and likewise to impersonate the system. Finally, it is also shown tags are traceable, as an attacker just has to listen to two non-consecutive iterations of the same tag, xor the captured messages, and due to the properties of CRC he is able to check whether both messages originated from the same tag. The reader is referred to [150] for a detailed exposition of the attacks.

The review of Chien’s scheme has clearly showed abusive use of CRC makes the provision of security and privacy very difficult. Indeed, CRC was also involved in some other infamous cases in security, e.g. WEP [151].

The second proposal we review in this category concerns Mitra’s scheme, presented in [66] in 2008. In this proposal, the author assumes the tag supports a PRNG and lightweight operations such as addition and
multiplication. Tags store a static EPC and a key K, which is shared with the system. The protocol is quite simple and consists of three messages:

1. The system sends a "hello" message to the tag.

2. On reception, the tag generates a random number N, computes \( f = N \cdot K + \text{EPC} \), and sends it to the system.

3. Once the system has received the message from the tag, it recovers the EPC from it as \( \text{EPC} = f \mod K \), where K is assumed to be greater than EPC for the recovery to be possible.

Once more, this scheme also falls prey to attacks which invalidate its security claims. In [152], the authors identify cloning and traceability attacks. Furthermore, they also identify a full disclosure attack, which encompass the previous ones. The main problem in this occasion is the use of addition and multiplication, where an standard cryptographic protocol would have used a cryptographic hash function or an encryption function instead. The use of those simple operations makes it possible to analyse the differences between captured messages, leading to definitive security weaknesses. We overview the full-disclosure attack, as it is final.

To disclose K, the adversary captures the messages of s sessions, \( f_1, f_2, \ldots, f_s \), and calculates the gcd of the s-1 differences, \( f_1 - f_2, \ldots, f_1 - f_s \). Both experimentally and in theory, the probability of success of the attack ranges from 60% when two sessions are eavesdropped to 100% when just eleven sessions are eavesdropped. Once K is derived, EPC is inferred from \( \text{EPC} = f \mod K \), completing the full disclosure attack. The reader is referred to [152] for a detailed description of the attacks.

In 2008, the Qingling et al mutual authentication protocol was presented [36]. As Chien’s scheme above had attempted, it also tried to both improve the security of the EPC C1G2 standard and remain compliant with it. Interestingly, the authors provided, in this case, a formal analysis based on BAN logic [93]. Unfortunately, the authors fell into the same pitfall, namely, they did not take into account the linearity of CRC. The BAN logic analysis was also invalid because the CRC was modelled as if it was a secure encryption function, but it is not.

In [152], the authors show how both tags and readers can be impersonated by an attacker. In short, the attacker can take advantage of the linearity of CRC and, after eavesdropping one authentication session, they can build a message which is accepted as valid. This is possible even if the adversary does not know any secret material. Furthermore, the authors also show a traceability attack is also possible. Just performing simple computations based on CRC linearity, they can determine whether two responses originate from the same tag. We refer the interested reader to [152] for a detailed description of the attacks.

In the literature, we can find more examples of proposals in this category which have been presented and subsequently found flawed to some extent. Yet another example involves the mutual authentication protocol by Chen and Deng [37] in 2009. It again tried to improve the security of the EPC C1G2 standard and remain compliant with it. To this end, only PRNG and CRC, apart from simple operations, were used. Unfortunately, the attempt proved flawed once more. In [153], the authors demonstrate several attacks which reduce the security of the scheme to that of the standard. The attacks include tag and reader
impersonation, tag traceability, and DoS attack. Once more, the linearity of CRC for all three attacks, and this time a weakness in the XOR operation for the latter, make the attacks possible.

We finish our review of this category of protocol proposals by highlighting it has been claimed it is almost impossible to attain security and privacy under the constraints of the EPC Class-1 Gen-2 standard [38]. The reason is the use of weak building blocks, such as CRC, to try and play the role of an encryption function in standard cryptography.

### B.2.3 Proposals Based on the LPN Problem

The **HB protocol**, presented in [67], relies for its security on the difficulty of the LPN problem. The protocol was adopted for the low-cost RFID field in [68]. It is very simple, and uses binary inner product and a noise bit only, which makes it very lightweight to be implemented in hardware. Tag and system share a secret k-bit vector $x$. The protocol is run $r$ times. One round of the protocol is as follows:

1. The system sends a random challenge k-bit vector $a$ to the tag.
2. On reception, the tag computes $z = (a \cdot x) \oplus v$, where $v$ is 1 with probability $\eta < 0.5$. We call $v$ the noise bit. Then $z$ is sent to the system.
3. Once the system has received the message from the tag, it counts it as correct if $a \cdot x = z$.

After $r$ rounds, the system accepts the tag if less than $\eta r$ responses are incorrect.

It can be shown a passive adversary attempting to impersonate a tag faces a problem equivalent to LPN [68], which is known to be NP-hard [154]. Therefore, the HB protocol is resistant to passive attackers.

However, an active attacker can send adaptive challenges to a tag. If the attacker sends the same challenge $a$ many times, they derive the secret $x$, as the noise is easily removed. As a result, HB is not resistant to an active attacker. To overcome this weakness, the authors in [68] introduced the **HB+ protocol**, which was meant to resist active adversaries. HB+ is still very simple and adds little overhead to HB. Essentially, an additional shared secret $y$ is added. Now, the protocol proceeds as follows:

1. The tag sends a random commitment k-bit vector $b$ to the system.
2. The system sends a random challenge k-bit vector $a$ to the tag, as in plain HB.
3. On reception, the tag computes $z = (a \cdot x) \oplus (b \cdot y) \oplus v$. The functionality of $v$ does not change. Then $z$ is sent to the system.
4. Once the system has received the message from the tag, it counts it as correct if $(a \cdot x) \oplus (b \cdot y) = z$.

Again, as in HB, after $r$ rounds, the system accepts the tag if less than $\eta r$ responses are incorrect, where $\eta < 0.5$ is the probability that $v$, the noise bit, is 1.

Intuitively, we see the inclusion of $b$ in the first step, and also $(b \cdot y)$ in the last, aims at preventing an active adversary from learning $x$ or $y$ by sending adaptive challenges $a$ to the tag. A security proof was also provided in [68].
B. LITERATURE REVIEW ON RFID SECURITY PROTOCOLS

Shortly afterwards, in [155], it was shown for both HB and HB+ that their security was preserved if their iterations are parallelised. As a result, they can be executed in less rounds.

In [135], however, a man-in-the-middle attack was presented against HB+. The attack consists in choosing a constant vector $\delta$ and xor it to each challenge $a$ at each round of the protocol. If authentication is successful $\delta \cdot x = 0$ with high probability, and 1 otherwise. This reveals one bit of the secret $x$. To reveal the whole of $x$, we can repeat the attack $k$ times choosing linearly independent $\delta$. Once $x$ has been obtained, the adversary can obtain $y$ as well. We refer the reader to [135] for further details.

After the presentation of the man-in-the-middle attack in [135], researchers worked on the improvement of HB+ so that it could resist this kind of attacks. However, most of them resorted to the inclusion of elements which increased the cost beyond the current tight limit of low-cost tags. For instance:

1. In [69], the authors presented the HB++ protocol. The protocol resists the attack by Gilbert et al [135] on HB+. However, the new version features a preliminary stage for every authentication where two challenges are exchanged between the communicating parties (system and tag). This stage resorts to a universal hash function.

2. In [70], Random-HB# and HB# are introduced. The Random-HB# protocol improves the security of HB+ as it resists a class of active attacks which includes the one presented in [135] and other active attacks. However, it requires the storage of two random binary matrices on tag, which is too high a cost. This is the reason why the authors present the HB# protocol, as an enhancement to the performance of Random-HB#. HB# uses two random binary Toeplitz matrices [156, 157], which have lower storage requirements, but still significant for low-cost tags.

3. Similarly to the previous attempt, the Trusted-HB protocol is presented in [71] as a new version improving on HB+. Nevertheless, in this occasion an LFSR-based Toeplitz matrix [156] is used to construct a hash function, which, again, leads to a significant cost for low-cost tags.

LFSRs, short for Linear Feedback Shift Registers, are typically found as components of many keystream generators. They comprise a number $N$ of 1-bit stages, which have one input and one output. At each clock unit, the contents of the stages are shifted. The bit contained in stage 0 is output and it is added to the keystream being generated. The new bit in stage N-1 is the XOR of a fixed subset of all the stages. For an in-depth treatment of feedback shift registers in general, and LFSRs in particular, the interested reader is referred to [50].

Finally, we want to highlight the proposal in [21] to improve on HB+, the HB-MP protocol. The protocol features $r$ rounds, and both system and tag share two $k$-bit secret keys $x$ and $y$. The exchange of messages for the $i$th round is the following:

1. The system generates and sends a random m-bit string $a$ to the tag.

2. The tag calculates $x = \text{Rot}(x, y_i)$, where Rot is the bitwise left rotation of $x$ by $y_i$ positions, and $y_i$ is the $i$th bit of the secret key $y$.

3. Then, the tag further calculates $z = a \cdot x_m \oplus v$. 


114
4. After that, the tag chooses $b$ such that $b \cdot x^m = z$, and sends it to the system.

5. On reception, the system computes $x = \text{Rot}(x, y_i)$, and checks whether $a \cdot x^m = b \cdot x^m$.

Again, as in HB, after $r$ rounds, the system accepts the tag if less than $\eta r$ responses are incorrect, where $\eta < 0.5$ is the probability that $v$, the noise bit, is 1. We have pointed out this proposal because we see it remains suitable for current low-cost tags, due to its lightweight nature, including storage and operations used. Furthermore, to protect against the man-in-the-middle attack in [135], the rotation of the secret $x$ is included.

Unfortunately, the rotation does not prevent a man-in-the-middle attack, as the rotated value remains the same at the $i$th rotation of every authentication session. Even if the secret value $x$ was updated at each authentication session, the attack would still be possible, as $x$ would eventually be rotated to its initial value. For this reason, the authors in [22] propose an improvement on HB-MP, the HB-MP+ protocol. The central idea is to make the rotation random in every round, so that it is unpredictable to an attacker, and thus, the attack thwarted. Nevertheless, HB-MP+ relies on a one way function, which, again, would be beyond the very tight limits of current low-cost tags.

We want to finish our exposition of the HB-family by noting a very important point. Transmission costs for an entire authentication session are very high. In [70], it is shown for HB+ that around 61,500 bits must be transmitted if a good security level is parameterised.

B.2.4 Minimalist Cryptography

The scheme works as follows: the tag emits the next pseudonym in a list each time it is interrogated by a reader. Once the list is exhausted, the tag cycles and starts from the beginning of the list again. This basic idea is combined with a throttling mechanism, that is, the tag replies to the reader’s queries at a low rate. In this way, the task of the attacker to capture the whole list of pseudonyms is made harder. In addition, there is also provision for the updating of the list of pseudonyms by authorised readers, which necessarily implies the tag must authenticate the reader. Finally, note the pseudonyms are generated by the reader, as the tag, being low-cost, cannot afford it.

The protocol aims at providing mutual authentication between reader and tag. The mechanism is essentially the following:

1. The tag releases a pseudonym $\alpha_i$, which identifies it to the reader.

2. Then, the reader replies with a key $\beta_i$ uniquely associated to $\alpha_i$, which authenticates the reader to the tag.

3. The tag will now reply with a key $\gamma_i$ uniquely associated to $\alpha_i$ and $\beta_i$, which authenticates the tag to the reader.

As described, the scheme would have its security and privacy limited to the period of time the adversary needs to capture the whole list of triples $\alpha_i$, $\beta_i$ and $\gamma_i$. To deal with this problem, legitimate readers are able to update the triples in the tag after every successful mutual authentication session. To provide privacy, the
values are updated using one-time pads transmitted to the tag from several protocol runs. This updating mechanism prevents an attacker who eavesdrops discontinuously from learning the updated values of the triples.

From the description of the protocol that we have just provided, it can be observed the operations involved are computationally very lightweight. Furthermore, the one-time pads applied require only the inexpensive XOR operation. Nevertheless, transmission costs for the updating mechanism are significant. They are linear in the length of the values stored in the tag and in the number of consecutive authentication sessions the one-time pads are formed over.

Finally, we observe that the author assumes limited adversarial power, which he considers adequate for a practical RFID system. The resistance of the proposed scheme is studied in relation to this weak adversarial model. In particular:

1. The adversary can interact with a tag a limited amount of times between two successful interactions with a legitimate verifier. This would follow from the throttling mechanism applied by the scheme.

2. An adversary is constrained in his ability to mount man-in-the-middle attacks:
   
   (a) If the adversary is stationary, the essential supporting argument is two-fold. Firstly, in many RFID environments, tags are mobile, such as at the checkout in a supermarket, where users queue up for a limited amount of time. This would restrict the ability of the attacker to obtain information for several interactions. Secondly, the throttling mechanism applied by the scheme itself would reinforce this effect.

   (b) If the adversary is mobile, the aim would be to interrogate tags and then use that information with readers. It is assumed it would be difficult for them to repeatedly move from tag to reader and the other way round.

B.3 Proposals Using Symmetric Key Cryptography

B.3.1 Protocols Providing Linear Complexity

One of the first proposals in the field was the Hash Lock Scheme [74]. It is based on the use of a hash function. Tags can be in either a locked or an unlocked state. The tag holder locks a tag by storing on it a hash of its ID, called its metaID. While locked, the tag responds to interrogating readers with the metaID. Legitimate readers forward the value to the database. The database identifies the tag by its metaID and sends the ID to the reader. The reader forwards the ID to the tag, which hashes and compares it to the metaID. If they match, the tag unlocks itself.

Unfortunately, as also noted in [11], the scheme suffers from many weaknesses. It is easy to mount a replay attack to impersonate the tag to the reader. The attacker just needs to capture the metaID from the tag and then send it to a legitimate reader. In addition, the attacker can trace the tag by its metaID. Furthermore, a man-in-the-middle attack can be made to unlock the tag: the attacker interrogates the tag,
forwards the reply, i.e. the metaID, to the reader, and then unlocks the tag with the reader’s response, i.e. the ID.

The authors of the Hash Lock Scheme propose in the same paper [74] an alternative mode of operation, namely the Randomised Hash Lock Scheme. The motivation is the prevention of the tracking of tag holders. To this end, tags now reply with a metaID which changes with every interrogation. The tag emits a message $r, h(ID, r)$ where $r$ is a random number.

We must note, though, the system now needs to perform an exhaustive search to identify the tag. In addition, it is still possible to mount a replay attack, as the adversary just needs to capture a response message from the tag and send it to the reader to be successfully identified. Furthermore, the tag can be traced if the attacker captures the ID sent by the reader to unlock the tag. Finally, no forward untraceability is offered, as compromise of the tag allows the attacker to link past messages sent from the tag.

The next protocol we review is the Rheee et al scheme [23]. It needs support for a hash function and a PRNG on the tag. It is a very simple and typical challenge-response protocol. Simply stated, the system sends a nonce to the tag in the first message. The tag replies with another nonce and a hash computation including the secret shared identifier and both nonces. Finally, in the third and last message, the system sends a hash computation including an identifier and the system’s nonce. To find the identifier of the tag, the database needs to perform an exhaustive search.

The scheme provides mutual authentication, anonymity and existential untraceability. Unfortunately, it does not provide forward untraceability because if the tag is compromised and its identifier captured, it is possible for the attacker to link past runs of the tag.

Finally, we review the PEPS protocol, standing for Private and Efficient Protocol based on a Stream cipher [75]. It was introduced in 2010 by Billet et al. It focuses on the use of a stream cipher, as an encryption primitive. Most notably, a lightweight stream cipher could be used, as those featuring the portfolio of the eSTREAM project [158]. It is a three-pass protocol achieving mutual authentication. The complexity of the search to authenticate the tag is linear, i.e. the reader exhaustively searches its database. In order to deal with desynchronisation, current and previous secrets are stored in the database for each tag. It offers excellent privacy properties (anonymity, existential untraceability and forward privacy). Nevertheless its complexity makes it non-scalable. We refer the interested reader to [75] for a detailed description of the exchanged messages.

### B.3.2 Protocols Involving the Reader in the Refreshment

In [24], two proposals by Dimitriou are presented:

1. The first one is intended to provide constant-time identification. The tag stores its secret identifier, which it shares with the database. In addition, the database stores the hash of the identifier as well. The hash is used by the database to efficiently identify the tag.

   The protocol is quite simple. Upon request from the system, the tag replies $<h(ID), \text{Nonce}, h(ID, \text{Nonce})>$, and updates its identifier with a one-way function. The interesting point for us is the $h(ID)$, which, as we observed, allows the database to identify the tag in minimal time. Once the reply from the tag is checked, the system updates the identifier in the tag’s entry as well.
We can observe the protocol is forward untraceable and offers existential untraceability and anonymity as well. However, we highlight that it is possible to launch a desynchronisation attack that requires only one unsuccessful interrogation. In that case, the tag refreshes its identifier but the database does not. This implies further authentication sessions require an exhaustive search on the database.

2. To address the attack previously stated, the author provides a second protocol proposal. The protocol is three-pass and very simple:

(a) In the first message, the reader sends a nonce <NonceReader> to the tag.

(b) In the second message, essentially, the tag replies with <H(ID), NonceTag, H(ID,NonceTag,NonceReader)>.

(c) The reader efficiently identifies the tag by H(ID), authenticates the tag, updates the ID as UpdatedID and then sends the third and last message <H(UpdatedID,NonceTag,NonceReader)> to the tag.

(d) Once the reader is authenticated by the tag as a consequence of the last message, it also updates the ID as UpdatedID.

This second proposal is essentially the same as the first one but adding mutual authentication. In this way, the tag only refreshes its identifier if the interrogating reader is legitimate. However, now the tag can be traced between two valid successful authentications, as the tag uses the same identifier in between.

The \( C^2 \) scheme [13] is another example of protocol involving the reader in the refreshment. It is influenced by the second proposal in Dimitriou in [24] we have just reviewed above. It makes some variations though. The first one is it removes \( h(ID) \) from the second message (tag to reader). In this way the tag becomes untraceable between two successful authentication sessions, which does not occur in the second proposal in Dimitriou in [24]. Unfortunately, the price to be paid is a far less efficient identification procedure as now the system has to conduct a linear complexity search to identify a tag, computing up to 3 hash operations per tag in the database.

As in the second proposal by Dimitriou in [24], there are three first messages playing similar roles. In particular, the tag updates its internal state after authenticating the reader in the third message. In the \( C^2 \) scheme, though, there is a fourth message (from tag to reader), which informs the reader that the tag has updated its identifier, and it is at this moment that the reader updates its database entry for the tag, whereas in Dimitriou the reader updated its entry after the second message. The interesting point here is a different way to deal with desynchronisation, namely, the database does not store the previous identifier. It stores the current one and, if needed, it calculates the next one on-the-fly, thus reducing the amount of database space needed.

In [11], Henrici and Muller, apart from pointing out several weaknesses of the Hash Lock and Randomised Hash Lock [74] schemes, propose a new protocol. The protocol is also based in a hash function. Essentially, the tag stores a varying identifier ID and two variables, namely a transaction number TID and a last successful transaction number LST. The database identifies the tag by the received hash(ID), which
makes the identification complexity minimal. The use of the transaction number prevents replay attacks. It is also noteworthy that it maintains two entries per tag in the database to mitigate desynchronisation.

Unfortunately, the protocol is not without its weaknesses. For instance, as highlighted in [79], an attacker can artificially increase one of the values sent from tag to reader, namely the number of unsuccessful authentication attempts since the last successful one, and then trace the tag by this abnormally high value. In the same paper, an attack to desynchronise the tag and database is also presented. The attacker manipulates the last message from the system, taking advantage of the use of the XOR operator. The Henrici and Muller protocol is treated in much more detail in section 4.2 where it is examined as a case study.

B.3.3 Protocols Providing Sublinear Complexity

B.3.3.1 Hash-Chain-Based

The first proposal we review in this group is a very influential one, the Ohkubo, Suzuki, and Kinoshita’s protocol (OSK, for short) [25]. The distinguishing characteristic of this protocol is the use of the hash-chain technique, which is used to update the tag’s secrets, and most notably provides forward untraceability, as an attacker compromising a tag is not able to link past sessions of the tag.

Both tag and database share an initial secret $\text{Secret}_1$. The protocol is quite simple, and can be described as follows:

1. Firstly, the system sends a request to the tag.
2. Then, the tag updates sets its new secret $\text{Secret}_{i+1}$ to $H(\text{Secret}_i)$ and sends $G(\text{Secret}_i)$ to the system, where $H$ and $G$ are hash functions.
3. On reception, to identify the tag, the system has to go through all its database entries. For each entry, it has to construct the hash chain from $\text{Secret}_1$ up until it finds the value received from the tag or the maximum length of the hash chain $m$ is reached.

As we can see, the complexity is $O(Nm)$, where $N$ is the number of tags in the system. It is an identification protocol, which does not provide any entity authentication. In addition, it is easy to see the protocol is susceptible to a replay attack. Indeed, just replying a previous query, the attacker is identified by the system. Nevertheless, it offers excellent privacy, as it provides anonymity, existential untraceability and, as we noted, forward untraceability. Finally, we note that the protocol can be desynchronised if the tag is queried by an illegitimate reader more than $m$ times. The tag holder, though, would be able to resynchronise the tag by out-of-band action.

The same authors propose some improvements to OSK in [159]. Unfortunately, they are at the cost of degrading privacy, as noted in [82]. For instance, the hash $H$ is calculated once every $c$ queries only, where $c$ is a predefined parameter. To take forward privacy to serve as an example, this modification negatively affects it, as it is then not provided in the period between two calculations of $H$.

The Avoine et al improvement to OSK in [58] adds:
B. LITERATURE REVIEW ON RFID SECURITY PROTOCOLS

1. Tag authentication to the protocol. The difference is a nonce Nonce is sent in the request from the system, and then the response from the tag is now G(Secret_i ⊕ Nonce). This modification effectively thwarts the replay attack on the original OSK.

2. Following previous work in [89], a time-memory trade-off to improve the complexity of the search at the server’s side in the original OSK. The approach is based on work by Hellman in [160]. The more memory is used, the faster is the identification procedure. The improvement ranges from O(1) to O(N), depending on the trade-off used. For instance, the authors in [89] give O(N^{2/3}) as an example complexity when O(N^{2/3}) memory is used. As a final note, it must be observed that to apply the time-memory trade-off to the version of OSK including tag authentication, it is necessary to add G(Secret_i) to the tag’s response. The reason is the randomness of G(Secret_i ⊕ Nonce).

B.3.3.2 Efficient Database Structures

Following the same objective as the Avoine et al improvement to OSK in [58], the reduction in the complexity of the identification procedure, we find several protocols using efficient database structures.

The first one we review is the Molnar and Wagner’s protocol [26]. This time, the idea is essentially to arrange database’s secrets in a tree structure. Unfortunately, even though complexity is indeed successfully reduced from O(N) to O(logN), privacy is degraded.

The basic protocol presented by Molnar and Wagner, without the application of the tree technique, is a challenge-response one offering mutual authentication and privacy, except for forward untraceability. It requires linear complexity, and therefore is not scalable. To reduce complexity to logarithmic, the tree approach is applied.

The main idea behind this approach is that secrets are arranged in a tree structure, where each edge stores one secret value and the leaves are the tags in the system. Instead of storing one secret per tag, as in the linear model, in the tree model, each tag stores as many secrets as edges exist from the root to the corresponding leave. The average number of accesses needed to identify one tag is (bf × log bf N) / 2, where bf is the branching factor of the tree and N the number of tags in the system.

The message exchange, applying the tree technique, changes with regard to the aforementioned basic protocol. Now, one run of the basic protocol is needed for each level of the tree from root to leaf.

Unfortunately, as we mentioned above, the technique also leads to a degradation of privacy, if an adversary can compromise at least one tag. The details of the attack leading to the probabilistic traceability of tags other than the compromised one can be found in [58].

In [72], Avoine, Buttyan, Holczer, and Vajda’s protocol is presented. Like Molnar and Wagner’s, it also aims at reducing the complexity of the identification by organising tags’ secrets in an efficient structure. This time, though, the scheme is group-based instead of tree-based. Essentially, tags are divided into equally-sized groups, and each group is assigned a secret group key. In addition, each tag possesses its own secret key. It is shown in [72] the group-based structure provides better privacy and complexity levels than the tree-based. Nevertheless, compromise of at least one tag leads to degradation of privacy as well.

Succinctly, the exchange of messages is as follows:
1. The system sends a nonce $\text{Nonce}_S$ to the tag.

2. The tag generates a nonce $\text{Nonce}_T$ and sends two values to the system. The first one is $E_{\text{group-key}}(\text{Nonce}_S, \text{Nonce}_T, \text{ID})$, where $E$ is an encryption function and $\text{ID}$ is the identifier of the tag. The second one is $E_{\text{tag-key}}(\text{Nonce}_S, \text{Nonce}_T)$.

3. On reception, the system identifies the tag by checking the first value through all groups and the second value through all tags within the identified group.

The last protocol in this category we review is the **Cheon, Hong, and Tsudik’s Protocol** in [73]. The structure used is a grid, instead of a tree or a group. The technique is based on the "meet-in-the-middle" strategy, which was used in the past to attack some ciphers [161].

The grid has $n$ columns and $n$ rows, such that $n \times n = N$, where $N$ is the number of tags in the system. We denote $K_1$ as the set of columns, and $K_2$ as the set of rows. Each element in $K_1 \cup K_2$ is unique. The system is set up so that each tag in the system is assigned two secret values, one from $K_1$ and another one from $K_2$. In other words, each position in the grid represents one tag.

The protocol is a two-pass one:

1. The system sends a nonce $\text{Nonce}_S$ to the tag $T_{i,j}$.

2. The tag generates a nonce $\text{Nonce}_T$ and computes $C = \text{PRF}_{K_1}(\text{Nonce}_S,\text{Nonce}_T) \oplus \text{PRF}_{K_2}(\text{Nonce}_S,\text{Nonce}_T)$. Then, the tag sends $\langle \text{Nonce}_T, C \rangle$ to the system.

3. On reception, the system identifies the tag in the following way:

   (a) For each element in $K_1$ it computes $\text{PRF}_{K_1}(\text{Nonce}_S,\text{Nonce}_T)$ and stores the $n=\sqrt{N}$ results in a table.

   (b) For each element in $K_2$ it computes $C \oplus \text{PRF}_{K_2}(\text{Nonce}_S,\text{Nonce}_T)$ and tries to find the result in the table.

   (c) If a match is found, the element is correctly identified. The search takes $2\sqrt{N}$ PRF operations, in the worst case.

As we can observe, the identification procedure is efficient, providing a sublinear complexity $O(\sqrt{N})$.

Regarding security, it is easy to notice that the protocol does not offer forward untraceability, as an attacker compromising one tag can reconstruct past messages corresponding to past transactions of the tag and thus link them. In addition, if an attacker can compromise several tags, they gain knowledge of one or both of the secrets of tags which have not been directly compromised. This issue was noted by the authors themselves in the same paper [73], and leads to impersonation attacks.

To effectively solve the impersonation attacks, the authors present an **authentication extension to** their plain protocol, i.e. the **Cheon, Hong, and Tsudik’s Protocol**. In essence, the differences from the plain version are:

1. Each tag has now a third key which is unique to it.

2. The grid has now $N^\alpha$ columns and $N^\alpha$ rows, which implies $N^{1-2\alpha}$ tags share the same key pair.
3. On reception of the nonce from the system, the tag now additionally calculates $C' = PRF_{K_3}(\text{NonceS,NonceT})$ and sends it to the system together with $\langle\text{NonceT},C\rangle$, where $C = PRF_{K_1}(\text{NonceS,NonceT}) \oplus PRF_{K_2}(\text{NonceS,NonceT})$, just as in the plain protocol.

4. The system performs the same identification procedure as in the plain version, but now additionally checks $C'$ afterwards.

Unfortunately, the extended version suffers from a traceability issue. An attacker compromising the pair $\langle K_1, K_2 \rangle$ of one tag is able to track with probability $1/N^{1-2\alpha}$ all tags sharing the same cell in the grid, as observed by the authors themselves in the same paper [73]. The situation is in fact much worse, because, as noted in [38], the probability that an attacker succeeds in a traceability attack after corrupting a given number of cells is much higher than the one stated by the authors in the original paper [73].

### B.3.3.3 The YA-TRAP family

In this subsection, the last subgroup within those protocols which have been designed to provide sublinear complexity is addressed. This subgroup is embodied by the YA-TRAP family. They are based on the use of monotonically increasing counters. The essential idea is the server maintains a hash table which is pre-computed each time the counter is increased. In this way, when an identification has to be dealt with, the server just has to conduct a constant-time search.

The first protocol in the family is an identification one, **YA-TRAP**, originally proposed by Tsudik in 2006 in [76]. The protocol was presented again as **YA-TRIP** by Tsudik in 2007 in [27]. Then, it was presented once more as **RIP** by Burmester et al in 2009 in [77]. In this work, the last name given to the protocol, RIP, is used. Succinctly considered, the protocol is fairly simple. The tag contains the current counter value $t_{\text{tag}}$, the maximum counter value $t_{\text{max}}$, and a secret key $k_{\text{tag}}$. The exchange of messages is as follows:

1. The system sends its current counter value $t_{\text{sys}}$ to the tag.

2. On reception, the tag checks whether $t_{\text{tag}} < t_{\text{sys}} < t_{\text{max}}$. If the inequality holds, $t_{\text{tag}}$ is updated to $t_{\text{sys}}$ and $H_{k_{\text{tag}}}(t_{\text{sys}})$ is sent to the system. Otherwise, the tag generates and sends a random value to the system.

3. Once the system has received the message from the tag, it just has to conduct a constant-time search for the received hash.

We can see that RIP does not provide authentication because the response from the tag is not fresh and thus subject to replay. It provides identification only.

In addition, it suffers from a straightforward DoS attack, as pointed out by the authors themselves [77]. An attacker can query the tag sending a counter value well into the future. As a result, the tag is disabled either temporarily, if the counter is less than $t_{\text{max}}$, or permanently, otherwise.

It is also well worth noting the protocol assumes the tag is never queried more than once with the same counter value $t_{\text{sys}}$ [77]. That leads to a heavy workload for the server if the interval between counter updates, and consequently the pre-computation of the table, is very short.
B.3 Proposals Using Symmetric Key Cryptography

Finally, we note forward privacy is not provided. An attacker compromising a tag obtains $k_{tag}$ and can reconstruct past protocol messages from the tag and thus link them.

The following protocol in the family was designed to add tag authentication to RIP, and it was presented as **YA-TRIP** by Tsudik in 2007 [27] and then re-named **RIP+** by Burmester et al in 2009 [77]. RIP+ slightly differs from RIP:

1. In the first message, from system to tag, a random nonce $r_{sys}$ is added.
2. In the second message, from tag to system, $<H_{k_{tag}}(r_{tag}, r_{sys}), r_{tag}>$ is added, where $r_{tag}$ is a nonce generated by the tag.
3. On reception, the system identifies the tag exactly as in RIP and then authenticates it by the added $H_{k_{tag}}(r_{tag}, r_{sys})$.

The security profile of RIP+ is the same as RIP but tag to reader authentication is added.

The next protocol in the family we review is **YA-TRAP**, proposed by Tsudik in [27] in 2007. Its aim was to extend RIP+ to mitigate those DoS attacks that we identified above where an adversary could disable a tag either temporarily or permanently by querying the tag with a counter value well into the future.

The idea is the system additionally presents an epoch token when it interrogates a tag in the first message. This token allows the tag to distinguish a counter which is too far into the future. The epoch token changes much less frequently than the counter. In addition, it is possible to derive previous epoch tokens from the received one, but no future ones. The implementation is:

1. The system generates and stores a large chain of hashes.
2. The system assigns the last hash calculated to the epoch token $ET_{tag}$ and stores it in all tags.
3. A system-wide parameter INT, the epoch duration, is defined. It equals a given number of counter updates.
4. Each time INT is updated, the next value in the chain of hashes is selected as $ET_{sys}$, i.e. the current epoch token.

The protocol is thus as follows:

1. The system sends its current counter value $t_{sys}$, a random nonce $r_{sys}$, and the current epoch token $ET_{sys}$ to the tag.
2. When the tag receives the message from the system, it:

   (a) Computes the number of epochs since the last authentication as $\nu = \lfloor t_{sys}/\text{INT} \rfloor - \lfloor t_{tag}/\text{INT} \rfloor$.
   (b) It checks whether $t_{tag} < t_{sys} < t_{max}$ and $H^\nu(ET_{sys}) = ET_{tag}$ are both satisfied, where the latter ensures $ET_{sys}$ is the correct predecessor of $ET_{tag}$ in the hash chain:
      i. If they both are satisfied, $t_{tag}$ is updated to $t_{sys}$, $ET_{tag}$ is updated to $ET_{sys}$, and $<H_{k_{tag}}(t_{sys}), H_{k_{tag}}(r_{tag}, r_{sys})>$ is sent to the system.
ii. Otherwise, the tag generates and sends two random values to the system.

3. On reception, the system identifies and authenticates the tag exactly as in RIP+.

YA-TRAP* indeed mitigates DoS attacks of the aforementioned type. Nevertheless, an adversary can still disable a tag for the duration of INT. Indeed, they can query a tag with the highest \( t_{sys} \) within the current epoch.

In addition, it is also possible for an attacker to force the calculation of a large number of hash computations on the tag if they query it with a big \( t_{sys} \). Depending on the size of the epoch, the computation load on the tag becomes insurmountable, which effectively amounts to a DoS attack.

YA-TRAP*, as it was the case in RIP and RIP+, does not offer forward untraceability due to the fact that tag key \( k_{tag} \) is static. As a result, Tsudik proposes an extension to the YA-TRAP* protocol in [27] in order to add that security property. The mechanism used is the tag also updates the key \( k_{tag} \) as \( H^\nu(k_{tag}) \), i.e. the key changes once per epoch. It follows, though, that the protocol is not forward untraceable within the period from the last update to the current time.

The last relevant protocol in the family we review, O-TRAP by Burmester et al., was proposed in [78]. In [77] it was re-named O-RAP by the same authors. In this approach, the primary key of the system’s database is a pseudonym \( r_{tag} \). The pseudonym \( r_{tag} \) is renewed on the tag by the tag itself at each protocol run, and accordingly by the system at the corresponding tag entry if the session ends successfully. \( r_{tag} \) is sent in clear to the system, which provides constant-time identification in normal operation. Unfortunately, as we will see, only an illegitimate interrogation is needed to desynchronise tag and system, and then the complexity of the next identification becomes linear in the number of tags. Nevertheless, it has to be pointed out re-synchronisation is obtained after each successful authentication.

The protocol is quite simple and can be described as follows:

1. The system sends a nonce \( r_{sys} \) to the tag.

2. On reception, the tag computes \( H(k_{tag}, r_{tag} || r_{sys}) \), which yields \( r \) and \( h \). Then, the tag sends \( h \) and \( r_{tag} \) to the system. Finally, \( r_{tag} \) is assigned the value of \( r \).

3. Once the system has received the message from the tag, it just has to conduct a constant-time search for the received \( r_{tag} \), as its database has this value as its primary key. Two situations can occur:

   (a) If \( r_{tag} \) is found, the system checks \( h \) to authenticate the tag.

   (b) If \( r_{tag} \) is not found, the system further conducts an exhaustive search through its database to locate the tag entry satisfying \( h \). If the entry is found, the tag is authenticated, and \( r_{tag} \) is assigned the value of \( r \) in the corresponding database entry. Note \( h \) and \( r \) are obtained in the same way as the tag did. It is important to highlight this re-synchronises the tag.

We note the protocol is not forward untraceable because if the tag is compromised the attacker can re-construct past messages sent by the tag from \( k_{tag} \) and the data exchanged, and thus link them.

Finally, we also point out a tag can be traced by the time taken by the system to complete its identification, which is linear in the number of tags if the tag has been desynchronised and otherwise constant-time [38].
Countermeasures such as constant execution, which are relevant in the smart card field [122] are generally discouraged in the low-cost RFID case [38], because the protocol would become non-scalable.
Appendix C

Analysis of RFID Security Protocols

C.1 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol

C.1.1 Suggestions for Improvement/Fixes

C.1.1.1 Asadpour et al Protocol

1. The database stores a table of tuples <identifier, shared secret key, ticket>. Two entries are stored per tag for desynchronisation resilience. In addition, there is a ticket pool containing those tickets which are not currently assigned to any tag.

   *We observe the structure is, in essence, very similar to our modified Alomair et al protocol. Their ticket plays the role of our pseudonym. In addition, their identifier, which we do not store, could be removed from their scheme without loss of neither security nor privacy, as the authors themselves note in the same paper [80].*

2. The tag stores <identifier, shared secret key, ticket, counter>. That is again essentially equivalent to our modified version. In our version, the identifier is not stored on tag, and the counter is removed. Nevertheless, we have just observed the identifier can be removed. In addition, the basic functionality of the counter, as we will see in the description of the protocol below, can be played by our PRNG.

3. The protocol is as follows:

   (a) The system sends a random nonce $r_1$ to the tag. *This does not change with regards to our modified version of the Alomair et al protocol.*

   (b) On reception, the tag adds 1 to its counter $n$. Then, it computes $r_2 = h(k, n)$ and $r_3 = h(k, T_b, r_1, r_2)$, where $k$ is the shared secret key, and $T_b$ is the current ticket on tag. Finally, it sends $<T_b, r_2, r_3>$ to the tag.

   *This first message is very similar to ours.* The ticket plays the role of our pseudonym. $r_2 = h(k, n)$ would be our new nonce $r_{tag}$ to guarantee freshness of the third message. Finally, $r_3$ plays the same role as our $r'$, as it is used by the system to authenticate the tag.

   (c) Then, the system:

      1. Locates the database entry corresponding to the tag by the received $T_b$ in minimal time, as tickets are the primary key of the table in the database. *This is essentially the same as in our version.*

      *From this point in the protocol onwards, both protocols continue to bear strong resemblance, as can be observed below from the components of the messages and their functionality. However,*
C.1 Second Case Study: Alomair, Clark, Cuellar and Poovendran’s Protocol

as far as our version is concerned, essentially, we have not varied the original Alomair et al protocol. Consequently, similarities also concern the original Alomair et al versus the Asadpour et al protocol.

2. Checks correctness of $r_3$ to authenticate the tag.
3. Randomly chooses a new ticket $T_c$ from the pool.
4. Computes $<T_c \oplus h(k, T_b, r_3), h(k, r_3, T_c)>$, and sends it to the tag.
5. The secret key $k$ of the tag is updated to $h(k)$ in the corresponding entry.

(d) Finally, the tag:

1. Checks the received $h(k, T_b, r_3)$ to authenticate the system and to obtain the new ticket $T_c$ from $T_c \oplus h(k, T_b, r_3)$.
2. Checks $h(k, r_3, T_c)$ for integrity of the received new ticket $T_c$.
3. Renews its pseudonym from $T_b$ to $T_c$.
4. Updates its secret key $k$ to $h(k)$. 

127
Appendix D

Classification of RFID Security Protocols

D.1 Analysis of the Classification

D.1.1 Presentation of the Results

1. In general, protocols that have linear complexity in the identification procedure at the server’s side, i.e. non-scalable, have excellent security and privacy properties. We note once more that this category of protocols pay a price for their high level of security and privacy, namely, they are non-scalable. Indeed, their identification complexity is linear in the number of tags, i.e. O(N).

For a prime example of a protocol providing existential untraceability at the cost of becoming non-scalable, we refer the reader to section 4.4, where the C² scheme is examined in detail as a case study. The PEPS protocol [75] and the C² scheme [13] are two examples of linear complexity protocols under this pattern. They satisfy all security and privacy properties in the tables. The Rhee et al protocol [23] is also another example, but it does not satisfy forward untraceability. The reason why the Rhee et al protocol does not satisfy forward untraceability is addressed in the next observation in this list.

2. Protocols that do not refresh tag’s secrets do not offer forward untraceability.

This is the reason why the Rhee et al protocol [23] does not offer forward untraceability. In this protocol, if a tag is compromised, and thus its static secret is captured, it is easy for the attacker to reconstruct past messages sent from the tag. The reason for this is that the only secret information involved is never updated, and it has been captured. Consequently, the attacker can link past runs of the tag, and thus forward untraceability is not satisfied.

We highlight that secret information must be updated using an irreversible computation, if forward untraceability is to be provided. That, though, is the object of our next point in this list.

The same applies to the Molnar and Wagner protocol [26], and the three protocols from the YA-TRAP family we have included, namely RIP [77], RIP+ [77] and O-RAP [77].

3. In general, protocols that refresh tag’s secrets using a one-way function provide forward untraceability.
Protocols in the tables that agree with the general pattern include: the already mentioned modified versions of the Henrici and Muller protocol and the Alomair et al protocol; the $C^2$ scheme [13] and its modified version in this thesis; the PEPS protocol [75]; the two proposals by Dimitriou [24]; and the Avoine et al improvement to OSK [58].

It is important to note that refreshment of secrets is a necessary condition to provide forward untraceability, but not a sufficient condition. If forward untraceability is to be provided, secrets must be refreshed so that it is not possible for an attacker to obtain past secrets from the current captured ones. For instance, a one-way function such as a cryptographic hash function can be used for that purpose, so that it is not feasible for the adversary to reverse the computation.

The Henrici and Muller protocol [11] is an example of a protocol that refreshes its secret, but does not use an irreversible computation. Instead, the secret is updated to the result of XORing it and a nonce which has been just received. Consequently, the previous secret can be recovered from the current one. As the attacker is also assumed to be in possession of all previous messages, it is easy for him to reconstruct past messages from previous secrets and corresponding message contents. Therefore, past messages can be linked to the same tag, and thus the protocol does not offer forward untraceability. The modified version of the protocol presented in this thesis fixes that weakness.

Another case of a protocol that does not offer forward untraceability despite the fact that it refreshes its secrets is the Alomair et al protocol [12]. In this case, the weakness is due to the consequences of an attacker learning the current secrets of a tag. These effects are examined in section 4.3.2, and lead to the loss of forward untraceability. The modified version of the protocol presented in this thesis fixes that weakness as well.

4. Sublinear protocols with reader-triggered refreshment do not offer existential untraceability.

It must be highlighted, though, that these protocols could offer existential untraceability, but then they would become non-scalable. We elaborate on this statement below. By way of example, we consider the typical sublinear protocol that refreshes tag’s secrets:

(a) *That typical protocol does not satisfy existential untraceability.* In such a protocol, a hash of the identifier ID, $h(\text{ID})$, is sent to the reader as part of the response from the tag in the second message of the protocol, just after the request from the reader. That makes the time required to identify the tag minimal, because the server stores $h(\text{ID})$ as a column in the database. The tag, though, can then be traced between two legitimate authentications, as the identifier is not refreshed in between, i.e. existential untraceability is not provided.

(b) *The protocol can trivially satisfy existential untraceability if it becomes non-scalable.* It would be trivial to randomise the tag’s response and replace $h(\text{ID})$ with $<r, h(\text{ID},r)>$, where $r$ is a nonce generated by the tag. Then, existential untraceability would be satisfied, because the responses
D. CLASSIFICATION OF RFID SECURITY PROTOCOLS

from the tag would be different at each interrogation, and unpredictable to an attacker. Unfortunately, the server would have to perform an exhaustive search to locate the tag entry, where a hash computation would have to be carried out per entry. That would turn the identification complexity linear in the number of tags, i.e. $O(N)$, and thus the protocol would become non-scalable.

The Henrici and Muller protocol \cite{11} exemplifies the pattern identified. It behaves as a typical sublinear protocol that refreshes tag’s secrets only after the tag has authenticated the system, and therefore it does not provide existential untraceability. This is also the case in our modified version of the protocol. We made no attempt to provide existential untraceability to the protocol because we wanted it to remain scalable. The second proposal by Dimitriou \cite{24} exhibits the same behaviour.

The Alomair et al protocol \cite{12} is also a sublinear protocol refreshing tag’s secrets only after the tag has authenticated the system. As a result, it should be possible to trace tags between two valid authentications. The protocol tries to mitigate that issue, though. Unfortunately, as we explain below, unsuccessfully:

(a) A monotonically increasing counter $c$ is introduced into the protocol. It is important to note the counter is incremented at each interrogation, unlike the ID, which is updated after a successful authentication only. The maximum value of the counter $c$ is set to a fixed integer $C$. In addition, the scheme still aims at minimal identification complexity, i.e. $O(1)$, at the cost of pre-computing all possible $h(ID,c)$ values.

(b) At each interrogation, the counter $c$ incremented and hashed together with the identifier ID, yielding $h(ID,c)$, which is then sent to the reader. The system can identify the tag in minimal time by the received $h(ID,c)$, as it is found pre-computed in its database. The main idea is that an attacker sees a different and unpredictable $h(ID,c)$ value at each interrogation, and thus existential untraceability is provided.

(c) Unfortunately, it is possible for an attacker to query the tag $C$ times, collect all its responses, and then track the tag until the next successful authentication.

(d) Therefore, this is one of the reasons the protocol does not provide existential untraceability. It must be noted the counter $c$ is also an important weakness in the protocol, as examined in section 4.3.2, where the reader is also referred to for more details.

Our modified version of the Alomair et al protocol does not provide existential untraceability either. Again, no attempt is made to add it to the protocol, as that would turn it non-scalable.

It is also interesting to mention the $C^2$ scheme \cite{13}, even though it is linear, and only sublinear protocols are concerned by the pattern being discussed in this point. The reason for that is that the authors themselves recognise the scheme provides existential untraceability at the cost of non-scalability. Our modified version of the protocol takes the opposite design decision.

The PEPS protocol \cite{75} is also linear and provides existential untraceability.
This pattern observed in the presented tables concerns directly to those sublinear protocols where secrets are refreshed only when the tag has authenticated the reader. For that reason, protocols which do not refresh its secrets are not discussed, such as the Rhee et al protocol [23], the Molnar and Wagner protocol [26], and the three protocols from the YA-TRAP family we have included, namely RIP [77], RIP+ [77] and O-RAP [77].

Finally, we note the Avoine et al improvement to OSK [58] and the first proposal by Dimitriou [24] provide existential untraceability, but in this case, the tag self-refreshes its secrets at each interrogation, so that the responses from the tag are randomised, i.e. they look random to an attacker, even between two valid authentications.

5. **All protocols in the tables provide identification.**
   Identification is a minimum requirement for any RFID protocol. The system must be able to identify the tags it manages, otherwise the RFID system serves no purpose.

6. **All protocols in the tables provide anonymity.**
   This privacy property is satisfied by all protocols in the tables. That suggests it is a desirable and important property for any general RFID security protocol.
   The consequences of the lack of anonymity include the leakage of sensitive information about an individual holding a tagged product. For instance, the possession of a book revealing a controversial political stance, or the need for a drug associated to a serious condition. We refer the reader to subsection A.2.3, where we studied the property, for further information.
   A much used mechanism to provide anonymity amongst RFID security protocols is the transmission of a cryptographic hash of the identifier, instead of transmitting it in clear, which would clearly breach the property. In fact, the hashed identifier can be seen as a form of pseudonym. Examples include the two proposals by Dimitriou [24], and the Henrici and Muller protocol [11] and our corresponding modified version.

7. **Most protocols in the tables provide mutual authentication.** Mutual authentication is identified in the tables as a usual design objective. Nevertheless, whether it is required or not will depend on the target application. Some applications might not need security beyond simple identification, e.g. identification of lost pets. However, many others would require some form of authentication (unilateral or mutual), as in anti-counterfeiting applications. Nevertheless, there are some cases in the tables that do not provide mutual authentication. We briefly comment on the reasons for that below:

   (a) The first proposal by Dimitriou [24] does not provide any form of authentication. The reason for that is that no freshness mechanism is used, which is a necessary condition to ensure authentication. Nonetheless, we note this protocol has been included for historical interest only. In particular, because Dimitriou built on it to present its second proposal in the same paper [24]. In this work, we have found this second proposal to be an interesting representative of protocols.
(b) The original OSK [25] is an identification protocol only. Then, Avoine et al extended it to provide tag to reader authentication. This Avoine et al improvement to OSK [58] is the protocol included in the tables. Later on, in [162], Avoine proposes the addition of reader to tag authentication to the protocol. For completeness, we mention this new version is vulnerable to a traceability attack, noted and fixed in [38], where the reader is referred to for further details.

(c) The three protocols from the YA-TRAP family we have included do not provide mutual authentication either, and we briefly mention the underlying cause for that. Firstly, RIP [77] offers no authentication, but it was conceived as an identification protocol only. Secondly RIP+ [77] and O-RAP [77] offer tag authentication only, but that was the design aim as well.

8. Most protocol surveyed provide universal untraceability.

Firstly, we note universal untraceability is weaker than existential untraceability. To provide universal untraceability protocols do not need to prevent the tracking of tags between two successful authentications, whereas to provide existential untraceability protocols must prevent tracking within that period as well.

Having made that observation, it is noted that universal untraceability is provided by most protocols in the tables, whereas existential is provided by around half of them only. Furthermore, the group of protocols that provide existential is, inherently, a subset of those providing universal. That would suggest that universal untraceability must be, in general, the minimum level of untraceability a protocol designer aims to achieve. For instance, the designers of the C² scheme [13], as we have already noted, recognise that they had universal untraceability guaranteed, but they decided to further add existential untraceability at the cost of making the protocol non-scalable.

Then, we comment on the provision of universal untraceability in:

(a) Protocols where tags refresh their secrets. In this case, it is usual that responses from the tag consist of nonces and/or hashes that include secrets just updated. Those messages look random to an attacker. Examples include our modified version of the Henrici and Muller protocol, the C² scheme [13] and our modified version, the two proposals in Dimitriou [24] or the Avoine et al improvement to OSK [58].

We find it interesting to note the case of our modified version of the Alomair et al protocol. The tag response includes a pseudonym which is sent in clear to the reader. However, that does not constitute a weakness, because the pseudonym is renewed at each successful authentication. The new pseudonym is chosen at random from a pool maintained by the system and sent in encrypted form to the tag. The current pseudonym of a tag, then, bears no correlation to the previous one, and thus universal untraceability is provided. The reader is reminded that secrets and pseudonyms are updated in this protocol only after a successful authentication has taken place.

132
(b) Protocols where tags do not refresh their secrets. In this case, even though secrets are not updated, it is also usual that the response from the tag looks random to an attacker as well. The reason for this is that it is typical that the response consists of hashes which include nonces, apart from static secrets. A clear example is the Rhee et al protocol [23]. The RIP [77] and RIP+ [77] protocols are examples as well. The response from the tag consists of hashes which include, apart from the static secrets, either random nonces or a monotonically increasing counter which is used once only.

(c) Finally, some exceptions exist amongst the surveyed protocols. For instance, the Henrici and Muller protocol [11] does not offer any untraceability, when it should fall into the group of protocols refreshing their secrets above. The reason for that was analysed in section 4.2.2, where the security properties of the protocol were studied. Succinctly, an attacker can desynchronise tag and database. The result is that no more successful authentications are possible. Consequently, the secrets are no longer updated and the tag is definitively traceable.

9. Most protocols in the tables provide desynchronisation resistance. The importance of this property for any RFID security protocol is supported by this observation. The consequences of a successful desynchronisation attack might include that the system will not be able to identify the involved tag any longer. See section 4.2.2, where the security analysis of the Henrici and Muller protocol is conducted, for an example.

The protocols in the tables can be subdivided into the following groups, according to the way desynchronisation is addressed:

(a) The database stores two pseudonyms per tag, current and old. If desynchronisation occurs, the database can locate the correct tag entry by the old pseudonym in the next authentication session. As examples of protocols using this technique successfully, the Alomair et al protocol [12] and our modified version can be named.

(b) The database stores the current pseudonym only, and if needed, the next one is calculated on-the-fly. In this case, if desynchronisation occurs, in the next authentication session the database can locate the correct tag entry by calculating the next pseudonym to the current stored one. In this work, the $C^2$ scheme [13] has been identified as the only protocol using this technique.

(c) The database stores a hash chain for each tag. In protocols such as the Avoine et al improvement to OSK [58], the database stores a hash chain for each tag. This is inherent to the protocol family. The tag replies at each interrogation with the next element of the chain, and the system explores each chain up until a pre-set maximum. That implies desynchronisation resistance is limited to that maximum. Once that limit is reached, the tag becomes definitively desynchronised. Unfortunately, that limit can also be maliciously exhausted by an attacker. For further details, the interested reader is referred to subsection B.3.3.1, where the scheme is reviewed.

(d) Protocols where secrets are not refreshed. In general, these protocols are desynchronisation resistant because there is no need to maintain any synchronisation for the secrets, as they are static. Examples include the Rhee et al protocol [23] and the Molnar and Wagner protocol [26]. It is
D. CLASSIFICATION OF RFID SECURITY PROTOCOLS

important to note that secrets might not be the only elements that a given protocol intends to maintain synchronised, as can be seen in the following point.

(e) Desynchronisation in the YA-TRAP family. Protocols such as RIP [77] and RIP+ [77] are easily desynchronised. Secrets are not refreshed, but the counters used are. For more information, the reader is referred to subsection B.3.3.3, where the YA-TRAP family is examined. A desynchronisation attack applicable to both protocols is succinctly described below:

i. It is relevant for the description of the attack that we observe that the tag receives a monotonically increasing counter from the system in the first message of the protocol. The tag expects the value of the received counter to be no greater than a pre-set maximum, but greater than the last valid received one, which is stored in the tag. Consequently, a given counter value can be used once only.

ii. In the first place, the attacker sends the tag a counter value which equals the maximum one.

iii. The tag accepts the value and updates its current counter value to the received one, i.e. to the pre-set maximum.

iv. As a result, the tag no longer accepts any other value from the system as valid, and thus becomes definitively desynchronised.

O-RAP [77] does not behave in the same way as RIP and RIP+. The protocol can also be easily desynchronised, as an attacker just needs to interrogate the tag once. However, the tag re-synchronises at the next successful authentication. Therefore, in this work, it is assumed that desynchronisation resilience is provided. The interested reader is referred to subsection B.3.3.3, where the protocol is reviewed, for further details.

(f) Finally, some exceptions exist amongst the surveyed protocols. For instance, the Henrici and Muller protocol [11] uses the first technique above in this point, i.e. the database stores two pseudonyms per tag, current and old. However, it does not provide desynchronisation resistance. The reason for that is that a weakness in the protocol exists, which makes it possible for an attacker to mount a desynchronisation attack. The attack is based on the manipulation of the last message from the system [79]. The reader is referred to section 4.2.2, where a security analysis of the protocol is conducted, for further details and a description of the attack.

The results obtained confirm the paramount importance of conducting an analysis of the requirements of the target application before the design of the security protocol starts. The analysis must include the expected security, privacy and performance levels, together with a risk assessment. Then, requirements and accepted risk would point to the adoption of a certain type of protocol and tradeoff between the desired properties.

For instance, if scalability is not a concern in the intended application, e.g. access control for a small to medium-sized company, it has been shown a linear complexity protocol, i.e. O(N), can offer an excellent degree of security and privacy. Otherwise, the protocol should be sublinear and the business should tradeoff scalability against security and privacy.
D.1 Analysis of the Classification

D.1.2 Example Application

The requirements of the target application at a major retail store, are:

1. A large number of items must be tagged. This is a realistic assumption in an RFID scenario such as a major retail store or a large library. For instance, the Queens Library System tags millions of books, DVDs and CDs [163]. Back in 2008, around 6.5 million items were tagged as part of the initial deployment, and it was calculated that about one million new items would be additionally tagged per year from then onwards.

2. This major retail store is concerned about the privacy of its customers. In addition, they are well aware of the existence of movements against the use of RFID based on the Big Brother concern, e.g. the the Spychips.com website [52]. In particular, they want to ensure:

   (a) It should not be possible for a malicious third party to scan the tags attached to any sold item and learn its type. That could reveal sensitive information about the customer, e.g. his or her political stance by the title of a book he or she might have bought at the store. That means anonymity should be provided.

   (b) Furthermore, it is also important the customer is not tracked by the tags attached to the items he or she has bought. That guarantee should extend to the period between valid authentications, i.e. existential untraceability is desired. The reason for that is that in this scenario, once an item has been sold at the checkout, it might not be legitimately authenticated again. The organisation identifies as an exception the possibility of a customer returning an item back to the store.

   (c) The major retail store has also considered a last possibility for the privacy of the customer to be breached. As we observed in the last point, a malicious third party, maybe an insider, might have an interest in tracking the customer by the tag attached to the item he or she bought. We already noted existential untraceability is required. Therefore, tracking was not possible while the tag was in the customer’s hands. Indeed, the tag replied at each interrogation with a value which looked random to the attacker, even during the period between valid authentications. However, if the item is returned to the store, the adversary can access it once it is at the shelf again. Alternatively, the adversary might steal the item directly. In both cases, he or she could tamper with the tag, as it is low-cost and therefore not tamper-resistant. If forward privacy is not provided, that adversary could track the previous owner of the item by past readings of the tag while it was in the previous owner’s hands.

   The store wants to guarantee not even the least concern about privacy can be raised by activist groups or individual customers. Therefore, forward privacy is also required.

   (d) The major retail store would also desire the highest possible level of desynchronisation resistance. It is well-known to the organisation that the consequences of a successful desynchronisation attack might include, depending on the type of protocol used, that the tag cannot be identified by the system any longer, unless out-of-band measures are taken. It is the organisation’s aim to
minimise the number of such out-of-band interventions, should they be needed. Nevertheless, desynchronisation resistance is assigned a lower priority level than the one assigned to privacy.

Using the results obtained in the analysis of the classification, and the requirements of the organisation, a step-by-step decision-making procedure can be described to obtain a suitable RFID security protocol for low-cost tags:

(a) The organisation states a large number of items must be tagged. As a result, scalability is required. Therefore, protocols featuring linear complexity, i.e. $O(N)$, should be excluded. In the tables, that would exclude the $C^2$ scheme [13], the PEPS protocol [75] and the Rhee et al protocol [23]. In addition, the first listed observation in our analysis of the classification indicates that the business has to consider the loss of some degree of security and/or privacy to obtain scalability. As we progress on the characteristics of the protocol, that tradeoff should become evident.

(b) The provision of anonymity should not be an issue, as it is generally provided by any RFID security protocol for low-cost tags. That is also stated in our sixth conclusion drawn from the analysis of the classification.

(c) The major retail store requires existential untraceability as well. Consequently, based on our fourth listed observation in the analysis, sublinear protocols, i.e. under $O(N)$, with reader-triggered refreshment cannot be considered. In the tables, the only protocols remaining after that criterion has been applied are: the Avoine et al improvement to OSK [58], and the three protocols from the YA-TRAP family, namely RIP [77], RIP+ [77] and O-RAP [77]. We note that we do not consider the first proposal in Dimitriou [24] because it was included in the analysis for historical interest only, and shows evident weaknesses.

(d) The provision of forward privacy points us to the second and third listed observations in the analysis. The second one notes that protocols that do not refresh tag’s secrets do not offer forward untraceability. For this reason, the three members of the YA-TRAP family are discarded, and the only option that is available is the Avoine et al improvement to OSK [58]. This protocol does provide forward untraceability. It must also be highlighted that the protocol satisfies our third listed observation as well. That observation states that, in general, protocols that refresh tag’s secrets using a one-way function provide forward untraceability. Indeed, a cryptographic hash function is used.

(e) Finally, the highest possible level of desynchronisation resistance is also desired. That directs us to our last observation in the analysis, where we discussed the different techniques used by the surveyed protocols to address desynchronisation. In particular, our chosen protocol, the Avoine et al improvement to OSK [58], stores a hash chain for each tag. At identification time, the system explores the hash chain up to a pre-set maximum length. The business should be warned that the level of desynchronisation resistance offered depends on that limit. Synchronisation, though, could be re-gained if out-of-band action was taken to re-synchronise both tag and database. If that was not functionality acceptable for the business, another model of protocol should be considered.
Appendix E

Tools for the Automated Formal Verification of Security Protocols

E.1 Motivation for the Formal Verification of Security Protocols

It is well known there is an increasing use of network infrastructures in today’s distributed world, being the Internet a major example. Network protocols are distributed programs that are needed to enable reliable communication in order to solve such problems as efficient and correct routing and transmission errors.

Moreover, it is often the case those network infrastructures feature untrustworthy channels. Consequently, network protocols must guarantee that a set of required security properties is satisfied. Those properties include entity authentication or privacy properties such as anonymity, to name just a few. Usually, these security protocols use cryptographic building blocks such as encryption, hash functions, or digital signatures. Major examples of those protocols on the Internet are the Transport Layer Security (TLS) [164] or the Secure Shell (SSH) [165, 166, 167, 168].

Unfortunately, security protocol design is an error-prone task. We can find numerous examples of protocols which have been shown flawed after their publication. Two well-known examples are:

1. The Needham-Schroeder Public-Key Protocol [39]. Seventeen years after its publication, it was broken and fixed by Lowe using the Casper/FDR tool [40]. For completeness, it must be noted the adversarial model assumed by Lowe was different than the one in the original publication. Nevertheless, the flaw was not noticed in a long period of time.

2. In 2008, the Single Sign-On (SSO) protocol used in the SAML-based Single Sign On for Google Applications was shown flawed by Armando et al using the tool SATMC [91]. Single Sign-On enables users to sign in once only and then access several applications.

As a result, formal verification techniques are needed. Protocols featuring a proof of security significantly increase confidence and avoid the consequences of those flaws, which might include financial and public-relations damage to the involved companies.
E. TOOLS FOR THE AUTOMATED FORMAL VERIFICATION OF SECURITY PROTOCOLS

E.2 Background on Formal Verification of Security Protocols

The main idea underlying formal verification techniques is twofold. Formal mathematical methods are needed to firstly, model the protocol and the properties it should satisfy, and secondly, analyse the model.

Unfortunately, the problem of protocol security is undecidable, in general. Moreover, it remains undecidable even when significant restrictions are imposed on protocols [169]. Clearly, undecidability poses a real challenge to automated tools. To address undecidability, restrictions can be applied. For instance, in [92], it is shown protocol insecurity is NP-complete with a bounded number of sessions, regardless of the size of the messages.

Formal methods can be subdivided into three main groups, namely methods based on logics, model checking, and theorem proving. Even though an in-depth treatment is outside the scope of this work, a succinct review is provided below for completeness and convenience. The interested reader is referred to the references provided for further information:

(a) Methods based on logics: To address this group, we review the most well-known and representative logic within it, namely the BAN logic, named after their authors Burrows-Abadi-Needham [93]. A description of its main characteristics is provided below:

i. The logic provides a series of formulas that model the initial assumptions of the protocol. For instance, agent A believes that agent A has public key $k_A$.

ii. Those formulas also model the protocol goals. For instance, at the end of the protocol run agent A believes that key $k_{ab}$ is a key only known to agent A and agent B.

iii. Ideally, each message in the protocol is expressed as a formula as well. For instance, a particular message could be this formula: $X$ encrypted under key $k$.

In addition, after a message from agent A to agent B containing formula X has been received, the following formula holds: agent B has received a message containing formula X.

iv. Furthermore, a set of rules is also defined which shows how the beliefs of the agents change as the protocol run evolves. For instance:

- If agent A has received a message containing formula X encrypted under key $k$, and
- Agent A believes key $k$ is a secret key shared between agent A and agent B only, then
- Agent A believes that agent B sent a message containing formula X at some point.

v. Firstly, then, the initial assumptions, the protocol goals and the messages have to be modelled.

BAN logic can provide useful insights to the protocol designer. For instance, if protocol goals are not achieved, that might point to the need for additional assumptions. It is also possible the protocol designer notices the same goals can be achieved in a simpler way.

However, BAN logic has its limitations as well. For example, it analyses authentication only,
but not other properties such as anonymity. The adversary model is also limited as agents are assumed honest.

(b) **Model checking:** The main idea is the analysis of protocols through a state space model. Typically, transitions represent actions of honest agents or the intruder, and states represent the knowledge of the intruder and the local state of honest agents. A logic formula is assigned to each state in the model, and a logic formula is also assigned to the property to be analysed.

Ideally, the model checker would check that the property holds in all states to ensure correctness. If the property did not hold in all states, a trace leading to the attack state would be produced. It must be taken into account, though, that the state space can be very large, or even infinite. For instance, the number of sessions can be potentially unlimited. If the number of sessions is bound, though, correctness cannot be ensured, as it might be possible that an attack is missed because it needs the execution of a number of sessions greater than the restricted number. Even in that situation, the model checker can still be very helpful to find attack traces.

It must be observed that model checkers which take into account an unbounded number of sessions have also been developed, e.g. Proverif, which is reviewed in subsection E.3.2. However, due to the undecidability of the problem of protocol security in an scenario where all states are taken into account and the number of sessions considered is unbounded, termination is not guaranteed [113].

Finally, it is noted that several methods have been developed to reduce the size of the state space without affecting the results of the analysis. A deeper treatment of this topic is outside the scope of this work, but the interested reader is referred to the AVASP Tutorial on Automated Validation of Security Protocols for further details [170].

(c) **Theorem proving:** In order to describe theorem proving, we use a salient example of an interactive theorem prover, namely Isabelle [94], which was written by Paulson et al. These are some essential points about Isabelle, which are highly representative of the general theorem prover:

i. In Isabelle, protocols are modelled as a set of traces. Traces are lists of events, and occur when agents run the protocol. An example of an event is *Agent A sends message X to agent B*. The intruder is one of the agents. Each protocol step is modelled as a rule, which has premises and conclusion. The conclusion of a rule adds new events to a trace. Target security properties are modelled as theorems, which are proved by induction on traces.

ii. In [171], automated theorem proving is defined as a research for using computers to reason in formal languages based on various logics. Interactivity in theorem provers depends on the particular theorem prover considered. For instance, even though Isabelle is strongly automated, it is said that it is an interactive theorem prover because complete automation is not achieved. Human intervention is required to guide the process. This is often the case when conducting proofs with theorem provers.
iii. Theorem provers in general, and Isabelle in particular, can address an unbounded number of sessions and agents. Isabelle, for instance, uses the inductive method [172], which copes with infiniteness.

iv. It can also be observed the method focuses on proving correctness rather than finding attacks. The reason for that is that it aims at proving theorems, i.e. modelled security properties. Nevertheless, if correctness is not proved then that can provide some helpful insights towards possible attacks.

E.3 Overview of Surveyed Tools

E.3.1 AVISPA/AVANTSSAR

In this subsection, the AVISPA tool [41] and its successor, the AVANTSSAR platform [6], are reviewed.

E.3.1.1 The AVISPA Tool

The AVISPA Tool is a push-button tool for the Automated Validation of Internet Security Protocols and Applications [173]. It includes:

(a) A modular formal language to model security protocols and the security properties to be verified. The language is HLPSL, short for High Level Protocol Specification Language.

(b) Four different validation backends which offer a number of automatic protocol analysis techniques.

The architecture of the AVISPA tool is shown in figure E.1 on page 141. The user of the tool models a security protocol and the security properties to be verified. HLPSL is used for that. That model is input to the tool. Then, the HLPSL2IF translator translates the HLPSL specification to IF, short for Intermediate Format. The IF specification is then input to up to four backends, which automatically verify it.

The backends receive an IF specification, which models an infinite-state transition system. Then, they search the system to find attack states, i.e. states where the security properties to be verified are violated. The four backends are described below. We refer the interested reader to the papers cited next to the name of the backend for an in-depth treatment:

(a) The On-the-Fly Model-Checker, OFMC for short: [174] It explores the input state transition system in a demand-driven way. Both falsification, which implies the identification of attack traces, and bounded session verification can be performed. OFMC’s strength is based on the implementation of a number of search-space reduction techniques with neither introduce nor exclude attacks. It also employs several efficient search heuristics. Finally, it is important to note that it supports the specification of algebraic properties and cryptographic operators [173].
E.3 Overview of Surveyed Tools

(b) The Constraint-Logic-based Attack Searcher, CL-AtSe for short: [175] It applies constraint solving to perform falsification and bounded session verification [173]. Both were also performed by OFMC, as we observed in the above paragraph. The algebraic properties of XOR and exponentiation are considered. In addition, optimisations are also made, which accelerate attack search. For instance, ignoring useless messages or reducing/removing redundancies in the symbolic protocol execution.

(c) The SAT-based Model-Checker, SATMC for short: [176] As the two previous backends, it performs both falsification and bounded session verification. A propositional formula is built, which incorporates all possible traces of bounded length in the protocol [177]. The formula is then input to a SAT solver, which determines whether it is satisfiable. Any model found represents an attack [173].

(d) The Tree Automata based on Automatic Approximations for the Analysis of Security Protocols, TA4SP for short: [178] It performs protocol verification for an unbounded number of sessions. An approximation of the intruder knowledge is made by using tree languages and rewriting. With regards to secrecy properties, it proves whether a protocol is safe or not, by over-approximation or under-approximation, respectively [173].

Verification in all backends assumes perfect cryptography, i.e. the adversary cannot break cryptography unless they are given the corresponding keys. In addition, a Dolev-Yao intruder is also assumed [104], i.e. they have full control of the network, including eavesdropping, injection and blocking of messages.
We introduced the Dolev-Yao intruder in subsection A.1.2 when describing the communication channel between reader and tags.

The output of the backend indicates whether the protocol goals were achieved. Furthermore, if an attack trace was found, it is shown.

The **AVISPA tool offers:**

2. **A package** which can be downloaded and installed on the user machine.

With regards to the web interface, the user can either load their own HLPSL specification or one of those included in the AVISPA Library of protocols. The loaded protocol can then be modified. After that, depending on the mode of operation:

   1. **Basic mode:** The protocol is verified by all backends when the user executes the verification.
   2. **Expert mode:** The user can choose the backend they are interested in and specify parameters.

The output of the backends is displayed. If an attack is found, it can be shown in text or graphically by means of a message sequence chart, msc for short [179].

The **language used by the AVISPA tool to model protocols and their security properties is HLPSL.** The main features of the language include. See figure E.2 on page 143 as well:

1. **It is a modular role-based language:**
   i. **Basic role.** The actions of each participant are specified in a different module. That module is called a basic role. See (a) in the figure. Each basic role shows the initial information of the agent playing it (parameters), its initial state, and how the current state changes as a result of the exchanged messages (transitions).
   ii. **Composed roles.** The first type of composed role is the session role, see (b) in the figure. It instantiates basic roles, which usually execute in parallel, in its composition section. The objective is to describe protocol sessions. The second type of composed role is the environment role, see (c) in the figure. It is a top-level role. In its composition section, it instantiates the sessions to be executed. The intruder might take part in those sessions, as a participant. The initial intruder knowledge is also specified in the environment role.

2. **Goal security properties can be specified.** The supported properties are secrecy, weak authentication and strong authentication. The difference between weak and strong authentication is that the former offers no replay protection.

   Security goals are specified by means of both:
   i. Including goal facts in the transitions of basic roles. For secrecy, for instance, the goal fact indicates the value to be kept secret and the agents sharing the secret. In addition, an identifier for the particular goal is also specified in the goal fact.
ii. Declaring the security goals in the goal section, see (d) in the figure. Each goal declaration includes the identifier of the goal indicated in the corresponding goal fact(s).

(c) It is possible to specify algebraic properties, such as concatenation, XOR or exponentiation. Nevertheless, that does not mean all of the backends necessarily support them.

(d) Channel specification, even though only Dolev-Yao channels [104] are supported [106].

(e) Cryptographic bases: Support for symmetric keys, public/private keys, hash functions and nonces [4].

For a complete and succinct list of the characteristics of the language, we refer the interested reader to [4]. For an introduction to the language, to [180], and for further information on HLPSL to [106].

![Figure E.2: HLPSL Main Elements. (a) Basic Role. (b) Session Role. (c) Environment Role. (d) Goal Section. Picture taken from [5].](image)

In this subsection, the AVISPA Tool [41] has been reviewed. In the following subsection the AVANTSSAR Platform [6], its successor, is addressed.

### E.3.1.2 The AVANTSSAR Platform

The AVANTSSAR Platform [6] is intended to provide a means to formally specify and automatically validate trust and security of service-oriented architectures (SOAs, for short) and other applications in the Internet of Services (IoS, for short) [6]. It is the successor of the AVISPA Tool [41], and extends it to include trust and security in SOAs and the IoS. It **includes**:

(a) Three different validation backends, which are new versions of those also found in the AVISPA Tool, namely OFMC [174], CL-AtSe [175], and SATMC [176], see subsection E.3.1.1 for further information on the backends. They offer a number of automatic reasoning techniques.
E. TOOLS FOR THE AUTOMATED FORMAL VERIFICATION OF
SECURITY PROTOCOLS

(b) All three backends receive the same input language, namely ASLan [181], short for AVANTSSAR Specification Language. ASLan is low-level and it is not expected that users of the platform model in it. Consequently, the Platform can translate several application-level languages into ASLan. Those languages are:

i. The standard BPMN (short for Business Process Modeling Notation) [182], which is well-suited for users used to Business Process (BP).

ii. ASLan++ [183], developed by the AVANTSSAR Team. It is intended to be an expressive and flexible language which makes it possible to model SOAs succinctly, and specify services at a high level of abstraction. In addition, the models obtained should be easy to understand and build by a wide range of users, regardless of whether they are experts in formal specification methodologies [6].

iii. AnB, short for Alice-and-Bob notation. It is based on an extended Alice-and-Bob notation. In particular, it aims to support algebraic properties [184]. This language is more accessible than ASLan++, and might be preferred by some security protocol/service professionals.

iv. HLPSL++, an extension of the HLPSL [4, 106, 180] of the AVISPA Tool. The HLPSL was reviewed in subsection E.3.1.1, where the AVISPA Tool was addressed. As AnB, HLPSL++ is more accessible than ASLan++. It is also noteworthy HLPSL++ offers support for the specification of communication channels other than the Dolev-Yao model, whereas HLPSL does not.

The architecture of the AVANTSSAR Platform is shown in figure E.3 on page 145. A description of the architecture is provided below. The interested reader is referred to [6] for further information:

(a) The input to the Platform is a model of the available services and a policy which specifies the functional and security requirements of a desired new service.

We believe a short note is needed at this point. The composition of available services to obtain a targeted new service is one of the desired characteristics of SOAs [185]. There are two main approaches to compose services, centralised, which is the one chosen in AVANTSSAR, and distributed. In the centralised approach, which is also called orchestrated, the result of the composition is a new service that orchestrates the component services. In the distributed approach, all services can communicate without mediator.

(b) The Orchestrator, see in figure E.3 on page 145, tries to find a composition of the available services satisfying the policy. The output of the Orchestrator is a targeted service meeting the functional requirements.

(c) The Validator, which encompasses the three aforementioned backends, receives the output of the Orchestrator in order to validate it for security. If the security requirements are met, then the output of the Orchestrator is in turn output as secure by the Validator. Otherwise, the security breach obtained, which includes a counterexample, is fed back to the Orchestrator, which tries to find another orchestration. This process is iterated until a successful orchestration is found, or it

144
is determined that no appropriate orchestration exists. It is also possible for the user of the Platform to skip the Orchestrator and input the targeted service and its goal security properties into the Validator.

(d) The Connectors’ layer, which allows the translation from application-level languages (BPMN, ASLan++, AnB and HLPSL++) into ASLan. See above in this section for further information on those languages. In addition, the connector’s layer performs the translation from the output of the Validator into a form of MSC, short for message sequence chart [179], which is high-level and thus helps understanding of the results.

Figure E.3: The AVANTSSAR Platform Architecture. Picture taken from [6]. It is noteworthy that the AVANTSSAR Platform offers the possibility of modeling other communication channels, apart from the Dolev-Yao [104]. For instance, the HLPSL++ modeling language makes
it possible to specify confidential and authentic channels [6]. That proves advantageous when such channels as SSL have to be modelled.

The backends receive an ASLan specification and corresponding security goals and verify whether those goals are satisfied by the specification. If they are not, an attack trace is shown.

**The AVANTSSAR Platform is offered:**

- *As a SOA (short for service-oriented architecture).* The main idea is to offer the platform components as services. Example components are the Orchestrator, the Validator, the HLPSL++ connector or the ASLan++ connector.

- *As a web interface* at [http://satmc.ai.dist.unige.it/avantssar/platform/web-interface/demo.php](http://satmc.ai.dist.unige.it/avantssar/platform/web-interface/demo.php). Three modes of operation are provided, namely demo, basic and expert. They suit different levels of expertise of the user. It is possible for the user to upload and edit their own specification or choose one from the AVANTSSAR library. Then, the execution can be requested, and the corresponding results examined. Furthermore, in expert mode, the user can completely control the platform functionalities offered [95].

In this subsection, the AVANTSSAR Platform [6] has been reviewed. In the next subsection, Proverif [42] is examined.

**E.3.2 Proverif**

In this subsection, the Proverif tool [42] for the automated formal verification of security protocols is reviewed. The essential characteristics of the tool are:

- *It supports a range of cryptographic primitives,* including symmetric and asymmetric encryption, digital signatures or hash functions [107]. As it is going to be shown in this subsection, protocols can be modelled in the applied pi calculus language [186]. In this language, many other cryptographic primitives can be defined through the use of an equational theory [187].

- *Verification is made with regards to an unbounded number of sessions and an unbounded message space.* Unfortunately, as it was noted in section 6.2 where some background on formal verification of security protocols was provided, the problem of protocol security is, in general, undecidable, even when major restrictions are set on protocols [169]. To analyse in an unbounded number of sessions and message space scenario, techniques that over-approximate the intruder knowledge can be used. We already found this technique in the TA4SP backend [178] of the AVISPA tool, in subsection E.3.1.1. The downside of over-approximations on the intruder knowledge is a false attack can be given. On the other hand, if the property holds on the over-approximation of the intruder knowledge, that implies the property holds on the real intruder knowledge.
E.3 Overview of Surveyed Tools

(c) *It supports the verification of reachability properties, correspondence assertions, and observational equivalence* [107]. As a result, secrecy and authentication can be analysed. Furthermore, privacy properties can be considered as well. We elaborate on this point below in this same subsection on Proverif.

**The architecture of Proverif** is shown in figure E.4 on page 148. A description of the architecture is provided below:

(a) *Input to the resolution:* In the figure, it can be observed that the resolution part of the tool accepts two possible inputs as protocol specifications. In the first one, the protocol is represented as a set of Horn clauses [188], i.e. clauses which have one positive literal, at most. However, that model is considered low-level for non-expert users. As a result, it is also possible to specify the protocol using the applied pi calculus [186], which is based on and extends the pi calculus [189]. The applied pi calculus specification is translated into Horn clauses by the tool.

The pi calculus is a model of concurrent computation. It allows the description and analysis of systems where participants interact amongst each other, and have a changing configuration [189].

The applied pi calculus [186], which is supported by Proverif, is an extension to the pi calculus. It allows the description and analysis of security protocols. Due to a rich term algebra, it can model cryptographic operations. In addition, several security properties can be formulated so as to analyse whether they are satisfied or not by the security protocol. We elaborate on these properties in this same subsection below.

It is also relevant to note the applied pi calculus bears some resemblance to the spi calculus [190]. A significant difference is, though, that the spi calculus has only a limited number of predefined cryptographic primitives, whereas the applied pi calculus makes it possible to define many others through an equational theory [187].

(b) *Output of the resolution:* Three possible outputs can be distinguished for the resolution (see figure as well):

i. *The property is true (and proved):* As it was noted above in this subsection, in order to analyse in an unbounded number of sessions and message space, the tool resorts to over-approximations, which might lead to the reporting of false attacks. However, if Proverif states some given property holds, then that property really holds [191]. In other words, Proverif is sound.

ii. *The analysis does not terminate:* Proverif cannot guarantee termination, in general [188]. That shortcoming is inherent to the undecidability of the protocol verification problem when an unbounded number of sessions and message space is considered, see section 6.2 for further information.

iii. *The property cannot be proved:* When that situation arises, the tool attempts attack reconstruction, which can lead to either:

   * **Potential attack:** A counterexample that falsifies the analysed property.
- *Attack cannot be reconstructed:* Again, that is also unavoidable due to the undecidability of the protocol verification problem for an unbounded number of sessions and message space [107].

![Proverif Architecture](image-url)

**Figure E.4:** The Proverif Architecture. Picture taken from [7].

It is important to note Proverif can address protocols which use the XOR operator or Diffie-Hellman, DH for short, exponentiation. That is possible due to the work in [99, 116], which makes it possible that those protocols can be reduced to the XOR-free, or DH-free case, respectively. In other words, to the case where the algebraic properties of XOR or DH, respectively, do not have to be taken into account.

Proverif supports the verification of:

(a) *Reachability properties:* This capability allows the tool to examine whether a protocol state that satisfies a given requirement can be reached. Secrecy can be expressed in terms of reachability. For instance, secrecy of a given value is preserved if it is not possible to reach a state where the adversary learns the value [107, 113].

(b) *Correspondence assertions:* In order to analyse correspondence assertions, processes of the specified protocol are annotated with so-called events. Then, correspondence assertions allow us to reason about relationships between events. Those relationships can be expressed as "if an event e has been executed, then event e' has been previously executed". In addition, events can have
arguments. Consequently, relationships between the arguments can also be examined [107, 187]. Authentication is an example of a security property which can be verified through correspondence assertions. As a generic example, we consider the case of a protocol which is executed between agent X and agent Y, and the objective is to verify whether authentication of Y to X is satisfied. The idea is to annotate process X and process Y with events. One of the events, e, is situated at the end of process X, and expresses that X believes he has completed the protocol with Y. The other one, e’, is situated at the beginning of Y, and provides evidence that Y believes he has accepted to execute the protocol with X. In a particular example, both e and e’ would be expected to contain arguments, such as a common symmetric key $k$.

Then, authentication of Y to X is expressed as a correspondence assertion, i.e. "if an event e has been executed, then event e’ has been previously executed", which can be verified by Proverif. We refer the interested reader to the Proverif User Manual and Tutorial [107] for further details and examples.

(c) Observational equivalence: If an attacker cannot distinguish between two given processes, it is said that they are observationally equivalent. Therefore, observational equivalence corresponds to indistinguishability [107]. A simple example of two processes which are observationally equivalent is:

i. A process which has one action only, namely it sends value $x$ encrypted under a shared secret key $k$ on public channel $c$. It is assumed the intruder does not know the key $k$.

ii. Another process which also has one action only, namely it sends value $(x,y)$ encrypted under the shared secret key $k$ on public channel $c$. It is assumed $(x,y)$ means $x$ concatenated with $y$.

These two processes are observationally equivalent because the intruder cannot differentiate between the first and the second process by observing the output on channel $c$.

An important consequence of observational equivalence for us in this work is that it makes it possible to reason about untraceability, which is an important property in the RFID privacy field. See subsection A.2.4 for a discussion on untraceability. For instance, in [114], the authors make an attempt to model untraceability in the applied pi calculus, which, as already mentioned, is supported by Proverif. In order to model untraceability, they use observational equivalence. To serve as an example, their definition of strong untraceability is described below:

i. Firstly, two processes P and P’ are modelled such that:

   • P is a process representing a generic RFID protocol. It includes the parallel composition of the following processes: a database DB, an unbounded number of readers R and an unbounded number of tags T. In addition, each tag process T may run an unbounded number of times.

   • P’ is a process representing an idealised version of P, as far as untraceability is concerned. It is defined exactly in the same way as P, but this time each tag process may run once, at most.
ii. Then, it is said that the RFID protocol is strongly untraceable if process $P$ is observationally equivalent to process $P'$. In other words, as far as the intruder is concerned, the protocol looks as if each tag runs once, at most.

As it was the case with AVISPA, Proverif offers:

(a) A web interface at http://proverif.rocq.inria.fr/. The user can either enter his or her own protocol or load one from a list of offered ones. Then, verification can be requested, and the output of the resolution examined.

(b) A package which can be downloaded and installed on the user machine.

In this subsection, the Proverif tool [42] has been reviewed. In the next subsection, the Scyther tool [43] is examined.

E.3.3 Scyther

In this subsection the Scyther tool [43] is reviewed. Scyther is intended to provide verification, falsification and analysis of security protocols [43]. The essential characteristics of the tool are:

(a) It offers support for the analysis of an unbounded number of sessions, and, at the same time, termination is ensured. Nevertheless, if verification for an unbounded number of sessions cannot be achieved, and no attack is found either, a bounded number of sessions is analysed instead [43].

(b) It is possible to generate proofs of correctness. These proofs can be generated as an output of the tool. Subsequently, the proof can be input into a theorem prover which can further check it [113]. In [192], the theorem prover used is Isabelle/HOL [94].

(c) It can avoid reporting false attacks. We have observed the TA4SL backend of the AVISPA tool and Proverif could report false attacks due to the use of the over-approximation technique, see subsections E.3.1.1 and E.3.2 respectively. The Scyther tool, though, does not use over-approximation and as a result can avoid reporting false attacks [113, 193].

(d) Secrecy and authentication properties can be analysed. It is possible to specify secrecy and authentication in the modelling language which is input to the tool. Then, those properties can be verified or falsified [8].

In subsection E.3.2, when reviewing Proverif, it was noted that it could analyse observational equivalence. Observational equivalence allows us to reason about untraceability, which is a significant property in the RFID security and privacy field. Unfortunately, the Scyther tool does not analyse observational equivalence [113].

(e) It does not directly support equational theories. These would allow the direct modelling of Diffie-Hellman exponentiation, for example. Nevertheless, an underapproximation can be made. The idea is to provide the attacker with an additional capability. For instance, for Diffie-Hellman
exponentiation, when the attacker knows one of $g^{ab}$, $g^{ba}$, he is assumed to learn the other. Unfortunately, this underapproximation does not work appropriately with all security properties \cite{8}. Therefore, in this work it is assumed Scyther does not currently support Diffie-Hellman exponentiation. Furthermore, exclusive-or is not supported either \cite{96}, by the same token. In \cite{97}, the author of the tool mentions that support for both of these algebraic operators is considered as future work.

(f) \textit{It provides support for multi-protocol verification}: Some protocols which are correct in isolation can be attacked when they are executed together on the same network. The adversary can attack one of the protocols by using messages from the others \cite{194}.

(g) \textit{It provides support for adversarial models other than the Dolev-Yao}: For instance, the adversary’s ability to learn session keys can be modelled, as well as the adversary’s ability to learn the long-term keys of a given agent \cite{108}. In Scyther, though, authenticated channels are not directly supported. Nevertheless, they can be approximated \cite{109}.

The tool can be freely downloaded from its webpage at \url{http://www.cs.ox.ac.uk/people/cas.cremers/scyther/install-generic.html}. It is offered for Windows, MAC OS X, and Linux users.

\textbf{The graphical user interface of the tool} is described below. See also figure E.5 on page 153, where the Scyther main window and result window are shown, and figure E.6 on page 154, where the attack window can be found:

(a) In figure E.5, on the left, the Scyther main window is found. The user of the tool can either edit its own protocol description or load a pre-existing one. In the figure, the Needham-Schroeder protocol \cite{39} is shown. The input language to Scyther is the \textit{spdl} language, short for Security Protocol Description Language, based on the operational semantics in \cite{195}. The language allows the modelling of a security protocol, which includes the security claims, i.e. security properties, to be made.

In \textit{spdl}, protocols are described as a set of roles. In figure E.5, \textit{roles} I and R are defined, corresponding to the initiator and the responder roles, respectively. In turn, roles feature a number of \textit{events}, which most importantly comprise send and receive events and claim events. Claim events specify the security properties to be verified. For instance, in the figure, the claim \textit{Secret} can be observed. It includes a term whose secrecy is intended to be verified. Finally, it can be observed that values that are local to each role are also declared. In the figure, these are $ni$ and $nr$, for both the initiator and the receiver roles.

(b) Once the protocol has been modelled in the main window, the user can select from three possible actions:
E. TOOLS FOR THE AUTOMATED FORMAL VERIFICATION OF SECURITY PROTOCOLS

i. **Verification of claims:** The verification or falsification of the claimed security properties can be requested. The result window will pop up, see figure E.5, on the right. It shows the results of the verification. It is noteworthy that if a claim is incorrect, an attack on the protocol exists, at a minimum \[8\]. If the button situated next to the claim is clicked, the attack window pops up, see figure E.6 on page 154.

ii. **Automatic generation of claims:** In case a protocol model contains no claims, this option can be chosen so that Scyther automatically adds them to the model. Then, analysis proceeds as in the previous point above.

iii. **Protocol role characterisation:** The tool offers the possibility to generate a representation of all possible behaviours. These execution patterns can then be examined so that potential problems with the protocol can be identified.

In this subsection, the Scyther Tool \([43]\) has been reviewed. In the following subsection the Casper/FDR Tool \([44, 45, 46]\) is addressed.

### E.3.4 Casper/FDR

In this subsection, Casper/FDR \([44, 45, 46]\) is reviewed.

Firstly, it has been considered important to elaborate on: the FDR tool \([45]\), the CSP process algebra \([196]\) and the Casper compiler \([44]\). The relationship between them is further illustrated in figure E.7 on page 155, where the corresponding workflow can be appreciated:

(a) **FDR, short for Failures Divergence Refinement:** FDR can be used as a model checker for the analysis of security protocols. Nevertheless, that is just one of the possible applications of the tool. Generally speaking, FDR receives as input a specification and an implementation, compares them, and outputs whether any execution of the implementation corresponds to the specification \([197]\).

It is important to note that FDR is based on enumeration and subsequent exploration of the state space resulting from the specified system, i.e. implementation. Consequently, only finite state systems can be considered. As a result of the search procedure, a counterexample falsifying the specification might be found, and then returned \([101]\).

With regards to the use of FDR as a model checker, protocols must first be modelled in CSP or any other semantically similar notation \([197]\). CSP is described below in the next point.

(b) **CSP, short for Communicating Sequential Processes:** CSP \([196]\) is a process algebra. It allows the specification and analysis of security protocols \([193]\), and, more generally, of systems formed from processes which interact via messages \([197]\). Each agent and the intruder are represented by CSP processes, which form the system. In addition, claimed security properties form the specifications. Both system and specifications are input into FDR, which checks whether the specifications are met by the system \([198]\).
Figure E.5: Scyther Main and Result Windows. Picture taken from [8].
Figure E.6: Scyther Attack Window. Picture taken from [8].
Modelling a system using CSP, though, requires considerable expertise, is time-consuming, and errors can go unnoticed. In order to address that issue, Casper, which is briefly introduced in the next point, and further examined below in this section, was developed.

(c) Casper: Casper [44] was developed to ease the task of preparing a CSP model. It is a compiler which receives a more abstract specification and outputs CSP code. That code can then be input into FDR.

![Figure E.7: Casper/FDR WorkFlow. Picture taken from [9].](image)

The input file to Casper consists of two parts. The reader is referred to the Casper User Manual and Tutorial [101] for further information. See also figure E.8 on page 157, where the Casper input file corresponding to the Needham Schroeder signature protocol is shown to illustrate the description below:

(a) Specification of the protocol: In the first part of the input file to Casper, the way the protocol works is specified generically. This part is in turn subdivided into several sections, whose headers are, see in the figure as well:

i. Free variables, where the types of variables and functions are declared.

ii. Protocol description, defining the messages exchanged.

iii. Processes. It has already been observed that each agent in the system is modelled as a CSP process. Under this header, names for the roles which the agents will play are declared, e.g. INITIATOR, RESPONDER and SERVER in the figure. These are also the names of the
corresponding CSP processes.

It is also important to note both agent name (first variable within the parentheses) and initial knowledge of the agents is specified under this heading as a template. In other words, as parameters to be instantiated later on in the second part of the input file, when the particular system to be verified is defined.

iv. *Specification.* In this section, the claimed security properties of the protocol are declared. Only secrecy and authentication properties are supported by Casper. In the figure, we observe the *agreement* and *aliveness* specifications. Both of them are forms of authentication requirements. For a complete list of Casper protocol specifications in general, and Casper authentication specifications in particular, the reader is referred to the Casper User Manual and Tutorial [101].

(b) *The particular system to be verified:* In this second part, the particular system to be checked is specified as an instantiation of the generic specification in the first part. Once more, this part is further subdivided into several sections, whose headers are:

i. *Actual variables:* Under this heading, the types declared in the aforementioned free variables section are instantiated.

ii. *Functions:* In a similar way, the functions are instantiated as well.

iii. *System:* This section declares the agents that compose the particular system, by instantiating the corresponding processes specified under the aforementioned heading *Processes.*

It is important to highlight that it can be specified that a given agent can take more than one role, just one role, or even no role. In addition, it can also be modelled that the agent runs the protocol just once or more than once.

For instance, in the figure, if it was desired that the server Sam could run the protocol twice, `;SERVER(Sam)` would be added after `SERVER(Sam)`.

iv. *Intruder Information:* In this section, the intruder’s identity and initial knowledge are declared.

In order to install Casper, the reader is referred to its webpage at [http://www.cs.ox.ac.uk/gavin.lowe/Security/Casper/installation.html](http://www.cs.ox.ac.uk/gavin.lowe/Security/Casper/installation.html).

Finally, Casper/FDR has a set of other features which have not been covered above in this section, but are relevant to the RFID environment. The justification for that relevance will be shown in section 6.4, where a number of suggestions are made for the improvement of the tools to better meet the requirements of the RFID environment:

(a) *Consideration of XOR for Vernam encryption:* Casper takes into account bit-wise exclusive-or for Vernam encryption [100, 101, 102]. The reader is reminded that the Vernam encryption of $m$ by $m'$ equals $m \oplus m'$.
E.3 Overview of Surveyed Tools

Figure E.8: Example Casper Input File. Needham Schroeder Signature Protocol. Picture taken from [10].

```plaintext
-- Needham Schroeder Signature Protocol
#Free variables
a, b : Agent
s : Server
m : Text
SKey : Agent -> ServerKey
f : RandFunction
InverseKeys = (SKey, SKey)

#Processes
INITIATOR(a, s, m) knows SKey(a)
RESPONDOR(b, s) knows SKey(b)
SERVER(s) knows SKey

#Protocol description
0. a -> b : m
1. a -> s : b, (f(m) = cs)(SKey(a))
2. a -> m : (s, cs = f(m))(SKey(b))

#Specification
Agreement(a, b, [m])
AisAlways(a, b)

#Actual variables
Alice, Bob, Mallory : Agent
Sam : Server
m : Text

#Functions
symbolic SKey

#system
INITIATOR(Alice, Sam, M)
RESPONDOR(Bob, Sam)
SERVER(Sam)

#Intruder Information
Intruder = Mallory
IntruderKnowledge = (Alice, Bob, Mallory, Sam, SKey(Sam))
```

Figure E.8: Example Casper Input File. Needham Schroeder Signature Protocol. Picture taken from [10].
(b) To the best of our knowledge, support for exponentiation is not currently available [96, 101]: In [198], the authors extend Casper so that the user can define types that allow the use of, for example, an exponentiation constructor \( \text{Exp}(f, x) \). Then, properties such as the commutativity of exponentiation can be modelled by defining them as algebraic properties under the separate \textit{equivalences} heading in the Casper input file. The \textit{equivalences} heading is currently available in Casper [101, 198]. As a result, in [198], the authors are able to analyse the Diffie-Hellman key exchange protocol. Unfortunately, to the best of our knowledge, user-defined types as defined in the extension to Casper in [198] are not included in the current Casper distribution [101].

(c) Support for flow control is limited: As noted above in this section when describing the Casper input file, the description of the protocol is provided under the heading \textit{protocol description}. It consists of a sequence of messages. The only support for flow control can be found in the form of a test-line. The test-line is inserted immediately after the line corresponding to the exchanged message it is related to. The tests described are conducted by the agent receiving that message. In case the tests are unsuccessful, the execution is aborted by the agent. For instance, it could be tested whether a timestamp is recent [101].

(d) Support for adversary models beyond the Dolev-Yao: Casper makes it possible for the user to model adversaries other than the Dolev-Yao. The mechanisms provided include:

i. \textit{Key compromise}: It allows the user to specify that all keys of a given type will be compromised. For example, for a protocol using timestamps, it can be modelled that the session key will be passed to the intruder after a particular number of time units [101].

ii. \textit{Channels other than the Dolev-Yao}: It is also possible to model channels other than the Dolev-Yao. This is declared under the \textit{Channels} section. For each exchanged message, it is possible to specify the properties of the channel. For instance, \textit{confidentiality}, where the intruder cannot learn the contents of the message, and/or \textit{no faking}, where the intruder cannot fake messages. The interested reader is referred to the Casper User Manual and Tutorial [101] for a complete coverage of the possible channel properties and further information.

In this subsection, the Casper/FDR Tool [44, 45, 46] has been reviewed.