UNIVERSITÉ DE MONTRÉAL

CLASSIFICATION, FORMALIZATION AND AUTOMATIC VERIFICATION OF UNTRACEABILITY IN RFID PROTOCOLS

ALI KHADEM MOHTARAM
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CLASSIFICATION, FORMALIZATION AND AUTOMATIC VERIFICATION OF UNTRACEABILITY IN RFID PROTOCOLS

présenté par : KHADEM MOHTARAM Ali
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

To Hoda.

ACKNOWLEDGEMENTS

First, and foremost, I would like to express my thanks and gratitude to my supervisor John Mullins for welcoming me to his research laboratory. John has dedicated a considerable amount of time and energy to the supervision of my research and his guidance, patience and support throughout this research and the writing of this dissertation was invaluable. I look forward to our future collaboration and continued friendship.

My thanks also aimed at my co-supervisor, Alejandro Quintero, president of the Jury, Guiliano Antoniol, as well as jury member, Bram Adams, for accepting evaluation of my dissertation.

My father always valued education and taught me hard work and perseverance. I would like to extend my thanks to my family ; My parents, Dara and Parivash, and my sister, Mojan, for their love and moral support in everything I have chosen to do ; Above all, my wife, Hoda, who showed me unquestioning faith in my abilities, unfailing support and unconditional love. They have been very supportive and concerned all along. I really missed having them by my side.

Finally I would like to thank my cousin Nima, for a great friendship and numerous hours spent together in Montreal, and also all the people, who their names are not mentioned and helped me with various aspects of conducting research and the writing of this dissertation.

RÉSUMÉ

Les protocoles sécurité RFID sont des sous-ensembles des protocoles cryptographiques mais avec des fonctions cryptographiques légères. Leur objectif principal est l'identification à l'égard de certaines propriétés de intimité comme la non-traçabilité et la confidentialité de l'avant. La intimité est un point essentielle de la société d'aujourd'hui. Un protocole d'identification RFID devrait non seulement permettre à un lecteur légitime d'authentifier un tag, mais il faut aussi protéger la intimité du tag. Des failles de sécurité ont été découvertes dans la plupart de ces protocoles, en dépit de la quantité considérable de temps et d'efforts requis pour la conception et la mise en œuvre de protocoles cryptographiques. La responsabilité de la vérification adéquate devient cruciale.

Les méthodes formelles peuvent jouer un rôle essentiel dans le développement de protocoles de sécurité fiables. Les systèmes critiques qui nécessitent une haute fiabilité tels que les protocoles de sécurité sont difficiles à évaluer en utilisant les tests conventionnels et les techniques de simulation. Cela a eu comme effet de concentrer les recherches sur les techniques de vérification formelle de tels systèmes pour assurer un degré élevé de fiabilité. Par conséquent, certaines recherches ont été faites dans ce domaine, mais une définition explicite de certaines de ces propriétés de sécurité n'ont pas encore été donnée.

L'objectif principal de cette thèse est de démontrer l'utilisation de méthodes formelles pour analyser les propriétés de intimité du protocole RFID. Plusieurs définitions sont données dans la littérature pour les propriétés non-traçabilité, mais il n'y a pas d'accord sur sa définition exacte. Nous avons introduit trois niveaux différents pour cette propriété en ce qui concerne les expériences de intimité existantes. Nous avons également classé toutes les définitions existantes avec différents points forts de la propriété non-traçabilité dans la littérature. De plus, notre approche utilise spécifiquement les techniques de calculs de processus pi calcul appliqués pour créer un modèle pour un protocole. Nous démontrons les définitions formelles de nos niveaux de non-traçabilité proposées et l'applique à des études de cas sur les protocoles existants.

ABSTRACT

Radio Frequency IDentification (RFID) is the wireless non-contact use of radio-frequency electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. Since RFID tags can be attached to clothing, possessions, or even implanted within people, the possibility of reading personally-linked information without consent has raised privacy concerns. Privacy is the essential part of today's society.

RFID protocols are subsets of cryptographic protocols but with lightweight cryptographic functions. Their main objective is identification with respect to some privacy properties, like untraceability. An RFID identification protocol should not only allow a legitimate reader to authenticate a tag but also it should protect the privacy of the tag. Although design and implementation of cryptographic protocols are tedious and time consuming, security flaws have been discovered in most of these protocols. Therefore the responsibility for reliable and proper verification becomes crucial.

Formal methods can play an essential role in the development of reliable security protocols. Critical systems which requires high reliability such as security protocols are difficult to be evaluated using conventional tests and simulation techniques. This has encouraged the researchers to focus on the formal verification techniques to ensure a high degree of reliability in such systems. In spite of the studies which have been carried out in this field, an explicit definition for some of these security properties is still missing.

The main goal of this work is to demonstrate the use of formal methods to analyse RFID protocol's untraceability. Untraceability generally defined as ensuring that an attacker can not get any information to be used to trace an object in time and space. Several definitions are discussed in the literature for untraceability property, however, there is no compromise on its exact definition. We have introduced three different levels for this property. We also have classified all former definitions of untraceability property in the literature. In order to create a model for a protocol and define privacy properties, our approach employs process calculi technique, applied pi calculus. We have demonstrated the formal definitions of suggested untraceability levels, extending them to some case studies on the existing protocols.

Keywords: RFID security protocol, privacy, untraceability, formal method, process algebra, applied pi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

BAN	Burrows–Abadi–Needham
FSM	Finite State Machine
IEC	International Electrotechnical Commission
IFF	Identify Friend or Foe
ISO	International Organization for Standardization
kbps	Kilobit per second
OBDD	Ordered Binary Decision Diagram
OSK	Ohkubo, Suzuki and Kinoshita
PCL	Protocol Composition Logic
PPTM	Probabilistic Polynomially-bounded Turing Machine
PRF	Pseudonym Random Function
PRNG	Pseudo Random Number Generator
RF	Radio Frequency
RFID	Radio Frequency IDentification
SATMC	SAT-based Model-Checker
YATRAP	Yet Another Trivial RFID Authentication Protocol

CHAPTER 1

INTRODUCTION

As we advance through the 21^{st} century, it has been demonstrated that, RFID technology will continue to grow and expand beyond our dreams. RFID systems are expected to become more common and useful tools in most of the remote object identification systems. Because of their low production costs and tiny size, they are currently deployed in many fields. Sensitive informations, such as credit card numbers, passport informations, social security, medical information, pass everyday through the insecure wireless channel. Unfortunately, there is some concerns within the use of RFID in view of the minimal privacy and security held. Privacy and security have been defined in (Marcella et Stucki, 2003) as, "when and with whom you share your personal information," and "how well information is protected from unauthorized access, alteration, or destruction."

The security and privacy objectives of so many systems have not been met. The computational limitations of RFID tags impose significant restrictions on the number and type of cryptographic primitives that can be implemented on them. Nonetheless, widespread deployment of these tags and also the contactlessness of communication bring up new threats to the user privacy. As an example, a tag can be embedded in a credit card or a passport which is linked to the person's identity,. In this case, the tracking of a tag turns into the tracking of a person. Chotia and Smirnov studied on privacy attacks on e-passports (Chothia et Smirnov, 2010), illustrating that, it is possible to trace the displacement of particular passport, without breaking the cryptographic functions.

This thesis will help the private design of RFID protocols and facilitate the analysis of existing schemes through the particular focus on untraceability property. Untraceability is one of the most important privacy properties which must be respected by an RFID system. We would consider the protocols private, if privacy requirement could not be violated by any adversary. Although, these properties are mostly used in the context of RFID systems, they are issues for any protocol, which is applied by a mobile device.

The remainder of this chapter is structured as follows: RFID systems will be introduced (Section 1.1). A brief description of cryptographic protocols will be revealed (Section 1.2) and their security properties will be discussed (Section 1.3). Formal approaches for protocol verification techniques will be investigated in (Section 1.4). The Problem statement will be notified (Section 1.5). The objectives of this thesis and the description of the contribution will be presented (Section 1.6). Finally, an overview of the structure will be summarised (Section 1.7).

1.1 RFID Systems

RFID is a system that transmits the identity (in the form of a unique serial number) of an object wirelessly, using radio waves. It's grouped under the broad category of automatic identification technologies. Auto-ID technologies include bar codes, optical character readers and some biometric technologies, such as retinal scans. The auto-ID technologies have been used to reduce the amount of time and labour needed to input data manually, like bar code systems, and to improve data accuracy. Some auto-ID technologies, such as bar code systems, often require a person to manually scan a label or tag to capture the data. RFID is designed to enable readers to capture data on tags and transmit it to a computer system without needing a person to be involved.

The beginnings of RFID technology can be traced back to work carried out during World War II on radar technology. In particular, an ancestor of modern RFID is the so-called Identify Friend or Foe (IFF) system, first introduced in second world war and still in use today to identify friendly or enemy aircraft. IFF, as used by the British Air Force at that time, was what we would describe today as a programmable tag that upon interrogation would return a code that identified the aircraft that carried it as friendly. Early IFF systems were further developed in the 1950s and nowadays are in common use in civil as well as military aviation.

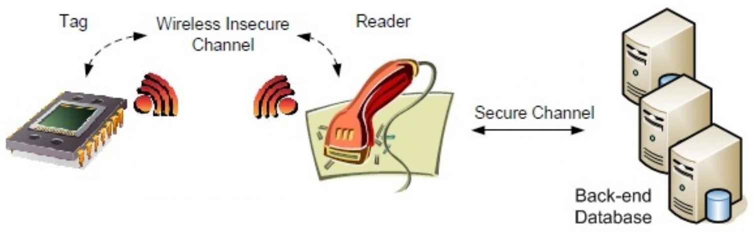


Figure 1.1 Overview of an RFID system

An RFID system, as shown in Figure 1.1 is composed of three main components, a tag (transponder), a reader (transceiver) and the reader's back-end database. A tag typically has a microchip to store basic information (ID, manufacturer's info) and an antenna for transmitting signals to RFID reader. A reader can interrogate a tag by sending a signal via electro-magnetic fields. The tag receives the signal through its antenna and responds with information stored on its microchip, which is verified by the reader against its back-end database. Although RFID systems are used as one of the most pervasive computing technologies, there are still so many security issues that need to be solved before their vast deployment. This technology poses critical privacy and security concerns (Chothia et Smirnov, 2010). In an RFID system, it is mostly supposed that the communication channel between a reader and its back-end database is via secured wired channel while the channel between a reader and a tag is wireless and insecure. Therefore studying the communication between the database and the readers is not relevant. Hence, readers and database are often considered as a single and unique entity in the security analysis.

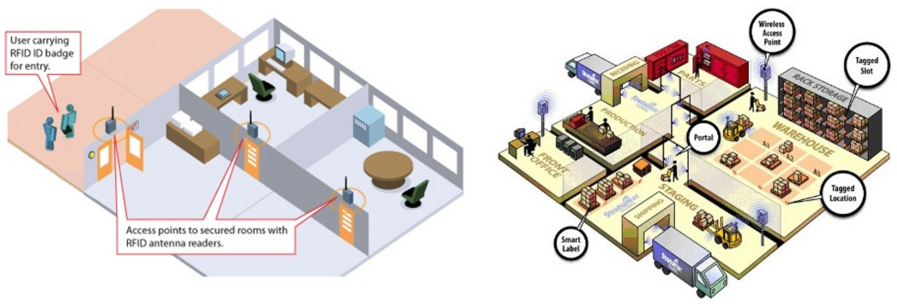


Figure 1.2 Two samples of RFID Applications: Access Control & Supply chain

The use of RFID in society attracts lots of attention in the past few years because of its beneficial features. The most well known application of the RFID tag is the supply chain and logistics. RFID tags can be attached to the objects in a supply chain to track, secure and manage goods throughout the entire production cycle. The overall cost of the supply chain could be reduced by using low cost tags and also adopting high-efficiency bulk-processing and non-human-intervention method. (Sarac *et al.*, 2010; Khashkhashi Moghaddam *et al.*, 2012). RFID chips can also be embedded into passports to record the holder's biometric information such as fingerprint and iris data (Kumar et Srinivasan, 2012). Some 95 countries, including the Canada and the United States have been using e-passports for several years (Passport Canada). RFID tags also can be a key and used in access control systems. They can store personal information for security check-ins. For example, an employee carries an ID card, embedded with a RFID chip, could authenticate his or her identity at the security entry in a facility very fast (Xiang, 2012). RFID system also has been used in a other industries like automatic payment systems, medical systems, animal tracking, libraries and smart appliances (Fosso, 2012; Wamba, 2012; Kushal *et al.*, 2012).

Different kind of tags are used in different places. Tags can be divided into three main types. Passive tags, which do not require batteries, can be much smaller, and have an unlimited life span. They are currently being developed by several companies globally. These tags are much less expensive than other types. Passive tags receive all power from the reader and necessarily cannot initiate any communications. Semi-passive tags are very similar to passive tags except for the addition of a small battery. This battery allows the tag to be constantly powered. Therefore, semi-passive tags are faster in response than passive, though less reliable and powerful than active tags. Active tags have their own internal power source, which is used to power tag to generate the outgoing signal. They are typically much more reliable (e.g. fewer errors) than passive tags due to the ability for active tags to conduct a "session" with a reader. Due to their on-board power supply, also transmitting at higher power levels than passive tags, allows them to be more effective in "RF challenged" environments like water, heavy metal (shipping containers, vehicles), or at longer distances. Any active tags have practical ranges of hundreds of meters. Table 1.1 depicts the main characteristic of each group.

Table 1.1 Tag Classification

Property	Type		
	Passive	Semi-Passive	Active
Frequency(MHz)	860-960	868-915 and 2.4GHz	860-960 and 2.4GHz
Internal Power	No	Yes	Yes
Bit Rate(kbps)	246	16	20/40/250
Read Range(m)	10	30	100
Cost(cents)	10-150	500-2000	1250-2500

1.2 Cryptographic Protocols

A security protocol also called a cryptographic protocol is a communication protocol that performs security functions while applying cryptographic methods. Participants to a protocol, called agents, have to follow some steps, known in advance, to communicate with each other.

Cryptographic techniques are used for the protocols with a security objective. The security objective might be confidentiality or might involve authentication, generation of random sequences, or partial sharing of a secret. Since the communication medium in RFID systems is public, the messages in the wireless channels can be eavesdropped by any adversary with the receiver equipment. An adversary may interface and even intercept, insert, block, modify the messages in the wireless channel. This has made the design of secure protocols difficult.

Using cryptographic protocols is one of the solutions to overcome security threats in RFID systems. Because of the resource limitation of RFID tags, it requires lots of efforts to find a light weight and secure protocol. Therefore simple and light weight cryptographic operations like logical operations(e.g. XOR, AND, OR) mostly used in these protocols. Many researchers have looked into the problem in order to design protocols which allow authorised persons to identify the tags without an adversary being able to trace them. Numerous papers addressing RFID cryptographic protocols have been published recently such as (Wang *et al.*, 2012; Bassil *et al.*, 2012; Qi *et al.*, 2012; Sun et Zhong, 2012; Lee, 2012; Jia et Wen, 2012).

Many works have used cryptographic primitives to encrypt the secrets like tag identifier. Hash function-based protocols like (Lee *et al.*, 2006; Henrici et Müller, 2004; Ohkubo *et al.*, 2003) are taking the advantage of one-way function to prevent direct exposure of secret. For example authors in (Ohkubo *et al.*, 2003) used two different hash functions, one for sending tag's internal state to the reader and the other for updating the tag's secret. In addition, Henrici in (Henrici et Müller, 2004) proposed a scheme that all the data management is done in back-end and the tag only requires a hash function. Therefore the communication channel does not need to be reliable and the reader need not be trusted and also no long-term secrets stored in tags. Later Dimitriou in (Dimitriou, 2006) presented an optimized scheme using hash functions, which do not require exhaustive search in the back-end database. Pseudonym Random Function (PRF) has also been used in the design of RFID protocols. For exapmle in (Tsudik, 2006), author presented an algorithm called YATRAP which needs only the secret ID and a Pseudo Random Number Generator (PRNG). Later Chatmon et al. (Chatmon *et al.*, 2006) presented YA-TRAP+ and OTRAP.

Most of these schemes are 3-round identification protocols as depicted in Figure 1.3. The first message is the starting request or may contain data such as a nonce. Nonce is an arbitrary number used only once in a cryptographic communication. The main part of the protocol lies on the information in the second message which changes at each new identification. It could be either the tag's identifier, or an encrypted version of its identifier. The third message is not used for the identification of the tag but it may be used by the tag to update its identifier.

Within this thesis we will represent protocols graphically using message sequence charts. Every message sequence chart shows a reader role R and a tag role T with the role names framed, on the top of the chart. Above the role names, the role's secret terms are shown.

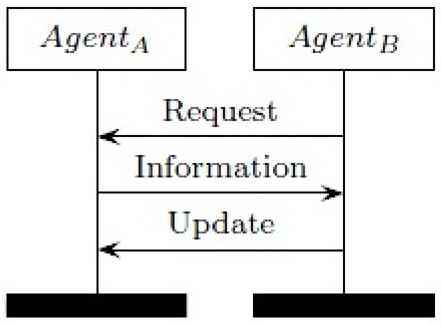


Figure 1.3 3-Round Identification Protocol

Events, such as nonce generation, computation, and assignments are shown in boxes. Messages to be sent and expected to be received are specified above arrows connecting the roles. It is assumed that an agent continues the execution of its run only if it receives a message conforming to the specification.

1.3 Security Properties

Cryptographic protocols are required to satisfy some security requirements also called security properties. Depending on the protocol's purpose, failure to preserve security properties may result in unveiling confidential information to an intruder and consequently, may lead to protocol failure. These includes classical properties like secrecy and authentication or privacy properties like anonymity or untraceability. These requirements must be overcome to consider a protocol secure against the threats.

Secrecy and authentication are the two basic standard requirements of any authentication protocol. To define secrecy informally, we could say that a protocol P preserves the secrecy of data M if P never publishes M, or anything that would permit the computation of M, even in interaction with an adversary. In addition to the secrecy, an authentication protocol should also enable an agent to prove her identity. Recalling the client-server handshake protocol, authentication ensures that the client C is only willing to share her secret with the server S; it follows that, if she completes the protocol, then she believes she has done so with S and hence authentication of S to C should hold. In contrast, server S is willing to run the protocol with the client C who he is willing to, and hence at the end of the protocol he only expects authentication of C to S to hold, if he believes C was indeed his interlocutor.

In addition to secrecy and authentication, there are other properties which could be conisdered as the main issue in RFID systems rather than other communication systems. An RFID authentication protocol should not only allow a legitimate reader to authenticate a tag but it should also protect the privacy of the tag. Here, we present the main privacy properties which are generally required for RFID protocols.

Anonymity allows an entity to make transactions that cannot be known by others. If the tag ID can be kept anonymous, the problem of leaking information pertaining to the user belongings would be solved. Anonymity is the topic of most researches. This property is highly important for tag owner privacy in the environment. It is informally defined by the ISO/IEC standard 15408 as ensuring that a user may use a service or resource without disclosing the user's identity.

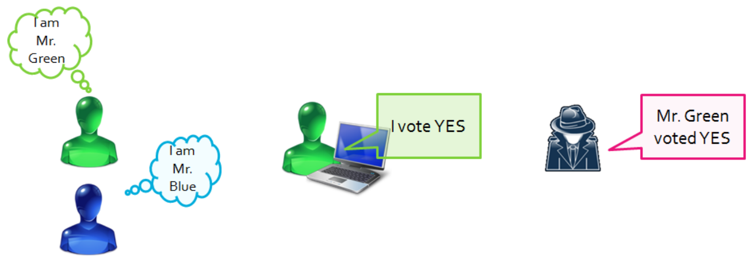


Figure 1.4 Anonymity

Intuitively, it says that an attacker cannot discern the tag from any information that the owner necessarily reveals during the identification. This might include nonces or keys which the tag owner is given during the protocol. Figure 1.4 shows the RFID tag anonymity.

Untraceability says that an attacker cannot trace the movement of a tag. RFID readers in strategic locations can record sightings of unique tag identifiers, which are then associated with personal identities. Passive tags respond to readers without alerting their owners when they powered by electromagnetic waves from a reader. The threat comes out when a tag identifier linked to personal information. For example, by using a credit card or an e-passport, if an attacker can establish a link between a person's identity and the tags he is carrying, at any such point the tracing of objects can become the tracing of a person.

Let's Consider set of messages sent by different agents, for example in a communication system, all messages sent by the same sender are related to each other for him but should not for an attacker. Untraceability as first defined by Avoine (Avoine et Oechslin, 2005) means that an attacker can not get any information to be used to trace an object in time and space. Figure 1.5 shows the concept of untraceability in a simple manner.

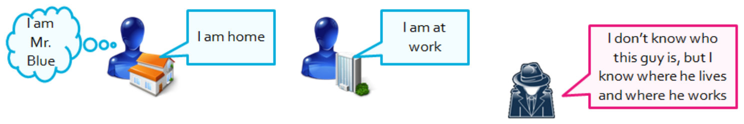

I am Mr. Blue
I am home
I am at work
I don't know who this guy is, but I know where he lives and where he works

Figure 1.5 Untraceability

Comparing with anonymity, in analysing untraceability the identity of the tag is not important, the point is that the link between different sessions of the tags cannot be established by an attacker. In this sense, untraceability is not intended to protect the link between a tag participated in the protocol and the identity of the user, but rather the link between different sessions of the tags.

Forward Secrecy ensures that even by compromising a tag at some later time, an adversary can not link the tag to its previous sessions. In other word she cannot trace the data back through past events in which the tag was involved. For example once the secret in the tag is stolen, all past activities can be traced by searching past logs. So the past activities should be protected from tampering. For forward privacy, the attacker is allowed to tamper with a tag and retrieve the data stored in it. We call a protocol forward secure if the attacker cannot use the obtained data to link the tag to past sessions. It requires that the adversary who only eavesdrops on the tag output, cannot associate the current output with past output.

In other words given set of observations between tags and readers and given the fact that all information stored in the involved tags has been revealed at time t, the adversary must not be able to find any relation between any observations of the same tag or a set of tags that occurred at time $\acute{t} < t$. It is depicted in Figure 1.6.

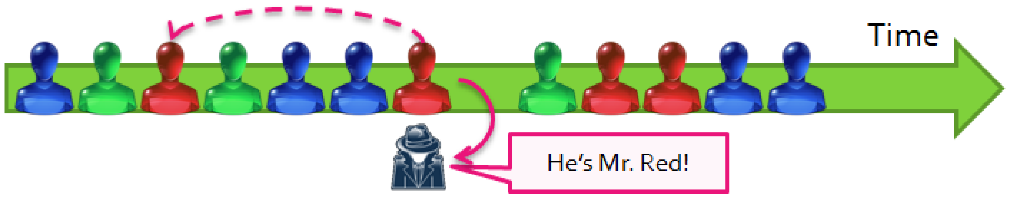


Figure 1.6 Forward Secrecy

1.4 Formal Verification of Security Protocols

Critical systems which require high reliability such as aircraft navigation systems, banking transactions, security protocols and other similar systems are difficult to be evaluated using conventional test and simulation techniques. Unlike testing that does not provide exhaustive coverage, verification using model checking gives a more thorough analysis of the system by checking satisfiability of the requirement through every branch of the model representing the system. It covers all possible scenarios of the behavior of the system. Formal verification techniques used in such systems to ensure a high degree of reliability. (Siminiceanu et Ciardo, 2012; Ahamad *et al.*, 2012; Souyris *et al.*, 2009)

Formal methods have proven to be a promising technique toward developing automated and generic protocol verification and testing methods. The main benefits of formal verification comparing to testing, is its soundness and exhaustiveness.

1.4.1 Formal Methods

Formal methods, as defined by Meadows (see Meadows, 2003), combine a language which can be used to model a cryptographic protocol and its security properties. It means the use of mathematical and logical techniques to express the system and its properties. By building a mathematically rigorous model of such a system and a mathematical definition of the properties, it is possible to verify the system's properties in a more thorough fashion than empirical testing. With these techniques, we can develop specifications and models, which describe all or part of a system's behavior at various levels of abstraction, and use them as input to an automated model checker. Formal methods in context of cryptographic protocols generally divided into two parts: Formal modelling of the protocol specification and Formal verification which refers to analysis of the security properties.

Although cryptographic protocols design is improving day by the day but there are still some security issues that are discovered mostly as the result of an informal approach. Formal methods successfully applied to find such problems. Many formal techniques and tools have been developed to reason about security and protocols. Probably the first mention of formal methods as a possible tool for cryptographic protocol analysis came in Needham and Schroeder (see Needham et Schroeder, 1978). However, the first work that was actually done in this area was done by Dolev and Yao (Dolev et Yao, 1983).

Modelling a cryptographic protocol begins with specification of the protocol and its requirements. Specification language could be a formal language or calculi. Milner's Pi-calculus(Milner, 1991) is a general-purpose calculus that has been successfully used to demonstrate security properties for some protocols. By extending it with specialized cryptographic primitives, Abadi obtained what he called the SPI calculus (Abadi et Gordon, 1997), a process algebra specifically designed for modelling cryptographic protocols (Abadi et Gordon, 1997), which allows verification of a broader spectrum of security protocols, as the encryption and decryption can now be checked. Since we used applied pi calculus as our modelling language, we will present the preliminaries of this language in chapter 2.

1.4.2 Verification Techniques for Cryptographic protocols

Generally formal verification techniques in context of cryptographic protocols divided into three main categories: *Logic oriented methods*, *Model checking* and *Theorem proving.*

Logic oriented methods

This method uses the modal logics to specify and analyse cryptographic protocols. Modal logic is a type of formal logic primarily developed in the 1960s that extends classical propositional and predicate logic to include operators expressing modality. They reason about the evolution of knowledge and beliefs within a system to show that certain conditions are satisfied. The basic idea of these approaches is the analysis of knowledge or belief during the protocol run. Inference rules are used to describe how beliefs or knowledge can be derived from other beliefs or knowledges.

The most famous in the class of logics is the Burrows–Abadi–Needham (BAN) logic (see Burrows *et al.*, 1989). It models messages and the beliefs of an agent. BAN logic is a set of rules for defining and analysing information exchange protocols. It is an example of a logic of belief, which consists of a set of modal operators describing the relationship of principals to data, a set of possible beliefs that can be held by principals (such as a belief that a message was sent by a certain other principal), and a set of inference rules for deriving new beliefs from old ones. Specifically, it helps its users determine whether exchanged information is trustworthy, secured against eavesdropping, or both. Note that belief logics such as BAN are generally weaker than state exploration tools since they operate at a much higher level of abstraction.

There are a numbers of others including Bieber's CKT5 (Bieber, 1990), Syverson's KPL (Syverson, 1990) and Protocol Composition Logic (PCL) (Datta *et al.*, 2005).

Model checking

Model checking (Clarke, 1997) is one of the most common formal verification techniques. Model refers to the mathematical representation of the system and using model checking technique a temporal logic formula besides the model of your system can determine whether the model satisfies the property or not. The basic idea is an automated method to determine the correctness property holds by exploring the state space of a model. If the property does not hold in each state, the model checking algorithm produces a counterexample and an execution trace leads to a state in which the property is violated. For models with a small state space this search can be exhaustive and the results are then equivalent to a formal proof. Since the state space might be too large to be examined directly, model checking is typically combined with abstraction techniques.

The inputs to the model checker are, a finite-state model M, a model that represents all possible scenarios of the behaviour of a system S, and a property P to be checked in every state of M. The model checker exhaustively explores all the paths through M while checking that P is true at each reachable state. If no branch in the model violates this property then the system is said to satisfy the property. Otherwise, the verifier outputs the branch that has violated the property, known as the counterexample.

Two general approaches to model checking are used: Explicit state enumeration and Symbolic model checking. In the first, the states of the system under verification are basically encoded and stored in a table and then checked against the desired properties. Many state reduction techniques have been developed to help reduce the state space without affecting the ability of the tool to discover insecure states. Other techniques are also used in order to optimize both the search and storage procedures. The second approach, symbolic model checking (Meadows et Meadows, 1995) brought some remedy to the state space explosion problem of model checking and is based on the use of Ordered Binary Decision Diagram (OBDD)s. Using OBDDs allows for an efficient representation of system state transition, thereby increasing the size of the system that could be verified. By using symbolic model checking, it is possible to verify extremely large reactive systems. This is possible because the number of nodes in the OBDDs that must be constructed no longer depends on the actual number of states or the size of the transition relation. Because of this breakthrough it is now possible to verify reactive systems with realistic complexity.

The most works on state machine approaches goes back to Dolev and Yao (Dolev et Yao, 1983). A Finite State Machine (FSM) includes a finite number of states and produces output on state transitions after receiving an input symbol. It is often used to model control portion of a protocol. In these models a state space is defined and then explored by the tool to determine if there are any paths through the space corresponding to a successful attack by the intruder. Much of this work was successful in finding flaws in protocols that had been previously undetected by human analysts. One of the first systems that used Dolev-Yao approach is Interrogator developed by Millen (Millen *et al.*, 1987). The tool attempts to locate protocol security flaws by an exhaustive search of the state space. In this class we can also mention NRL that is proposed by Meadows (Meadows, 1996). It is also based on the Dolev-Yao approach and use the similar strategy as Interrogator. Other researches has also focused on state exploration such as MurΦ (Mitchell *et al.*, 1997), Maude (Denker *et al.*, 1998), SATMC (SAT-based Model-Checker) (Armando *et al.*, 2003) or FDR (Lowe, 1996). The most significant problem of this kind of methods is that the search space of the paths to find the insecure state can be infinite which leads to a termination problem.

Theorem proving

This techniques model the computations performed in a protocol and define security properties as theorem, automated theorem checkers are then used to verify these theorems and thus prove properties of the model. Automated theorem proving is the oldest technique of formal verification, and it has been studied and practised since the 1960s. The logic is given by a formal system based on a set of axioms and inference rules. Theorem proving is the process of finding proofs using axioms of the system. The idea is to represent two models or properties as two formulas f and g in a reasonably expressive logic, consequently prove that $f \Rightarrow g$ (Pradhan et Harris, 2009).

Theorem provers are being used more and more in the mechanical verification of safety critical properties of hardware and software designs. Logic based proofs for security protocols build on traditional mathematical reasoning, using models extended for representing security concepts. Logical notations typically offer a wide range of data structures and operators which makes them capable of accurately representing the messages transmitted in security protocols. Unfortunately, a particular weakness of logic-based models is that they have no in-built notions of communication concepts such as message sequencing and these must be explicitly expressed in the model. Paulson in (Paulson, 1994) used the theorem prover *Isabelle* to analyze protocols.

In contrast to model checking, theorem proving can be applied to systems with *infinite* state spaces because it doesn't have the state space exploration problem. It also relies on techniques such as *structural induction* to prove over infinite domains. As a disadvantage to this technique, we can point that interactive theorem provers, require human interaction, so the theorem proving process is slow and sometimes error prone.

1.4.3 Adversary Model

In security scenarios typically we assume that cryptographic protocols should achieve their objectives in the presence of a malicious entity that seeks to break the security of the system, called adversary. The characterization of the adversary is essentially done by specifying the actions that she is allowed to perform (i.e. the oracles she can query), the goal of her attack (i.e. the game she plays) and the way in which she can interact with the system (i.e. the rules of the game).

Consider a public-key cryptosystem including a plaintext x underlying a challenge ciphertext y. The classic attacks in order of increasing strength are *chosen-plaintext attacks* (CPA), *non-adaptive chosen-ciphertext attacks* (CCA1), and *adaptive chosen-ciphertext attacks* (CCA2) (Naor et Yung, 1990). Under CPA, the adversary has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts. For example, in the public-key setting, giving the adversary the public key suffices to capture these attacks. Under CCA1, the adversary gets, in addition to the public key, access to an oracle for the decryption function. The adversary may use this decryption function only for the period of time preceding her being given the challenge ciphertext y. Under CCA2, the adversary again gets (in addition to the public key) access to an oracle for the decryption function, but this time she may use this decryption function even on ciphertexts chosen after obtaining the challenge ciphertext y, the only restriction being that the adversary may not ask for the decryption of y itself (Rackoff et Simon, 1992).

For RFID systems, no universal adversary model has been defined yet, up until now designs and attacks have been made in an ad hoc way. Even though there are concepts like CPA, CCA1 and CCA2 for confidentiality, in RFID systems the adversaries resources are defined in an ad hoc manner and vary depending on the publication. They can be passive, active but limited in the number of queries on the tag, active but cannot modify exchange between the legitimate tag and a reader or active but cannot tamper with the tags. For example Avoine defined the adversary as a Probabilistic Polynomially-bounded Turing Machine(PPTM) (Avoine, 2005), which interacts with RFID tags and readers through some oracles. Vaudenay (Vaudenay, 2007) also proposed a hierarchical model for adversary. His model captures eight classes of adversary capabilities ranging over four different types of tag corruption and two modes of observation. There are also other works in modelling the adversary in RFID systems like (Canard *et al.*, 2010; Cai *et al.*, 2012)

In this dissertation we consider the adversary model described by Dolev and Yao (Dolev et Yao, 1983). It is based on its initial knowledge which is extended by observing the communications. Since the adversary has complete control of the network in this model, it is assumed that the adversary has unlimited inference capabilities and messages may be read, modified, deleted or injected. It means that she has the ability to influence all communications between a tag and the reader, she can change the order of messages, combine the information in his knowledge to construct or interpret new terms and can therefore perform man-in-the-middle attacks on any tag that is within its range. In brief Dolev-Yao adversary is as follows:

- Eavesdrop all messages transmitted on the network
- Replay to captured messages
- Inject new messages deduced from captured messages
- Intercept messages

However, these capabilities are restricted due to the assumption of perfect cryptography. This means that the adversary cannot reverse hash functions and that she is not able to learn the contents of an encrypted term, unless she knows the decryption key.

1.5 Problem Statement

The contactlessness of communication and the expected ubiquity of RFID systems encourage nefarious entities to track and analyze tags in time and space. If at any such point a tag is linked to an individual, the tracking of a tag becomes the tracking of a person. The need for RFID protocols to be secured against these threats has been realized early on (Juels, 2006; Breu, 2011). A number of papers discussed about the untraceability problems, raised by RFID technologies and its importance, however, very few defined a formal model and the untraceability property precisely.

In the literature, there are several definitions of untraceability, while there is no agreement on its comprehensive definition. Generally two type of models have been considered in order to formalize untraceability; The Symbolic model and Computational model. Most of the definitions have been accomplished in computational settings. These experiments are poorly supported by automatic tools. In the symbolic world, in one hand Arapinis defined the notions of weak and strong untraceability in the process algebra (Arapinis *et al.*, 2009), on the other hand, Bruso defined a single definition for untraceability (Brusó *et al.*, 2010). By investigating the existing models we found that, there are some cases that make it impossible to satisfy the defined properties, or the definitions are not complied properly with some type of protocols. All the above models will be fully presented in the related works. Therefore, the need to sum up and classify all these definitions is essential.

These problems made us to think of studying the former definitions, providing more clear and explicit definitions for untraceability with different strength levels. Finally a solution is proposed for the automatic analysis of untraceaility of protocols.

1.6 Research Objectives

1.6.1 Research Question

- How to formalize and automatically verify untraceability of RFID protocols?

1.6.2 General Objective

The main idea of this thesis is to provide the design of private cryptographic protocols and facilitate the privacy evaluation of existing schemes, using formal verification techniques. This thesis presents an approach for modelling and verifying privacy property, particularly in RFID identification protocols. Our work is an attempt to the formal analysis of a more challenging privacy property in RFID protocols, so-called untraceability. Thus, by proving untraceability in an RFID protocol, it prevents a real world attacker to invalidate the protocol, making it impossible to trace a tag, and hence its owner as well.

1.6.3 Specific Objectives

- Establish different strength levels for untraceability properties
- Formalize untraceability properties using applied pi calculus
- Classify all existing definitions of untraceability in the literature
- Automatic Verification of untraceability in RFID protocols

1.6.4 Original Scientific Hypothesis of our Contribution(OSHC)

Hypothesis:

- Verifying untraceability in each proposed levels for any RFID protocol, prevents an adversary invalidate the untraceability of modelled protocol.

Justification of originality: Few researches have looked at formalizing untraceability in symbolic settings. Providing explicit definitions for different levels of untraceability in process algebra, considering the game based definitions in computational setting has not been provided in any research.

Refutability: The hypothesis will be refuted if an adversary can interfere the privacy of the protocol which satisfied our untraceability levels.

1.7 Outline of the Thesis

The rest of the dissertation is organized as follows.

- Chapter 2 introduces the syntax and semantics of the language of our symbolic setting: applied pi calculus.
- Chapter 3 surveys the related works in the application of formal methods to the analysis of untraceability in RFID protocols. Discussion about symbolic and computational settings will be presented in this chapter.
- Chapter 4 begins with classifying existing privacy games. Then we will propose our levels of untraceability related to the former games. After that we will presents our protocol model and formalization of proposed untraceability levels based on applied pi calculus. Finally we will compare all existing definitions of untraceability in the literature and provide a classification of this property.
- Chapter 5 includes some case studies which describe the results of verifying the RFID identification protocols from the implementation using proposed framework. Three existing RFID protocols will be modelled. Finally Verification results will be demonstrated.
- Chapter 6 summarizes the work presented and underlines our original contributions. We will also explain the limitations of the work and future improvements in this chapter. Finally, the Bibliography section lists all the articles, reports, books and etc.

CHAPTER 2

BACKGROUND: APPLIED PI CALCULUS

The applied pi calculus is a process calculi, a mathematical formalism for describing and analysing concurrent systems and their interactions. It is an extension of the pi calculus (Milner, 1999) with the addition of a rich term algebra, value-passing, primitive function symbols and equational theory to enable modelling of the cryptographic operations used by security protocols. It is specifically targeted at modelling security protocols by adding the possibility to model wide variety of complex cryptographic primitives, including, non-deterministic encryption, digital signatures, and proofs of knowledge, while the pi calculus has a fixed set of primitives built-in like symmetric and public key encryption. Symbolic model relies on Dolev-Yao model(Dolev et Yao, 1983) in which messages are modelled as algebraic terms instead of computational model. The applied pi calculus permits formal modelling of properties including: reachability, correspondence and observational equivalence. In the context of cryptographic protocols, these properties are particularly useful, since they allow the analysis of traditional security goals such as secrecy and authentication. Moreover, emerging properties such as privacy and traceability can also be considered. Here we briefly recall its basic notions, for an extended descriptions see (Abadi et Fournet, 2001).

2.1 Syntax and Informal Semantics

The calculus consists of terms including infinite set of names($a, b, c, \ldots$), variables(x, y, z) and a signature Σ consisting of a finite set of function symbols each with an associated arity(e.g. one-way hash functions, encryption, digital signatures and data structures as pairing), plain processes and extended processes.

Definition 2.1. (Algebraic Term) The set $T(\Sigma)$ of algebraic terms is the smallest set defined by the grammer of table 2.1

Table 2.1 Syntax for terms

L, M, N, T, U, V ::=	terms
$a, b, c, \ldots, k, \ldots, m, n, \ldots, s$	names
x, y, z	variables
$f(M_1, \ldots, M_{ar})$	function application

where f rages over the functions of Σ and ar matches the arity of f. A function symbol with arity 0 is called *constant*, with arity 1 is called *unary* and with arity 2 is called *binary*. Metavariables such as u, v, w are used to denote both names and variables. Tuples $u_1, \ldots, u_{ar}$ and $M_1, \ldots, M_{ar}$ are abbreviated to $\tilde{u}$ and $\tilde{M}$ respectively. We assume a type system for terms generated by a set of base types such as Integer, Key, or simply a universal type, Data. In addition if τ is a type, then $Channel(\tau)$ is a type too. Typically a,b, and c are used for channel names, s and k as names of some base types, and m and n as names of any type. We always assume that terms are well-typed and the substitutions preserves types.

Definition 2.2. (Ground Term) A term is called ground or closed if it contains no variables, and it is denoted by T_Σ.

Example 2.3. Let a be a constant, f be a unary function and g be a binary function. The ground terms that could be built by above signature are like followings:

$$a \quad f(a) \quad g(a,a) \quad g(f(a), f(a)) \quad f(g(a,a)) \quad \ldots$$

In the applied pi calculus, one has plain processes and extended processes. The grammar for processes is similar to the pi calculus, except that messages can contain terms (rather than just names) and that names need not be just channel names. The syntax of process is shown below:

Table 2.2 Syntax for processes

P, Q, R	::=	processes (or plain processes)
	0	null process
	$P \vert Q$	parallel composition
	$!P$	replication
	$\nu n.P$	name restriction
	if $M = N$ then P else Q	conditional
	$u(x).P$	message input(also written as $in(u, x).P$)
	$\bar{u}\langle M \rangle.P$	message output (also written as $out(u, x).P$)
	let $x = g(M_1, \ldots, M_n)$ in P else Q	destructor application
	let $x = M$ in P	binding

The null process 0 does nothing; $P|Q$ is the parallel composition of process P and Q to represent the agents of the protocol running in parallel; the replication $!P$ behaves as an infinite composition of P running in parallel $(P|P|\ldots)$ which mostly used to show the unbounded number of protocol sessions. The name restriction $\nu n.P$ makes a new, private name n inside P which is often used to represent nonces, keys, private channels and other fresh random numbers. The conditional construct if $M = N$ then P else Q is standard, but

$M = N$ represents equality rather than strict syntactic identity. If Q is a null process we can abbreviate it to if $M = N$ then P. Finally, the communication between agents is shown by message input and message output. The process $u(x).P$ is ready to input from channel u, then to run P with the received message replaced with parameter x, while $\bar{u}\langle M \rangle.P$ is ready to output M on channel u, then to run P. In both of these, the P is omitted when it is a null process. The set of free names in P, denoted $fn(P)$, consists of those names n occurring in P not in the scope of a restriction νx or input $u(x)$. In opposite, The set of bound names $bn(P)$ contains every name n which is not in free in P.

Further, the processes are extended with active substitutions:

Table 2.3 Syntax for extended processes

A, B, C	$::=$	extended processes
	P	plain process
	$A \vert B$	parallel composition
	$\nu n.A$	name restriction
	$\nu x.A$	variable restriction
	$\{M/x\}$	active substitution

Processes are extended with active substitution to take into account the knowledge exposed to the adversarial environment and reflect in its interaction with the protocol. The active substitution $\{M/x\}$ denotes the substitution that replaces the variable x with the term M in every process. In other words M is available to the environment but the variable x is just a way to refer to M. This allows access to terms which the environment cannot construct. Although the substitution $\{M/x\}$ concerns only one variable, large substitution can be written as: $\{M_1/x_1, \ldots, M_{ar}/x_{ar}\}$ or $\{\tilde{M}/\tilde{x}\}$.

Contexts may be used to represent the adversarial environment in which a process is run; that environment provides the data that the process inputs, and consumes the data that it outputs. A context $C[\_]$ is an expression (a process or extended process) with a hole. $C[P]$ is obtained as the result of filling $C[\_]$'s hole with P. An evaluation context $C[\_]$ is a context whose hole is not under a replication, a conditional, an input, or an output.

Active substitutions are useful because they allow us to map an extended process A to its frame $\varphi(A)$ by replacing every plain process in A with 0. A *frame*, denoted φ or ψ, is an extended process built from 0 and active substitution $\{M/x\}$; The frame $\varphi(A)$ represents the static knowledge exposed by a process A to its environment. The domain $dom(\varphi)$ of a frame φ is the set of the variables that φ exports.

Example 2.4. We are now able to model processes of the applied pi calculus. As an example consider the following processes:

$$P = c(x).\text{if } x = req \text{ then } \bar{c}\langle hello\rangle \text{ else } 0$$

This process waits for a message on the public channel c. If the received message is req, then the process sends *hello* on the public channel c.

2.2 Operational Semantics

Operational semantics of the applied pi calculus are defined by the means of two relations: *structural equivalence* and *internal reductions*.

2.2.1 Structural Equivalence

Informally, two processes are structurally equivalent if they model the same thing, but the grammar permits different encodings. For example, to describe a pair of processes A,B running in parallel, the grammar forces us to put one on the left and one on the right; that is, we have to write either $A|B$, or $B|A$. These two processes are said to be structurally equivalent.

Definition 2.5. (α-Conversion) is renaming a bound name or variable without changing the semantics.

Definition 2.6. (Structural equivalence $\equiv$) is the smallest equivalence relation on extended processes that is closed under α-conversion of both bound names and variables and application of evaluation contexts which satisfies the rules in table 2.4.

Table 2.4 Semantics for processes (1)

$A\|0$	$\equiv$	A	$\nu n.0$	$\equiv$	0
$A\|(B\|C)$	$\equiv$	$(A\|B)\|C$	$\nu u.\nu w.A$	$\equiv$	$\nu w.\nu u.A$
$A\|B$	$\equiv$	$B\|A$	$A\|\nu u.B$	$\equiv$	$\nu u.(A\|B)$
$!P$	$\equiv$	$P\|!P$			if $u \notin fn(A) \cup fv(A)$
$\nu x.\{M/x\}$	$\equiv$	0	$\{M/x\}$	$\equiv$	$\{N/x\}$
$\{M/x\}\|A$	$\equiv$	$\{M/x\}\|A\{M/x\}$			if $M =_E N$

The parallel composition, replication and restriction rules are self-explanatory. We briefly describe the substitution rules here. The rule $\nu x.\{M/x\} \equiv 0$ called *ALIAS*, enables us to introduce active substitutions with restricted scopes. The second rule, $\{M/x\}|A \equiv$

$\{M/x\}|A\{M/x\}$ called *SUBST*, describes the active substitution to a process. The final rule called *REWRITE*, allows terms that are equal modulo the equational theory to be swapped.

Example 2.7. Consider the following process:

$$P \equiv \nu s.\nu k.\, (\bar{c_1}\langle enc(s,k)\rangle | c_1(y).\bar{c_2}\langle dec(y,k)\rangle)$$

The first component publishes the message $enc(s,k)$ on channel c_1. The second receives a message from channel c_1, then using the secret key k decrypt it and sends the result on channel c_2. This process is structurally equivalent to the following extended process A:

$$A \equiv \nu s.\nu k.\nu x.\, (\bar{c_1}\langle x\rangle | c_1(y).\bar{c_2}\langle dec(y,k)\rangle | \{enc(s,k)/x\})$$

Using structural equivalence, every closed extended process A can be rewritten to consist of a substitution and a closed plain process with some restricted names: $A \equiv \nu.\tilde{n}.\{\tilde{M}/\tilde{x}\}|P$, where $fv(P) = \emptyset$, $fv(\tilde{M}) = \emptyset$ and $\{\tilde{n}\} \subseteq fn(\tilde{M})$.

2.2.2 Internal Reduction

Internal reduction ($\rightarrow$) is informally defined as the execution of a process with respect to control flow and communication. It is formally defined as follows:

Definition 2.8. (Internal reduction $\rightarrow$) is the smallest relation on extended processes that is closed under structural equivalence and application of evaluation satisfying the rules of table 2.5.

Table 2.5 Semantics for processes (2)

COMM	$\bar{c}\langle x\rangle.P|c(x).Q \rightarrow P|Q$
THEN	if $N = N$ then P else $Q \rightarrow P$
ELSE	if $L = M$ then P else $Q \rightarrow Q$

The reflexive and transitive closure of $\rightarrow$ is written as $\rightarrow^*$. To better understand the substitution and reduction, we provide some examples as follows:

Example 2.9. Consider the process P described in Example 2.7:

$$P \equiv \nu s.\nu k.(\bar{c_1}\langle enc(s,k)\rangle)|c_1(y).\bar{c_2}\langle dec(y,k)\rangle$$

We have:

$$\begin{array}{llll}
P & \equiv & \nu s.\nu k.\nu x_1.\,(\bar{c_1}\langle x_1\rangle)|c_1(y).\bar{c_2}\langle dec(y,k)\rangle|\{enc(s,k)/x_1\}) & \text{(by ALIAS)} \\
 & \xrightarrow{\nu x_1.\bar{c_1}\langle x_1\rangle} & \nu s.\nu k.(c_1(y).\bar{c_2}\langle dec(y,k)\rangle|\{enc(s,k)/x_1\}) & \text{(by COMM)} \\
 & \xrightarrow{c_1(x_1)} & \nu s.\nu k.(\bar{c_2}\langle dec(x_1,k)\rangle|\{enc(s,k)/x_1\}) & \\
 & \equiv & \nu s.\nu k.\nu x_2.\,(\bar{c_2}\langle x_2\rangle|\{enc(s,k)/x_1\}|\{dec(x_1,k)/x_2\}) & \text{(by SUBST)} \\
 & \xrightarrow{\nu x_2.\bar{c_1}\langle x_2\rangle} & \nu s.\nu k.(\{enc(s,k)/x_1\}|\{dec(x_1,k)/x_2\}) & \text{(by COMM)}
\end{array}$$

Example 2.10.

$$\begin{array}{rcll}
\bar{c}\langle M\rangle.P|c(x).Q & \equiv & \nu x.(\bar{c}\langle x\rangle.P|c(x).Q|\{M/x\}) & \text{(by ALIAS)} \\
 & \rightarrow & \nu x.(P|Q|\{M/x\}) & \text{(by COMM)} \\
 & \equiv & \nu x.(P|Q\{M/x\}|\{M/x\}) & \text{(by SUBST)} \\
 & \equiv & P|Q\{M/x\}|\nu x.\{M/x\} & \text{(by restriction rules)} \\
 & \equiv & P|Q\{M/x\} & \text{(by ALIAS)}
\end{array}$$

2.3 Equivalences

In addition to secrecy and correspondence properties, there are more complex properties which enables us to define privacy. Intuitively, two processes are said to be equivalent if an observer has no way to tell them apart. In order to define observational equivalence, we could say that processes P and Q are equivalent if they can output on the same channels, no matter what the context they are placed inside.

2.3.1 Observational Equivalence

We write $P \Downarrow_c$ when P emits a message on the channel c, that is, when $P \rightarrow^* C[out(c, M); Q]$ for some evaluation context C that does not bind c and some process Q.

Definition 2.11. (Observational Equivalence $\approx$) is the largest symmetric relation R on processes such that PRQ implies:

1. if $P \Downarrow_c$ then $Q \Downarrow_c$;
2. if $P \rightarrow^* P'$ then there exists Q' such that $Q \rightarrow^* Q'$ and $P'RQ'$;
3. $C[P] \quad R \quad C[Q]$ for all evaluation contexts $C[\_]$.

$\Downarrow_c$ is called a *barb* on c in the pi calculus, and $\approx$ is one of the two usual notions of barbed bisimulation congruence. Intuitively, a context may represent an attacker, and two processes are observationally equivalent if they cannot be distinguished by any attacker.

Two terms M and N are said to be *equal* in the frame φ, and written as $(M = N)_\varphi$, if and only if $\varphi \equiv \nu\tilde{n}.\sigma$, $M\sigma = N\sigma$, and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ for some names $\tilde{n}$ and substitution σ.

2.3.2 Labelled Bisimilarity

The quantification over contexts makes the definition of observational equivalence hard to use in practice. Therefore, labelled bisimilarity is introduced, which is more suitable for both manual and automatic reasoning. Labelled bisimilarity relies on an equivalence relation between frames, called static equivalence, which we define first. Two substitutions may be seen as equivalent when they behave equivalently when applied to terms. This is shown by $\approx_S$ and called *static equivalence.*

Definition 2.12. (Static equivalence $\approx_S$). Two closed frames φ and ψ are statically equivalent, and is written $\varphi \approx_S \psi$, when:

1. $dom(\varphi) = dom(\psi)$
2. For all terms M and N we have that $(M = N)_\varphi$ if and only if $(M = N)_\psi$

We say that two closed extended process are statically equivalent, and denoted by $A \approx_S B$, when their frames are statically equivalent $\varphi(A) \approx_S \varphi(B)$.

This is called static because by using frames we are able to examine only the current state of the processes and not their dynamic behaviour. Static equivalence captures the static part of observational equivalence, more precisely they coincide on frames(Abadi et Fournet, 2001).

Example 2.13. Let's assume the simple signature $\Sigma = \{hash, pair, fst, snd\}$ and the equational theory satisfying $fst(pair(x, y) = x$ and $snd(pair(x, y) = y$ over variables x and y, where $hash$, fst and snd are unary functions and $pair$ is a binary function. We could have the following statements:

- $\nu m.\{m/x\} \approx_S \nu n.\{n/x\}$; Since they are structurally equivalent.
- $\nu m.\{m/x\} \approx_S \nu n.\{hash(n)/x\}$
- $\{m/x\} \not\approx_S \{hash(n)/x\}$; Since the first one satisfies $x = m$ but the second one does not.
- $\nu s.\{pair(s, s)/x\} \not\approx_S \nu s.\{s/x\}$; Since the first one satisfies $pair(fst(x), snd(x)) = x$ but the second one does not.

The operational semantics is extended by a labelled operational semantics enabling us to reason about processes that interact with their environment. Labelled operational semantics depicted in table 2.6 defines the relation $\xrightarrow{\alpha}$ where α is a label of the form $c(M)$, $\bar{c}\langle u\rangle$ or $\nu u.\bar{c}\langle u\rangle$ such that u is either a channel name or a variable of base type. It enables us to capture the dynamic part of observational equivalence.

The transition $A \xrightarrow{c(M)} B$ means that the process A performs an input of the term M from the environment on the channel c, and the resulting process is B. If the item is a free variable x or a free channel d, then the label $\bar{c}\langle x\rangle$, respectively $\bar{c}\langle d\rangle$, is used. If the item being

Table 2.6 Semantics for processes (3)

IN	$c(x).P \xrightarrow{c(M)} P\{M/x\}$	
OUT-ATOM	$\bar{c}\langle u\rangle.P \xrightarrow{\bar{c}\langle u\rangle} P$	
OPEN-ATOM	$\dfrac{A \xrightarrow{\nu u.\bar{c}\langle u\rangle} A' \quad u \neq c}{\nu u.A \xrightarrow{\nu u.\bar{c}\langle u\rangle} A'}$	
SCOPE	$\dfrac{A \xrightarrow{\alpha} A'}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$	u does not occur in α
PAR	$\dfrac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A\|B \to \alpha A'\|B}$	
STRUCT	$\dfrac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$	

output is a restricted channel d, then the label $\nu d.\bar{c}\langle d\rangle$ is used. Finally, if the item is a term M, then the label $\nu x.\bar{c}\langle x\rangle$ is used, after replacing the occurrence of the term M by x and wrapping the process in $\nu x.(\{M/x\}|\_)$.

Definition 2.14. (Labelled bisimilarity $\approx_L$). Labelled bisimilarity is the largest symmetric relation R on closed extended processes, such that ARB implies:

1. $A \approx_S B$
2. if $A \to A'$, then $B \to^* B'$ and $A'RB'$ for some B'
3. if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A'RB'$ for some B'.

Clauses 2 and 3 of this definition correspond to classical notions of bisimilarity (Milner, 1999). Clause 1 asserts static equivalence at each step of the bisimulation.

Example 2.15. Consider a handshake protocol between a client C and server S as illustrated below.

This protocol is started with the request from a client C, then the server S generates a fresh session key k, and encrypts it using the client's public key pk_C. When C receives this message he decrypts it using his private key and extracts the session key k. Finally C uses this key to symmetrically encrypt the secret s and sends the encrypted message to S. Note that the above notation is an incomplete description and does not cover all specific parts of the protocol such as the creation of nonces or the various checks, made by participants.

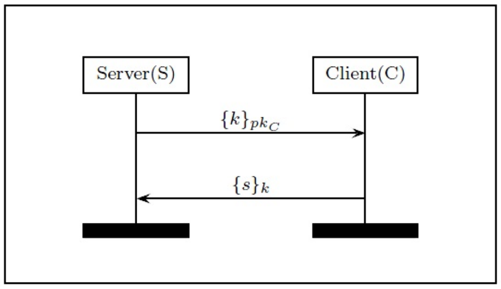


Figure 2.1 A Sample Protocol

This protocol is defined with respect to the signature $\Sigma = \{adec, aenc, sdec, senc\}$, where all are binary functions representing the asymmetric and symmetric key encryption. We define following equation over the variables x and y, in order to model the behaviour of encryptions:

$$adec(x, aenc(pk(x), y)) = y$$
$$sdec(x, senc(x, y)) = y$$

The protocol can now be modelled in pi calculus as the process P, defined as follows:

$$\begin{aligned} P &= \nu\ sk_S.\ \nu\ sk_C\ \nu\ \text{s}. \\ &\quad \text{let } pk_S = pk(sk_S) \text{ in let } pk_C = pk(sk_C) \text{ in} \\ &\quad (\bar{c}\langle pk_S\rangle | \bar{c}\langle pk_C\rangle | !S | !C) \\ \\ S &= c(pk).\nu k.\bar{c}\langle aenc(pk, k)\rangle. \\ &\quad c(x).\text{ if } sdec(k, x) = tag \text{ then } Q \\ \\ C &= c(x).\text{ let } k = adec(sk_C, x). \\ &\quad \bar{c}\langle senc(k, s)\rangle \end{aligned}$$

In addition to the above used symmetric and asymmetric encryption, the theory allows us to model different cryptographic primitives such as: pairing, digital signature schemes and etc, which are briefly presented below.

Ordered Pairing Algebraic data structures such as ordered pairs, tuples, arrays, and lists appear in several examples. It is not difficult to encode them in the Π-calculus. For example, the signature Σ may contain the binary function symbol pair and an element can be extracted using the unary functions *fst* and *snd*. $pair(x, y)$ may abbreviated to (x, y) and the *fst* and *snd* equations are as follows:

$$fst(pair(x, y)) = x$$
$$snd(pair(x, y)) = y$$

One-way Hash function A one-way hash function is represented as a unary function symbol h with no equation. The absence of any equational theory associated with it ensures the one-wayness of h and its collision resistance by the fact that $h(x) = h(y)$ only when $x = y$.

Asymmetric encryption An alternative and equivalent formalism for asymmetric encryption considers two unary constructors *pk* and *sk* for generating public and secret keys from an agent, to capture the notion of constructing a key pair from an agent:

$$adec(aenc(m, pk(ag)), sk(ag)) = m$$

Digital signature In a similar manner to asymmetric encryption, digital signatures rely on a pair of signing keys. In each pair, the secret key serves for computing signatures and the public key for verifying those signatures. In order to model digital signatures and their checking, in addition to the unary function symbol *pk* from asymmetric encryption, the new binary function symbol *sign* for constructing signatures, the unary function symbol *getmass* that allows the adversary to get the message m from the signature even without having the key and the binary function checksign to check the signature, and returning m only when the signature is correct.

$$getmess(sign(m, k)) = m$$
$$checksign(sign(m, k), pk(k)) = m$$

It is also possible to model signatures that do not reveal the message m, in the Handbook of Applied Cryptography four different classes of digital signature schemes are defined which all can be modelled (Menezes *et al.*, 1997).

Exclusive-Or Finally, the XOR function can be modelled by:

$$xor(xor(x, y), y) = x$$
$$xor(xor(x, y), x) = y$$

CHAPTER 3

RELATED APPROACHES

In order to formally verify a protocol we first need to build a mathematical model of that system. Generally two kind of models have been considered in the literature; The Symbolic model and Computational model.

This chapter surveys the related works in the application of formal methods to the analysis of untraceability in RFID protocols. We will present works based on the both symbolic and computational model. We will also compare all definitions in this chapter with our proposed definitions and will provide a classification for untraceability property in next chapter.

3.1 Works in Symbolic Settings

In symbolic settings, the most closest works to our approach are the models provided by Arapinis (Arapinis *et al.*, 2009) and Bruso (Brusó *et al.*, 2010). They presented approaches for verifying protocols based on process algebra, written in applied pi calculus. Their approach models privacy properties as indistinguishability properties using the concept of observational equivalence in applied pi. Moreover in the symbolic world, Deursen et al. proposed a formal definition of untraceability in a particular trace model (Van Deursen *et al.*, 2008).

We start with the works that used process algebra in order to model the RFID protocol and the properties. An RFID protocol is modelled as some processes running concurrently in parallel. As it is mentioned in the previous chapter, to describe processes in the applied pi calculus, we define set of names presenting the communication channels and other constants. A signature Σ which consists of the function symbols which will be used to define terms. Generally in the symbolic settings the cryptographic primitives are represented by function symbols and the messages are considered as the terms on these primitives. One can also describe the equations which hold on terms constructed

3.1.1 Arapinis Model

Arapinis et al. (Arapinis *et al.*, 2009) developed a definition of untraceability in the applied pi calculus independent of the computational models. They defined two levels of untraceability. Strong untraceability that holds if an attacker cannot tell the difference between an RFID system in which all tags are different and a system in which some tags appear twice. The weak untraceability holds if an attacker cannot identify two particular runs of a protocol as having involved the same tag. An RFID protocol due to their framework is modelled as a closed plain process P such that:

$$P = \nu n.(DB|!R|!T)$$

where DB is the database process, R is the reader and T is the process modelling a tag as: $T = \nu m.init.!main$. $init$ is the process of registering the tag to the database DB, and the $main$ models one session of the tag's protocol.

They defined their strong untraceability as ensuring that an intruder thinks that each tag session is initiated by the different tag. It is modelled as a observational equivalence between process P and P' where the process P' defined as: $P' = \nu n.(DB|!R|!T')$ where $T' = \nu m.init.main$ and P is said *strong untraceable* if: $P \approx P'$
In other words the intruder should not be able to tell the difference between the protocol P and the ideal version of P' in which the the tags are allowed to execute themselves at most once.

On the other hand, their *weak untraceability* definition ensures that a tag can execute its protocol multiple times without an intruder being able to link these executions together. This is modelled using the equivalence of two processes Q and Q' which in the former T_1 and T_2 are modelling two different tags such that each one initiated a different session($session_1$ and $session_2$), and in the later T'_1 and T'_2 are modelling two tags such that both of the mentioned sessions are initiated from T'_1.

$$Q = \nu n.(DB|!R|!T|T_1|T_2)$$
$$Q' = \nu n.(DB|!R|!T|T'_1|T'_2)$$

where:

$$T_1 = \nu m.init.(!main|main_{session_1}|main_{session_3})$$
$$T_2 = \nu m.init.(!main|main_{session_2})$$
$$T'_1 = \nu m.init.(!main|main_{session_1}|main_{session_2})$$
$$T'_2 = \nu m.init.(!main|main_{session_3})$$

and Q is said *weak untraceable* if: $Q \approx Q'$.

3.1.2 Bruso Model

Bruso et al.(Brusó *et al.*, 2010) defined untraceability property as equivalences in the applied pi calculus. They expressed their definition based on unlinkability game presented by Chatmon (Chatmon *et al.*, 2006).

In their model, the attacker communicates with a tag through a tag interface, modelled as a channel. $Tag(c)$ presents a complete tag with interface c. It can perform an unbounded number of protocol executions. Then $Tag(c_1, c_2)$ models a single tag (which contains same id) but with two interfaces c_1, c_2. These processes are defined by:

$$Tag(c) = \nu w.\nu s.(!P(w, c)|InitSt(w, s)$$
$$Tag(c_1, c_2) = \nu w.\nu s.!P(w, c_1)|!P(w, c_2)|InitSt(w, s)$$

where: $InitSt(w, s)$ is a process that initializes the tag's state(Tag's unique secret) and the tag main session $P(w, c)$ which is a single tag session that communicates with the rest of the system using two channels. First a restricted channel w to update the tag's state and second, a public channel c used to communicate with the reader. Finally they modelled the complete RFID system as:

$$(ReplTag|Reader|DB)$$

where $ReplTag =!\nu c.Tag(c)$ models an unbounded number of tags, each with own interface.

They defined untraceability using the following scenario. An attacker communicates with two interface which could be either the same tag or different ones. Then the attacker must not be able to distinguish whether two interfaces correspond to the same tag or two different tags. Regarding their model, a protocol satisfies untraceability iff:

$$C\,[Tag(c_1, c_2)] \approx C\,[Tag(c_1)|Tag(c_2)]$$

where C is a system context defined as: $C\,[\ ] = (ReplTag|Reader|DB)$.

3.1.3 Deursen's Trace-based Model

Rather than providing a full description of syntax and semantics of their framework, we will only present the basic requirements needed for understanding their definition of untraceability. A full semantics of this framework can be found in (Cremers et Mauw, 2005).

They defined the security protocol by defining the behaviour of the roles(e.g. initiator, responder) that an agent (e.g. bob, alice) can execute. The agents execute the protocol to achieve some security goal. While agents pursue their goals, an intruder may try to oppose them. The adversary is assumed as a standard Dolev-Yao adversary, who may eavesdrop on any message exchanged between tag and reader, modify or block any message sent from tag to reader or vice versa, and may inject his own messages making them look like they were sent by tag or reader.

A protocol consists of some *event*s. An event is an abstract description of internal events and external events, executed by an agent. An internal event could be a value assignment, an equivalent check, a decryption, a type check, etc. An external event is the action of sending and receiving message.

A role in a security protocol is specified as a sequential list of events executed by an agent. An agent executes its role description sequentially, waiting at receive events until an expected input message becomes available which means that an agent ignores unanticipated messages. For instance we have *initiator(i)* and *responder(r)* role in a protocol.

A role performed by an agent(e.g. *Tag* or *Reader*) is called a *run*. Run is an execution of a role performed by an agent. Within this model a trace t is defined as a sequence of events comes from numbers of interleaved or uncompleted runs. They defined t_R, denoting the subtraces of t, consisting of the events of the run R which are observable by the adversary. The i-th such subtrace is denoted by t_i^R.

They defined untraceability as a trace property of a role in a protocol. Informally, a protocol is called untraceable if for every trace of the protocol in which two sessions are initiated from the same agent, there is a trace that is indistinguishable to the adversary, in which that two sessions are not initiated from the same agent. They used the notion of reinterpretation (Garcia *et al.*, 2005) to discuss about similarity of message consequently indistinguishability of traces.

By their definition two traces are indistinguishable to the adversary, if the adversary cannot see any meaningful difference between the two traces. The trace t is indistinguishable from a trace t' if there is a reinterpretation π such that $\pi(t_i^R = \acute{t}_i^R)$ for all subtraces. Finally they proposed that untraceability is satisfied if:

$$\begin{array}{l} \forall_{t \in Traces} \\ \quad \forall_{i \neq j} \;\; L(t_i^R, t_j^R) \Rightarrow \\ \quad\quad \exists_{\acute{t} \in Traces} \;\; (t \sim t') \wedge \neg L(t_i^R, t_j^R) \end{array}$$

where $L(t_i^R, t_j^R)$ denotes that two subtraces $t_i^R and t_j^R$ are instantiated from the same agent, and $t \sim t'$ means that the trace t is indistinguishable from the trace t'.

We will transform this trace based definition into applied pi and will compare it with other definitions of untraceability in next chapter.

3.2 Works in Computational Settings

Untraceability property is firstly defined in computational settings by means of privacy experiment called games with players: the adversary, against the honest tag and reader in the literature. Avoine in (Avoine, 2005) were the first to give a definition of untraceability in the computational model with a game played between an adversary and a collection of reader and the tag instances. Some other similar attempts then followed like Juels-Weis model (Juels et Weis, 2007) and Vaudenay (Vaudenay, 2007). Ouafi and Phan also proposed a privacy model (Ouafi et Phan, 2008) to demonstrate the notion of untraceability in RFID protocols which might be considered as an alternative definition of Juels-Weis model with some difference in the adversary model. All these work are poorly supported by automatic tools.

Before presenting the works based on privacy experiment, we first introduce the RFID framework used in these games, the adversary model and the concept of untraceability experiment.

RFID System Model. The system is considered as comprising a set of n tags $T_1, \dots T_n$, and a single reader R. Each tag T is a passive transponder identified by a unique ID. It can operate just when interrogated by a reader and only for a short time. It has limited memory and limited computational abilities. The memory of the tags contains a state S, which may change during the lifetime of the tag. The tag's ID may or may not be stored in S. Each tag can perform only basic cryptographic calculations: hash calculations, pseudo random generation, logical operations and symmetric encryption. Tags can also be corrupted. The adversary may have the capability to extract secrets and other parts of the internal state from the tags it chooses. The reader R is a device composed by one or more transceivers and a backend processing subsystem. The reader's task is to identify legitimate tags and to reject all other incoming communication. It is assumed that reader and backend database are linked by a secure channel. Therefore there are two party protocols, one party a tag T and the other a reader R with secure access to backend database system. Each party is a probabilistic interactive Turing machine with an independent source of randomness and unlimited internal storage.

Adversary. The adversary in computational setting is a probabilistic Turing machine which controls the communications between all protocol parties (tag and reader) by interacting with them as defined by the protocol. It is given access to all tags in the system. It can modify the conversation between any pair, and also initiate and terminate a session as her choice. It is also able to learn the outputs of the sessions. The characterization of the adversary is done by specifying the actions that she is allowed to perform, the game she plays and the way she interacts with the system. Different adversarial model have been proposed in the literature such as (Avoine, 2005; Juels et Weis, 2007; Vaudenay, 2007; Ouafi et Phan, 2008; Chatmon *et al.*, 2006) which are different regarding treatment of the adversary's ability.

Untraceability Experiment. Untraceability is firstly defined using the game G played between a malicious adversary A and a collection of tag and reader instances. The idea is that an RFID protocol may be considered private for some parameter values if no adversary has a significant advantage in this experiment. The goal of the adversary in these experiments is to distinguish between two different tags within the limits of its computational power. The success of A in winning game G and thus breaking the notion of Privacy $Priv$ is quantified in terms of A's advantage in distinguishing two cases. This advantage is denoted by Adv_G^A. A wins if her advantage is non-negligible and an RFID protocol may be considered private if the adversary has a negligible advantage in this experiment. In other words if the adversary cannot distinguish two cases with probability higher than random guessing. It is formally written as:

$$Adv_G^A = P[Correct\_Guess] \leq P[Random\_flip\_coin] = \frac{1}{2}$$

In order to better understand the computational model, we provide a sample of such game. Let's consider the adversary as a Probabilistic Polynomially-bounded Turing Machine (PPTM), which interacts with RFID tags and readers through following oracles:

- **Execute**: Models passive attacks. The adversary gets access to the messages exchanged between honest agents in a protocol session by eavesdropping.
- **Send**: Models active attacks. The adversary impersonate a reader in a protocol session and send message to an instance of a tag.
- **Corrupt**: This query allows the adversary to learn the stored secret of the tag.
- **Test**: This query does not correspond to any of adversary's abilities. It allows to define indistinguishability-based notion of untraceability.

Then the privacy game can be divided into three following phases:

- **Learning phase**: Adversary is given access to tags T_0 and T_1 randomly, then she is able to send any *Execute*, *Send*, and *Corrupt* queries of its choice to T_0, T_1, and the *Reader*.
- **Challenge phase**: Adversary chooses a fresh session to send a *Test* query. Depending on a randomly chosen bit $b \in \{0, 1\}$, A is given a tag from the set $\{T_0, T_1\}$. Then it continues to make any *Execute*, *Send*, and *Corrupt* queries.
- **Guess phase**: Adversary outputs a bit $b' \in \{0, 1\}$ and terminates the game G which is the guess of b's value.

The success of A in winning game G and thus breaking the notion of privacy denoted by $Adv_A^{Priv}(k)$ is quantified in terms of A's advantage in distinguishing whether she receives T_0

or T_1 where k is the security parameter. Now we say that a protocol satisfies untraceability if the advantage in privacy game is negligible:

$$Adv_G^A \leq \frac{1}{2} + \epsilon$$

Now we presents some related works which defined untraceability property like the above notion.

3.2.1 Avoine Model

In their model the adversary is a PPTM, which interacts with RFID tags and readers through following oracles. Assume a tag T, and a reader R which participating in the protocol P. They denoted tag instances by π_T^i, and likely reader by π_R^j. The oracles are:

- $Query(\pi_T^i, m_1, m_3)$: Models adversary sending a request m_1 to tag, subsequently sending it the message m_3 after having received its answer.
- $Send(\pi_R^j, m_2)$: Models adversary sending the message m_2 to reader and receiving its answer.
- $Execute(^*)(\pi_T^i, pi_R^j)$: Models adversary executing an instance of the protocol, obtaining the messages from both sides.
- $Reveal(\pi_T^i)$: Models adversary obtaining the contents of the memory of the tag which can be used only once and after that no other queries can not be used.

Within the above adversary model, the authors define two types of untraceability, *Existential* and *Universal.* An existentially untraceable protocol allows the adversary to trace a tag for a restricted period of time, while a universally untraceable protocol does not. The difference between existential and universal comes from the manner of the interactions of the adversary with the tag. Here is a brief scenario of their definition:

- Adversary request for target tag T.
- Adversary interacts with the received tag.
- Adversary request for challenge tags, T_1 and T_2.
- Adversary interacts with both of the tags.
- Adversary decides which of T_1 or T_2 is T, then output the guess.

If the adversary's advantage in guessing the answer is negligible the protocol is said to be *Untraceable.*

3.2.2 Vaudenay Model

Vaudenay proposed a more flexible, hierarchical model for untraceability. His model captures eight classes of adversary capabilities, ranging over four different types of tag corruption and two modes of observation. The adversary of his model has the ability to influence all communications between a tag and the reader, and therefore perform a man-in-the-middle attack on any tag that is within its range. The above framework takes place over following eight oracles that the adversary may invoke:

- *CreateTag(ID)*: Create a free tag with unique identifier
- *DrawTag(distr)* $\rightarrow$ *(vtag)*: Moves from the set of free tags to drawn tags with the probability distribution *distr*. Which *vtag* denotes a virtual tag reference.
- *Free(vtag)*: Moves the virtual tag back to the set of free tags.
- *Launch* $\rightarrow \pi$: Makes the reader launch a new protocol instance.
- *SendReader(m,*π*)* $\rightarrow m'$: Sends a message m to a protocol instance π for the reader and receive the answer m' (resp. *SendTag(m, vtag)* $\rightarrow \acute{m}$)
- *Result(*π*)* $\rightarrow x$: When π is complete returns 1 and 0 otherwise.
- $Corrupt(vtag) \rightarrow S$: Returns the current state of tag S.

Due to this model, privacy game has two phases. An attack phase that the adversary start with issuing oracle queries. Then an analysis phase that the adversary receives the table T that maps every *vtag* to a real tag ID. Then it outputs either true or false. The adversary wins if the output is true. For the complete definition of the oracles and the system (see Vaudenay, 2007).

The adversaries have been divided into different classes, depending on restrictions regarding their use of the above oracles. A *weak* adversary is never allowed to corrupt a tag, that is, he may never query the corrupt oracle. A *forward* private adversary may corrupt a tag at the end of the attack, a *destructive* adversary may corrupt a tag at any time, which leads to the destruction of the tag, that is, the adversary may no longer interact with the tag. A *strong* adversary may corrupt a tag at any time without destroying it.

Orthogonal to these four attacker classes there is a notion of *wide* and *narrow* adversary corresponded to the two modes of observation. An adversary is called wide if he may observe whether the protocol ended successfully, and narrow else. Since the four types of corruption are orthogonal to the narrow/wide separation, eight different adversarial classes are considered.

3.2.3 Juels-Weis Model

The Juels-Weis model (Juels et Weis, 2007) is based on the notion of indistinguishability. They provide a slightly stronger definition of untraceability. Untraceability which is called RFID privacy in their model is defined through a privacy experiment. The adversary selects two tags as candidates. After the challenge, one of the candidates is chosen by the adversary. The simplified version of their defined privacy experiment phases are briefly described below:

1. Setup Phase:
   - Generate random keys $(key_0, \ldots, key_n)$
   - Initialize the reader R with keys
   - Create tags and set keys
2. Learning Phase:
   - Adversary A communicate with the system
3. Challenge Phase:
   - Adversary A select two tags
   - Adversary communicate with the system
   - Adversary outputs a guess bit

The protocol is said to be private if no adversary has non negligible advantage in successfully guessing the tag in the experiment.

Comparing with Vaudenays model it does not have a *DrawTag* query and instead of *Corrupt* query it has a *SetKey* which returns the current secret of the tag and allows the adversary to set a new secret. Moreover, in terms of Vaudenay's model it would considered as a weak adversary. It is worth mentioning that Vaudenay's wide-strong adversary is the strongest of all the adversaries.

CHAPTER 4

FORMALIZATION AND CLASSIFICATION

In formalizing untraceability, most of the attempts have been relied on computational models, typically in term of games. These models are poorly supported by automatic tools. There are also a few definitions of untraceability in symbolic setting, however there is still no agreement on the exact definition and the strength levels.

Therefore we decided to work in symbolic setting, using process algebra. The main idea of these approaches are analysis of the similarity of system's behaviour based on the viewpoint of an external viewer. Employing a symbolic model helps the explicitness of models and provides automatic verification, using tools like ProVerif. Nonetheless, in symbolic analysis we assume perfect cryptography functions which might result in missing attacks, exploiting weaknesses of the cryptographic primitives.

The goal of this chapter is to collect all definitions of untraceability, introducing a classification for this property and providing a symbolic model of the privacy games in applied pi calculus. It also facilitate the process of automatic verification of such protocols.

In section 1, we will classify former privacy games into three different levels and provide their relations. In section 2, we will propose a general model for RFID identification protocols in the applied pi calculus. In section 3, three untraceability levels with respect to the mentioned privacy game types will be offered. In addition, we will show, formalization of these levels in the applied pi calculus. Finally, in section 4, we will categorize all the introduced notions of untraceability.

4.1 Classifying Privacy Games

In this section we would like to classify privacy games in the literature, into three different types. These games have different steps modelling interaction between the adversary and the tag. Let's divide a privacy game into two steps. The first step is the interaction of the adversary with the tags and the readers. Then the adversary's knowledge is tested in the second step. Now, we describe the second steps of the three games in more details, since the first step of the attacker abilities, is almost the same for all.

4.1.1 Game Type I

The game scenario is defined as the adversary is given access to two tags T_1 and T_2, which could be either same tag or a different one. She can eavesdrop on communications and query the tags and the readers in the system. Finally, the game ends up, with the adversary, announcing her guess of whether these tags are the same or not. Untraceability is satisfied if the adversary cannot distinguish the two cases with the probability higher than a random guess. Following scenario describes the above model briefly:

- Adversary is given two tags T_1 and T_2.
- Adversary interacts with both of the tags.
- Adversary decides whether T_1 and T_2 are equal or not, giving out the guess.

According to the 1st game type, The protocol is untraceable, if Adv^{A}_{TypeI} is negligible. Similar games can be found in the works of (Juels et Weis, 2007; Vaudenay, 2007).

4.1.2 Game Type II

The second game is slightly different compared to the previous one. Such that, the adversary A must trace some tag T, therefore she is given a tag instead of two, unlike the previous type. Then the adversary interacts with all tags. Finally, she is given access to a challenge tag T' which must tell whether T' is T or not, better than random guessing to win. It is summarized as following scenario:

- Adversary is given a target tag T.
- Adversary interacts with the received tag.
- Adversary is given a challenge tags, T'.
- Adversary interacts with all tags.
- Adversary decides whether T' is T or not.

The protocol is untraceable according to game type II, if Adv^{A}_{TypeII} is negligible. Followed a similar game of the Chatmon's work (Chatmon *et al.*, 2006).

4.1.3 Game Type III

This game is the strongest concept, among all untraceability models in the literature. Chatmon, in (Chatmon *et al.*, 2006), named this game as unlinkability. It is quite the same as second game type, except that in that one, the adversary already knows T but in this game, both T and T' are challenge tags. Therefore in this scenario the adversary could have access to any two challenge tags T and T', then through interacting with T and T', as well as all other normal tags and readers, adversary must figure whether it is interacting with same tags or not. Adversary wins if his score is non negligible. Adversary game can be summarized as following scenario:

- Adversary A communicate with the system
- Adversary A is given access to two challenge tags T and T'
- Adversary interacts with the system.
- Adversary outputs a guess bit

Due to game type III, the protocol is said to be untraceable, if $Adv^{A}_{TypeIII}$ is negligible. Similar game can be found in the work of (Chatmon *et al.*, 2006).

4.1.4 Comparison

In this section we compare the aforementioned games with respect to the indistinguishability experiment, showing that the first type is the weakest and the last is the strongest among all.

Theorem 4.1. *If a protocol respect type III untraceability it will respect type II untraceability.*

Proof. Let's assume protocol P satisfies third type untraceability, therefore we have:
For any two challenge tags T_1 and T_2, P respects untraceablity thus, the adversary can not distinguish whether it is interacting with same tags.
We want to prove that P satisfies the second type untraceability. Let T' be a given tag, we should prove that the adversary could not tell whether T' is T or not, better than random. We know that this property is admitted for any T_1 and T_2, then it is true for T and T'. □

Theorem 4.2. *If a protocol respect type II untraceability it will respect type I untraceability.*

Proof. Let's assume protocol P satisfies the second type untraceability, therefore we have:
To trace a tag T, the adversary is given access to a challenge tag T' and she is not able to tell whether T' is T or not, better than random guessing.
We want to prove that P satisfies untraceability of the first game. Let T_1 and T_2 be two given tags, we should proof that the adversary could not tell whether T_1 is T_2 or not, better than random. We know that this property is true for T and any given tag T', then it is true for T_1 and T_2. □

Generally, we can conclude that the third type untraceability is the strongest among the others, and also the second type implies the first type untraceability, then in brief we have:

$$\text{3rd game} \quad \Rightarrow \quad \text{2nd game} \quad \Rightarrow \quad \text{1st game}$$

We also comp up with counter examples to show, the above relations can not be true from the other side. We want to demonstrate *1st game* $\not\Rightarrow$ *2nd game* and *2nd game* $\not\Rightarrow$ *3rd game*.

Let's assume a trace of a protocol as depicted in figure 4.1. The shapes are sessions of different or same tags participating in the protocol run. Each color represents a specific tag. Therefore, where two shapes have the same color, it means, those two sessions are initiated by the same tag.

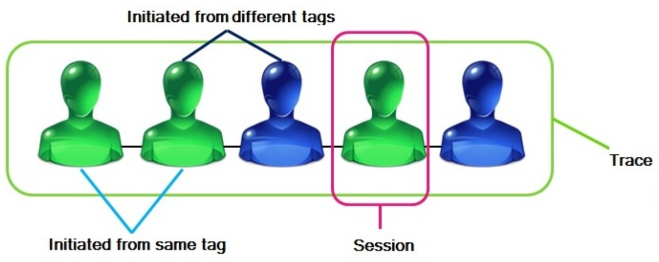


Figure 4.1 A protocol trace

Now consider the traces in figure 4.2. In the right box, you see that if the observer points at any two given sessions, it could not tell weather they are initiated from the same tag or not, therefore it can be said, it satisfies 1st type untraceability. But if the observer sees the output as in the left box, it could see that the first session is initiated from same tag as the third one, so it can not be true for any session, therefore it does not satisfy 2nd type untraceability.

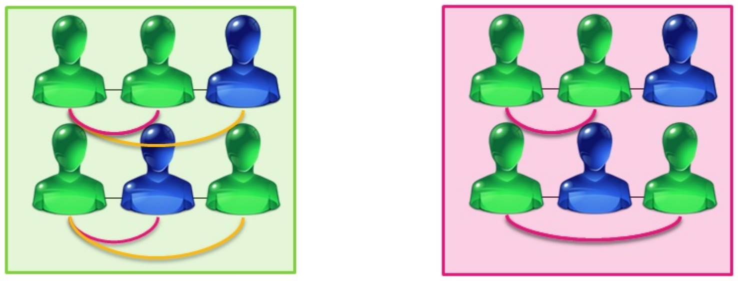


Figure 4.2 1st type Untraceability Vs. 2nd type Untraceability

Now Consider the traces in figure 4.3. This example satisfies 2nd type untraceability. By the definition of the 2nd type game, for the given purple tag, the observer could not find any other session initiated from the same tag in any other traces. But it does not respect the requirement of 3rd type which says that the observer must not link any two session in the traces of the protocol. In this circumstance, the observer is able to find two sessions which are initiated from the gray tag in other traces. Therefore it does not respect the third type game.

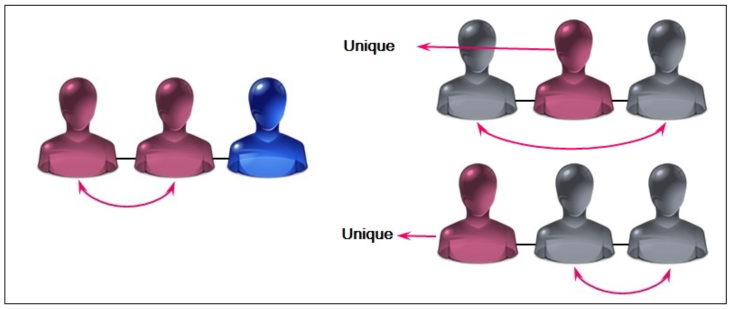


Figure 4.3 2nd type Untraceability Vs. 3rd type Untraceability

4.2 Modelling RFID Protocols

Before formalising security properties, we need to show how we model an RFID protocol in applied pi calculus. Applied pi calculus provides a framework to model security protocols. It allows us to model the interaction between the agents using communication primitives, as well as the cryptographic functions using equational theory. Different protocols often have differences. However, we believe that a large class of RFID protocols can be represented by processes corresponding to the following structure.

An RFID protocol is a closed plain process

$$P \equiv \nu\tilde{n}. \quad (Tag_{\sigma_1}|\dots|Tag_{\sigma_n}|Reader_1|\dots|Reader_n|A_1|\dots|A_k)$$

An RFID system commonly consist of tags and readers. The Tag_{σ_i} are the tag processes which model an unbounded number of tags. Since the link between the reader and the back-end database is secured and also to simplify the model, we assume that back-end database is included in the reader. So, we let $Reader_j$ be the process modelling the reader and the back-end database. The A_ks are the protocol other authorities such as key distributor and the $\tilde{n}$ are channel names. We also define an evaluation context $C[\_]$ which is same as P, but has two holes instead of two of the Tag_{σ_i}s.

The tag model depends on the protocol that is modelled. Typically in an RFID system any tag has a unique secret id_T which differentiates it from other tags and makes it distinguishable for the reader(back-end database). This secret might be a nonce, or an ID or any other secrets.

The secret never transmitted in plain text and depending on each protocol it is transmitted using a hash function, key encryption or logical operations. The tag should be initialised to the back-end database, since this procedure is done before using tag in any environment, we consider it as a *null* process and do not include it our model. We suppose that $l \in dom(\sigma_i)$ is a variables which refers to the id_T. Since each tag can perform an unbounded number of protocol execution, we need to consider different tag sessions. So we use s_i to identify a tag session.

The adversary is considered as a Dolev-Yao attacker in our model, therefore we do not need to explicitly define the adversary and we only give the equational theory describing the intruder. Generally the adversary in our setting has access to any message sent on public channel. These public channels model the network. Cryptographic primitives modelled by means of the equational theory.

4.3 Formalizing Untraceability

In this section we will define three levels of untraceability according to three game scenarios described above. We will show how the untraceability properties described above in term of games, can be formalised in our setting. First we illustrate the informal definition of each property with a simple example then we formally define these concepts in the applied pi calculus, which makes it possible to automatically check if an RFID protocol respect these properties.

4.3.1 Low Untraceable

Low untraceability concept is taken from first game type which ensures that the adversary is not able to infer whether two given sessions s and s' initiated from the same tag in any trace. In this case if an intruder point to any two specific sessions in any trace, she cannot tell whether they have same identities.

Example 4.3. Assume the process P with an unbounded number of tags modelled as $P = !T$. Each tag, identified by a distinct identity id, can run unbounded number of sessions, modelled as $T = \nu id.!(\nu s.main)$ where $main$ models single session of a tag. Every session is associated with a session identifier νs. Now consider two possible branches tr_A and tr_B of the process P which is an equivalent of a trace in trace-based model:

$$\begin{aligned} tr_A \;\equiv\;& \nu id_1.\nu id_2.\nu s_1.\nu s_2.\nu s_3. \\ & (main_{id_1}^{s_1}|main_{id_1}^{s_2}|!(\nu s.main_{id_1})| \\ & main_{id_2}^{s_3}|!(\nu s.main_{id_2})|!T) \end{aligned}$$

$$\begin{aligned} tr_B \;\equiv\;& \nu id_1.\nu id_2.\nu s_1.\nu s_2.\nu s_3. \\ & (main_{id_1}^{s_1}|main_{id_1}^{s_3}|!(\nu s.main_{id_1})| \\ & main_{id_2}^{s_2}|!(\nu s.main_{id_2})|!T) \end{aligned}$$

where $main_{id_i}^{s_j}$ denotes the single session s_j of tag id_i.

In these traces, there are two different tags with identities id_1,id_2 and three sessions of the protocol are executed. To satisfy low untraceability, the adversary must not be able to distinguish whether two given session s_1 and s_2 are executed by the same tag (in tr_A sessions s_1 and s_2 are initiated by the tag id_1), or by two different tags (in tr_B sessions s_1 and s_2 are initiated by the tags id_1 and id_2 respectively). It is worth mentioning here that in this level, the adversary does not see all the outputs of the protocol. If it sees all the outputs may find other two sessions which may initiated from the same tag, like s_1 and s_2 in tr_A and s_1 and s_3 in tr_B which both initiated from tag id_1.

Now we say that an RFID protocol respects low untraceability whenever a process where session s_1 initiated from Tag_A and s_2 initiated from Tag_B, is observationally equivalent to a process where Tag_A initiates both s_1 and s_2. In order to give a reasonable definition of untraceability, we consider two distinct tag sessions s_1 and s_2. Formally it is defined as follows:

Definition 4.4. (Low Untraceability) An RFID protocol respects low untraceability if

$$C\left[Tag_A\{s_1/s\}|Tag_A\{s_2/s\}\right] \approx C\left[Tag_A\{s_1/s\}|Tag_B\{s_2/s\}\right]$$

where $Tag_A = Tag\{id_A/id\}$ and $Tag_B = Tag\{id_B/id\}$ in which id_A and id_B denote two honest tags. Note that we do not consider a separate id for the tags when it is not necessary, instead we use the tag's secret keys in order to identify tag's identity.

The intuition is that if an intruder cannot detect if two given single sessions are initiated by the same tag or by two different tags, then in general she cannot link between the two tags which results in tag's untraceability.

It is classical to formalise untraceability property through some kind of observational equivalence in a process calculus. This method relies on the two notions introduced in chapter 2, static equivalence $\approx_S$, and labelled bisimilarity $\approx_L$. Static equivalence captures the static part of observational equivalence and labelled bisimilarity captures the dynamic part. Abadi and Fournet in (Abadi et Fournet, 2001) stated that observational equivalence and labelled bisimilarity coincide: $\approx\ =\ \approx_L$. They also proved that given closed extended processes A, B and a closed evaluation context $C[\_]$, we have $A \approx_L B$ implies $C[A] \approx_L C[B]$. It is very important, because labelled bisimilarity can be proved instead of observational equivalence avoiding to have to deal with the universal quantification over evaluation contexts, required by the observational equivalence.

Furthermore, since the only difference between two processes P and P' lies in the tag processes, It is only required to verify that:

$$(Tag\{id_A/id, s_1/s\}|Tag\{id_A/id, s_2/s\}) \approx (Tag\{id_A/id, s_1/s\}|Tag\{id_B/id, s_2/s\})$$

The tool ProVerif enables us to prove such equivalence in presence of an adversary. Note that the adversary is not modelled explicitly, she is considered as a part of the environment and the observational equivalence guarantees that no environment will be able to distinguish the two cases. In our research we benefit from this tool to demonstrate automatic verification of the proposed properties for some case study protocols. Following two properties also could be verified in the same way.

4.3.2 Mid Untraceable

This definition is inspired from the game type II. The adversary wants to trace a tag T with chosen session s, then she is given access to any session s'. In this case the adversary must not be able to infer whether s' is initiated from the same tag as s is initiated from.

Example 4.5. Let's consider the same process in example 4.3, the branches tr_A, tr_B and tr_C are defined as follows:

$$\begin{aligned} tr_A \equiv\ & \nu id_1.\nu id_2.\nu s_1.\nu s_2.\nu s_3. \\ & (main^{s_1}_{id_1}|main^{s_2}_{id_1}|!(\nu s.main_{id_1})| \\ & main^{s_3}_{id_2}|!(\nu s.main_{id_2})|!T) \end{aligned}$$

$$\begin{aligned} tr_B \equiv\ & \nu id_1.\nu id_2.\nu s_1.\nu s_2.\nu s_3. \\ & (main^{s_1}_{id_2}|main^{s_3}_{id_2}|!(\nu s.main_{id_2})| \\ & main^{s_2}_{id_1}|!(\nu s.main_{id_1})|!T) \end{aligned}$$

$$\begin{aligned} tr_C \equiv\ & \nu id_1.\nu id_2.\nu s_1.\nu s_2.\nu s_3. \\ & (main^{s_1}_{id_2}|main^{s_2}_{id_2}|!(\nu s.main_{id_2})| \\ & main^{s_3}_{id_1}|!(\nu s.main_{id_1})|!T) \end{aligned}$$

In tr_A there are two sessions s_1 and s_2 which instantiated by the tag id_1, but in other two branches tr_B and tr_C, you could see that there is only one instantiation of tag id_1. Therefore it satisfies the mid untraceability which requires that for the given session s initiated from the tag id_1 there is no s' in other traces which initiated from the tag id_1.

Notice that in the above branches of the protocol run, there might be other sessions than the given session which initiated from the same tag, like s_1 and s_3 in tr_B and s_1 and s_2 in tr_C which instantiated from the same tag id_2. This shows the reason why we have middle level of untraceability and it is not the strongest.

To formalize this property, we consider a given session s_1 initiated from Tag_A. We say that RFID protocol respects mid untraceability whenever a process including s_1 and s_2 which is initiated from the same tag as s_1 be observationally equivalent with a process including s_1 and s'_2 which is initiated from a different tag from which s_1 is initiated from. Formally it is defined as follows.

Definition 4.6. (Mid Untraceability) An RFID protocol respects mid untraceability if

$$C\left[Tag_A\{s_1/s\}|Tag_A\right] \approx C\left[Tag_A\{s_1/s\}|Tag_B\right]$$

where Tag_A and Tag_B defined in the same way as in low untraceability as $Tag\{id_x/id\}$. The intuition is that with a given session s_1 if an adversary cannot detect that for any second session, it is executed by the same tag or by different tag, then it can not link between two execution of tags and results in tag's untraceability.

4.3.3 High Untraceable

This level of untraceability is a strong notion of untraceability. It is inspired from the game type III, that the adversary must not be able to infer whether there exist two sessions in any trace that initiated from the same tag.

The difference between high and mid untraceability comes from the challenge session that in mid untraceability adversary already knows the challenge session s, but in high untraceability both s and s' are challenge sessions.

Example 4.7. Let's consider the same process in example 4.3, the branches tr_A and tr_B are defined as follows:

$$\begin{aligned} tr_A &\equiv \nu id_1.\nu id_2.\nu s_1.\nu s_2.\nu s_3. \\ &\quad (main^{s_1}_{id_1}|main^{s_2}_{id_1}|!(\nu s.main_{id_1}| \\ &\quad main^{s_3}_{id_2}|!(\nu s.main_{id_2}|!T) \end{aligned}$$

$$\begin{aligned} tr_B &\equiv \nu id_1.\nu id_2.\nu id_3.\nu s_1.\nu s_2.\nu s_3. \\ &\quad (main^{s_1}_{id_1}|!(\nu s.main_{id_1}| \\ &\quad main^{s_2}_{id_2}|!(\nu s.main_{id_2}| \\ &\quad main^{s_3}_{id_3}|!(\nu s.main_{id_3}|!T) \end{aligned}$$

In the above example, you could find no session s' in tr_B that initiated from the same tag as session s in tr_A is initiated from. As you can see in the above traces there must not be any link between any two sessions to fulfil the high untraceability.

High untraceability is formally defined in the similar way as low untraceability with a slight difference. In low untraceability, sessions are given as inputs to the model, which in high untraceability modelled using new operators ν that means that for any sessions it should be true. We say that an RFID protocol respects high untraceability whenever a process including two different tags with an unbounded number of sessions, is observationally equivalent to a process where sessions are initiated from one tag. Formally it is defined as follows.

Definition 4.8. (High Untraceability) An RFID protocol respects high untraceability if

$$C\left[Tag\{id_A/id\}|Tag\{id_A/id\}\right] \approx C\left[Tag_{\{}id_A/id\}|Tag\{id_B/id\}\right]$$

The intuition behind the definition is that any sessions of protocol P should look to the adversary as if it was initiated by a different tag. In other words the adversary can not be able to infer whether there exist two sessions initiated by the same tag in any trace.

4.4 Classification of Untraceability Definitions

Now we are able to compare between all untraceability notions such as Arapinis, Bruso, and our proposed properties. In addition to the presented definitions based on process equivalence in applied pi, Deursen in (Van Deursen *et al.*, 2008) defined a model based on trace equivalence. We briefly present their framework to be able to also relate their notion of untraceability with ours. In this section we provide the relation between all formal untraceability definitions in the literature. Before comparing the models, since the model proposed by Deursen is based on traces, we need a transformation from trace-based model to process algebra. We first provide this transformation using some examples.

As we presented in related works, Deursen et al. defined their definition in terms of traces(Van Deursen *et al.*, 2008). In order to compare these two models we need to transform the trace based model into process algebra. Therefore we start with modelling a protocol.

They defined the security protocol by defining the behaviour of the roles, therefore we assume a closed plain process P including l roles, defined as follows:

$$P = \nu \tilde{n}.(R_1 | \ldots | R_l)$$

where R_i models the i_{th} role of the protocol. Moreover, we consider P such that all channels occurring in it are ground, and private channels are never sent on any channel.

A role consists of a sequence of events that an agent can execute, So the agents are instances of the roles. To consider an unbounded number of agents, we put the R_is under replication, so we will have:

$$P = \nu \tilde{n}.(!R_1 | \ldots | !R_l)$$

Now we should consider an unbounded number of sessions for each role. Each role also should have a session identifier. In addition we consider a distinct identity for each user so that we would be able to find out whether two sessions are instantiated by same or different agents. Therefore we could define a role R_i as follows:

$$R_i = \nu id.\nu \tilde{m}.!(\nu s.main_i)$$

where id is user identity, s is session identifier and $main$ is the process models one session of a role.

Example 4.9. Consider an RFID toy protocol including two roles: Reader R and tag T. The protocol starts with sending a request from the reader to the tag. Then the tag responds hello to the reader. The adversary is assumed as a standard Dolev-Yao adversary that has the full control over the network. This protocol can be modelled by the following closed plain process:

$$\begin{aligned} P &= (!R|!T) \\ R &= \nu id.(\nu s.\bar{c}\langle req\rangle.c(x)) \\ T &= \nu id.(\nu s.c(x).\bar{c}\langle hello\rangle) \end{aligned}$$

Within this model, a trace t is defined as a sequence of events coming from numbers of interleaved or uncompleted runs. They defined t_R, denoting the subtraces of t, consisting of the events of the run R, which are observable by the adversary. The i-th such subtrace is denoted by t_i^R. By knowing that the only observable parts by the adversary in a protocol run, are the messages sent and received through public channel, we could say that a subtrace t_R represents all the communications that a specific tag does within a protocol run. Then the subtrace t_i^R corresponds to a specific session of t_R. Moreover, the notion of linkability of subtraces in their model, denoted by $L(t_i^R, t_j^R)$, represents the concept that two sessions are initiated from the same agent.

To better understand the above trace based model, we illustrate an example of equivalent of a trace in terms of applied pi calculus.

Example 4.10. Consider the following processes:

$$P = !T \qquad\qquad T = \nu id.!(\nu s.\bar{c}\langle hello\rangle)$$

where, P denotes a protocol with an unbounded number of tags and the process T, models a tag that has a distinct id, indicated by νid. Each tag executes an unbounded number of sessions, indicated by the bang operator !, and each session involves outputting the message "hello" on the public channel c. Every session is also associated with a distinct session identifier, indicated by νs. A possible branch of the process P could be:

$$\begin{aligned}
P \quad &\equiv \quad (!T|\nu id_1.!(\nu s.\bar{c}\langle hello\rangle)|\nu id_2.!(\nu s.\bar{c}\langle hello\rangle)| \\
&\qquad\ \nu s_1.\bar{c}\langle hello\rangle|\nu s_2.\bar{c}\langle hello\rangle) \\
\\
&\equiv \quad \nu id_1.\nu id_2.\nu s_1.\nu s_2. \\
&\qquad\ (!T|!(\nu s.\bar{c}\langle hello\rangle)|!(\nu s.\bar{c}\langle hello\rangle)| \\
&\qquad\ \bar{c}\langle hello\rangle|\bar{c}\langle hello\rangle); \qquad \text{Using the rule: } A|\nu u.B \equiv \nu u.(A|B) \\
\\
&\xrightarrow{\nu x.\bar{c}\langle x\rangle} \quad \nu id_1.\nu id_2.\nu s_1.\nu s_2. \\
&\qquad\ (!T|!(\nu s.\bar{c}\langle hello\rangle|!(\nu s.\bar{c}\langle hello\rangle)| \\
&\qquad\ (\bar{c}\langle hello\rangle|\{hello/x\}) \\
\\
&\xrightarrow{\nu y.\bar{c}\langle y\rangle} \quad \nu id_1.\nu id_2.\nu s_1.\nu s_2. \\
&\qquad\ (!T|!(\nu s.\bar{c}\langle hello\rangle|!(\nu s.\bar{c}\langle hello\rangle)| \\
&\qquad\ (\{hello/y\})
\end{aligned}$$

which represents two agents with the identities id_1 and id_2 executing two sessions. The sessions may executed by the same agent, or different agents.

Proposition 4.11. If a protocol is low untraceable, it will satisfy untraceability proposed by Deursen.

Proof. To compare the definitions of untraceability we have to understand the definition of untraceability in terms of our language. Deursen defined untraceability as follows:

$$\begin{aligned}&\forall_{t \in Traces}\\ &\quad \forall_{i \neq j}\ \ L(t_i^R, t_j^R) \Rightarrow\\ &\qquad \exists_{t' \in Traces}\ \ (t \sim t') \wedge \neg L(t_i^R, t_j^R)\end{aligned}$$

It is inferred from the above definition that, untraceability is satisfied if for every trace of the protocol in which, two subtraces are instantiated by the same agent $\forall_{i \neq j}\ \ L(t_i^R, t_j^R)$, there exists another indistinguishable trace $(t \sim t')$, in which two subtraces are not instantiated from the same agent $\neg L(t_i^R, t_j^R)$.

This is similar to our definition of low untraceability with a slight different, both definitions requires that the observer cannot distinguish whether two given single sessions are initiated by the same tag or by two different tags. But their definition only requires to exist only one indistinguishable trace. This makes their definition weaker than ours and we could conclude that our low untraceability implies Deursen's untraceability. □

Proposition 4.12. A protocol is low untraceable if and only if it is weak untraceable.

Proof. Let's assume protocol P satisfies low untraceability, therefore by hypothesis we have the following:

$$\begin{gathered}P \equiv Tag\{id_A/id, s_1/s\}|Tag\{id_A/id, s_2/s\}|A_1|\ldots|A_n\\ \approx\\ P' \equiv Tag\{id_A/id, s_1/s\}|Tag\{id_B/id, s_2/s\}|A_1|\ldots|A_n\end{gathered}$$

where P represents the system in which two sessions s_1 and s_2 are executed by the same tag (id_1); and P' represents the system in which session s_1 is initiated by the tag id_1, respectively s_2 is initiated by tag id_2. According to Arapinis's weak definition, an RFID protocol Q, modelled as:

$$\begin{gathered}Q = \nu n.(DB|!R|!T|T_1|T_2)\\ T_1 = \nu m.init.(!main|main_{session_1}|main_{session_2})\\ T_2 = \nu m.init.(!main|main_{session_3})\end{gathered}$$

must be equivalent to Q' modelled as:

$$\begin{gathered}Q' = \nu n.(DB|!R|!T|T_1'|T_2')\\ T_1' = \nu m.init.(!main|main_{session_1}|main_{session_3})\\ T_2' = \nu m.init.(!main|main_{session_2})\end{gathered}$$

Considering that a session s_1 in $Tag\{s_1/s\}$ is modelled using $main_{s_i}$ in Arapinis framework, $Tag\{id_A/id\}$ in our model corresponds to T_1 and respectively $Tag\{id_B/id\}$ corresponds to T_2 in Arapinis model. So processes P and Q represent two protocols, both including two tag processes that refers to the same tag, modelling two distinct sessions initiated by the same tag. Therefore we have:

$$P \quad \equiv \quad Q$$

and respectively for process P' and Q' we could show that $P' \equiv Q'$. □

Proposition 4.13. If a protocol is untraceable due to Bruso's definition, it will satisfy low untraceability.

Proof. Let's assume protocol P satisfies Bruso's untraceability, we have the following equivalence:

$$\begin{array}{rcl} P \equiv (R|DB|Tag(c_1, c_2)) & \approx & (R|DB|Tag(c_1)|Tag(c_2)) \equiv P' \\ (R|DB|!main_s(c_1)|!main_s(c_2)) & & (R|DB|!main_{s_1}(c_1)|!main_{s_2}(c_2)) \end{array}$$

where P represents two execution of a same tag, but on a different interfaces where each one can executes infinitely many sessions; and P' represents a system in which two independent tags executes infinitely many sessions. Note that since processes R and DB models the same process on both definitions and the only difference is in tag process, they could be considered as null. Now we have to prove that protocol P satisfies following equivalence relation:

$$\begin{array}{rcl} \nu n.(DB|!R|!T|T_1|T_2) & \approx & \nu n.(DB|!R|!T|T_1'|T_2') \\ T_1 \equiv (\nu s_1.main_{id_1}) \quad \& T_1 \equiv \nu s_2.main_{id_1}) & & T_1' \equiv (\nu s_1.main_{id_1}) \quad \& \quad T_2' \equiv (\nu s_2.main_{id_2}) \end{array}$$

The main difference between two above models is that, Bruso's model requires a system with a tag executing *infinitely many sessions* in contrast with our model which considers only *single session* in the indistinguishability game. Therefore it is trivial that Bruso's definition is stronger and implies our definition. □

Proposition 4.14. If a protocol is strong untraceable due to Arapinis, it will satisfy our High untraceability.

Proof. Let's assume protocol P satisfies Arapinis's strong untraceability, we have the following equivalence:

$$\begin{array}{rcl} \nu n.(DB|!R|!T) & \approx & \nu n.(DB|!R|!T') \\ \nu n.(DB|!R|!(!main) & & \nu n.(DB|!R|!(main)) \end{array}$$

This equivalence requires that the RFID system with unbounded number of tags where each can execute multiple sessions, be equivalent to one where each tag executes only one session since the macro presenting tag's session in process T' is not under replication. But our definition only consider two tags in the indistinguishability game. Therefore it is obvious that Arapinis definition is stronger. It is worth mentioning that, although their definition is the strongest notion of the untraceability, real-life protocols can not satisfy this property since it requires a tag only execute one session. $\square$

4.5 Results and Discussions

In this chapter we discussed about all the definitions of the untraceability in the literature and suggest a new classification for this property in RFID protocols.

In Arapinis model, the strong property requires that the RFID system with unbounded number of tags where each can execute multiple sessions, be equivalent to one where each tag executes *only one session* since the macro presenting tag's session in process T' is not under replication.

This model suffers from an attack in which the attacker is in the proximity of the tag and queries the same tag multiple time in a short period, knowing that, it is the same tag. Assume that T and T', pointing the same tag and each has a single session. Within this model, since a tag interface always corresponds to the same tag, the adversary is not able to choose the tag which communicating with. The adversary has also the ability to query a tag several times and might get response from any tag. Regarding their strong property, She must not be able to distinguish these two tags but their proposed model makes it impossible to satisfy the definition that has only a single session unless a different definition is given to the single tag session. This could be a main blind spot of such definition.

On the other hand, Arapinis's weak untraceability, requires a system in which two sessions are executed by a particular tag, to be equivalent to a system where one of these two sessions is executed by a different tag. This model is somehow similar to Bruso's model. In both models the adversary must not be able to distinguish between a system including two executions of the same tag and a system including the execution of two independent tags. But there is a difference in these two model. Bruso's model requires a system with a tag executing *infinitely many sessions*, while Arapanis model, considers only a *single session* in the indistinguishability game. This model also does not work properly in state based protocols like the protocol proposed by Ha et al. (Ha *et al.*, 2007), ProVerif verifies the protocol untraceablility, which could not be true due to the following simple scenario.

Consider a tag communicating with a reader. Depending on the tag's current session, two scenarios could occur on reader side: the reader might complete the run by responding the tag or it terminates the execution of the protocol by rejecting the tag in the case of getting an incorrect respond from tag. Assuming that the tag is in its first session(initial state) the reader cannot verify whether the message sent by the tag has changed during transmission. Thus, malicious modification of this message does not result in rejection of the tag by the reader. So the adversary performs a man-in-the-middle attack. She gets a challenge from the reader and sends it to the tag to obtain a response. Then, she replaces the message, provided by the tag, with a different value and submit the response to the reader. If the reader rejects the response, the tag was in the middle of its session and if the reader accepts the response, the tag was in the initial state. Therefore the adversary gets some information about the tag which could result in tracing the tag.

In view of aforementioned reasons, we propose definitions for three levels of untraceability. So we reviewed untraceability definitions based on privacy games and divided them into three types. Then we proposed three untraceability levels corresponding to each game type with some examples, to clarify the difference between these levels. In brief we divide untraceability into the following levels:

- Low Untraceability, as ensuring that the adversary can not be able to distinguish whether two given sessions in any trace initiated from the same tag or two different tags.
- Mid Untraceability, as ensuring that for a given session, the adversary must not be able to infer that there exist another session, initiated from the same tag, as the given session is initiated from.
- High Untraceability, as ensuring that the adversary must not be able to infer whether there exist two sessions in any trace that initiated from the same tag.

We define these properties using observational equivalence in applied pi calculus, so that Proverif gives us the ability to verify them. We also present the relationship between our definitions and the definitions in the literature. Table 4.1 depicts the results of comparison between the proposed untraceability levels as met by different definitions in the literature. We will also illustrate some case studies on different RFID protocols from the literature in the following chapter.

Table 4.1 Comparison of Untraceability properties

$$\begin{tikzcd}
 & & \text{Bruso Unt.} \arrow[d, Rightarrow] \\
\text{High untraceability} \arrow[r, Rightarrow] & \text{Mid untraceability} \arrow[r, Rightarrow] & \text{Low untraceability} \arrow[d, Leftrightarrow] \\
\text{Arapinis Strong Unt.} \arrow[u, Rightarrow] & & \text{Arapinis weak Unt.} \arrow[d, Rightarrow] \\
 & & \text{Deursen Untraceability}
\end{tikzcd}$$

CHAPTER 5

CASE STUDIES

This chapter illustrates how to automatically verify the proposed untraceability property in RFID protocols. As mentioned by Deursen, it was a problem in such protocols (van Deursen et Radomirovic, 2009). Many protocols like (Sun et Zhong, 2012; Kim *et al.*, 2007; Qi *et al.*, 2012; Bassil *et al.*, 2012) have been proposed and has not been automatically verified yet. In this chapter we cover the protocol proposed by Kim et al (Kim *et al.*, 2007) along with the protocol presented by Feldhofer (Feldhofer *et al.*, 2004). Our third case study is the sample protocol used by Arapinis (Arapinis *et al.*, 2009), and the last one which belongs to the group of hash-based, single step protocols is proposed by Ohkubo et al. (Ohkubo *et al.*, 2003).

In this chapter, we employ the tool ProVerif(Blanchet *et al.*, 2001) to model our case study protocols and verify the properties. Before going through the case studies, we briefly introduce the tool and present how to verify secrecy, correspondence and equivalence properties.

5.1 ProVerif

ProVerif is a fully automatic tool for analysing the security of cryptographic protocols. It can handle many different cryptographic primitives including symmetric and asymmetric encryption, digital signature and hash functions. ProVerif is capable of proving reachability properties, correspondence assertions, and observational equivalence which can be useful to analysis of secrecy, authentication, privacy and traceability properties. Its architecture is depicted in Figure 5.1. ProVerif accepts two kinds of input files: Horn clauses and the Applied Pi calculus. It gives us the ability to automatically prove following properties in order to analyse security protocols. Recently R. Kuster and T. Truderung (Küsters et Truderung, 2011) proposed a new tool XOR-ProVerif which is a small program, transforms a protocol using Exclusive-Or mechanism into a protocol in Horn clauses compatible with ProVerif.

Secrecy Intuitively, the secrecy of the term M is preserved in a protocol if it is only known to participants which are entitled to access it and no adversary can obtain M by substitution and deduction from output of the protocol. The adversary is formalized as a process running in parallel with the protocol which outputs M on a public channel after constructing it. If she cannot construct M, the secrecy is preserved. In ProVerif to test secrecy of the term M

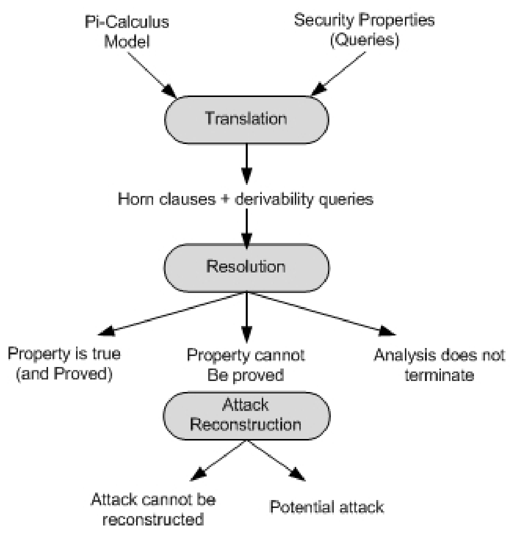

Pi-Calculus Model
Security Properties (Queries)
Translation
Horn clauses + derivability queries
Resolution
Property is true (and Proved)
Property cannot Be proved
Analysis does not terminate
Attack Reconstruction
Attack cannot be reconstructed
Potential attack

Figure 5.1 ProVerif Architecture

in the model, the following query is included in the input file before the main process: *query attacker(M).* where M is a ground term.

Authentication (Correspondence properties) To reason with correspondence properties processes are annotated with events, marking important stages reached by the protocol which do not otherwise affect behaviour. Events are analogous to breakpoints used in software development. Correspondence properties are used to represent the relationships between events that can be expressed in the form "if an event e has been executed then event e' has been previously executed." Authentication can be captured using correspondence assertions. For example, recall the handshake protocol authentication property that the client is only willing to share her secret with the server S; it follows that, *if* she completes the protocol, *then* she believes she has done so with S and hence authentication of S to C should hold. In contrast, server S is willing to run the protocol with the client who he is willing to, and hence at the end of the protocol he only expects authentication of C to S to hold, if he believes C was indeed his interlocutor. Accordingly, the grammar for processes including events are extended with *event* $e(M_1, \dots, M_n)$;. The syntax to query a basic correspondence assertion in ProVerif is:

query $x_1 : t_1, \dots, x_n : t_n$; *event*$(e(M_1, \dots, M_j)) \rightarrow$ *event* $(e'(N_1, \dots, N_k))$.

Observational Equivalence The notion of indistinguishability is a powerful concept which allows us to reason about complex properties that cannot be expressed as reachability or correspondence properties. Indistinguishability is generally named observational equivalence in the formal model. If an agent receives a message or an adversary eavesdrop a message for which it does not have the decryption key, this message looks just like a random bit string. Intuitively, two sequences of messages look the same to an agent and the agent understands the same form both of the messages, these messages are called observational equivalence. In other words if a message looks like a random bit string to an agent, then the corresponding message also looks like a random bit string.

Strong secrecy A first class of equivalences that ProVerif can prove is strong secrecy. Strong secrecy means that the attacker is unable to distinguish a session of the protocol where a data m has been used from a session where m has been replaced by m'. In other words, the value of the secret should not affect the observable behavior of the protocol.

Equivalences between processes that differ only by terms ProVerif includes some queries that enable us to prove the most general class of equivalences where the processes P and Q have the same structure and differ only in the choice of terms. Intuitively, two processes P and Q are observationally equivalent, written $P \approx Q$, if they can output on the same channels, no matter what the context they are placed inside. Roughly speaking, processes P and Q are said to be observationally equivalent when an active adversary cannot distinguish P from Q where the processes P and Q have the same structure and differ only in the choice of terms. This is written in ProVerif by a single biprocess that encodes both P and Q. Such a biprocess uses the construct ”$choice[M, M']$“ to represent the terms that differ between P and Q. P uses the first component of the choice, M, while Q uses the second one, M'.

The most general class of equivalences that ProVerif can prove are equivalences $P \approx Q$ In other words if they can output on the same channels, no matter what the context they are placed inside. Roughly speaking, processes P and Q are said to be observationally equivalent when an active adversary cannot distinguish P from Q where the processes P and Q have the same structure and differ only in the choice of terms. These equivalences are written in ProVerif by a single biprocess that encodes both P and Q. Such a biprocess uses the construct ”$choice[M, M']$“ to represent the terms that differ between P and Q. P uses the first component of the choice, M, while Q uses the second one, M'.

5.2 Feldhofer Protocol

In this section, we study a mutual authentication protocol proposed by Feldhofer et al. in (Feldhofer *et al.*, 2004). We first present an informal definition of the protocol. Then, we model it in the applied pi calculus. Deursen in (Van Deursen *et al.*, 2008) proposed that, it is untraceable regarding their definition of untraceability. Here we show that, the protocol neither satisfies Mid and Hight untraceability, nor Bruso's untraceability. It only satisfies the Low untraceability property. The protocol is depicted in figure 5.2 using a message sequence chart.

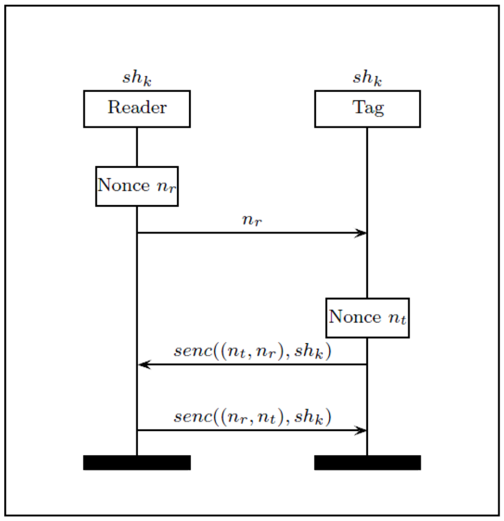


Figure 5.2 The Feldhofer Mutual Authentication Protocol

5.2.1 Description

The protocol includes a reader, a tag and a key distributor. During the first phase, every pair of reader, *Reader* and tag, Tag receives a unique key k. These shared keys are initially not part of the adversary's knowledge. The reader initiates the protocol by sending a freshly generated nonce n_r to the tag. The tag generates a nonce n_t encrypts the pair (n_r, n_t) under the shared key k, and sends it to the reader. The reader decrypts the message using the same shared key, reverses the order of the two nonces, encrypts the message under the shared key, and sends it to the tag.

5.2.2 The Model In Applied Pi

The protocol is defined with respect to the signature $\Sigma = \{senc, sdec\}$. Cryptography is modelled in a Dolev-Yao style as being perfect. So the cryptographic primitives are modelled as an equational theory. The encryption function is modelled by the equation: $sdec(senc(m, k), k) = m$; where both *senc* and *sdec* are binary functions, representing symmetric key encryption and decryption. The ProVerif code used to model the protocol is shown in table 5.4.

Table 5.1 Feldhofer protocol ProVerif code

```
(* a generic symmetric encryption function *)
fun senc/2.
reduc sdec(k, senc(k, m)) = m.


(* Key Distributor Process *)
let Key =
new sk;
out(privcR,sk);
out(sk1Ch,sk).


(* Tag Process *)
let Tag =
in(privcT,sk);
in(c,nr);
out(c,senc((nt,nr),sk));
in(c,msg1);
let (ntt,nrr)=sdec(msg1,sk) in
if (ntt=nt) then
if (nrr=nr) then 0.


(* Reader Process *)
let Reader =
in(privcR,sk);
new nr;
out(c,nr)
in(c,msg2)
let(nt,nrr)=sdec(msg2,sk) in
if (nrr=nr) then
out(c,senc((nr,nt),sk)).


(* Main process *)
process new nt; ( Key | Reader |
(let privcT = sk1Ch in let nt = nt2 in Tag) )
```

In the proposed protocol model, defined processes are, the tag, the reader, the key distributor and the main process.

Main Process. The main process specifies the parallel combination of the processes and sets up the private channels that mainly used for key distributions. We model the above protocol for only one tags and launch a copy of the reader for the tag. To model the untraceabilities two tags are needed to be run in parallel, then we launch two copies of the readers, one for each tag.

Key Process. Our model includes a dedicated process for generating and distributing keying material for shared key encryption.

Tag Process. Within this process, first, each tag obtains its secret key from the key process. The remainder of the specification follows directly the informal description given above. The statement "$let(nt, nrr) = sdec(msg2, sk)\ in$" in this process uses destructor and pattern matching with type checking to verify that if the first and the second element are the readers's and respectively its own nonce.

Reader Process. The reader process also follows the informal description of the reader mentioned above, after getting its shared key from the key process.

We simulated the secret key sharing through sending message via secret channels. We did not model tag identities, instead we considered the shared key as tag identifier. In this thesis the aim of our work is to investigate the untraceability achieved by our modelled protocols. According to our proposed definition high untraceability holds when it is impossible for an attacker to distinguish between a system in which two sessions s_1 and s_2 are initiated from tag_A and a system where s_1 is initiated from tag_A and s_2 is initiated from tag_B. In formal terms, this means proving the equivalence ($\approx$) of the following processes:

$$P \equiv Key|Reader|Reader|Tag_A|Tag_B \approx Key|Reader|Reader|Tag_A|Tag'_B \equiv P'$$

$$\begin{aligned} Tag_A &= \nu sk_1.\nu sk_2.Tag\{sk_1/sk\} \\ Tag_B &= \nu sk_1.\nu sk_2.Tag\{sk_1/sk\} \\ Tag'_B &= \nu sk_1.\nu sk_2.Tag\{sk_2/sk\} \end{aligned}$$

$$Tag = privcT(sk).c(nr).c\langle senc((nt, nr), sk)\rangle.c(msg1)$$

where P represents the system in which two single sessions are initiated by two different tags and P' represents the system in which two single sessions are initiated by the same tag. The session identifiers coincide with the nonces in our model and the ids represented by the secret keys. The notation $Tag\{sk_i/id\}$ indicates the execution of the protocol using the tag id_A.

5.2.3 Automatic Verification Results

We first used ProVerif to verify the confidentiality of the secret values, sent through the protocol. Then we checked the authentication properties by adding required events to the both beginning and end of the tag and the reader processes as follows:

Table 5.2 Feldhofer protocol - Correspondence properties

```
(* Tag Process *)
let Tag =
event beginRparam(tag);    in(privcT,sk);
in(c,nr);
out(c,senc((nt,nr),sk));
in(c,msg1);
let (ntt,nrr)=sdec(msg1,sk) in
event beginRfull(tag, nr, nt, sk);
if (ntt=nt) then
if (nrr=nr) then 0;
event endTparam(tag);
event endTfull(tag, nr, nt, sk).


(* Reader Process *)
let Reader =
event beginTparam(reader);
in(privcR,sk);
new nr;
out(c,nr)
in(c,msg2)
let(nt,nrr)=sdec(msg2,sk) in
event beginTfull(reader, nr, nt, sk);
if (nrr=nr) then
event endRparam(reader);
event endRfull(tag, nr, nt, sk);
out(c,senc((nr,nt),sk)).
```

Table 5.3 Feldhofer protocol - secrecy and correspondence queries

```
query attacker:nt;
attacker:sk.
query evinj:endTparam(x) ==> evinj:beginTparam(x).
query evinj:endTfull(x1,x2,x3,x4) ==> evinj:beginTfull(x1,x2,x3,x4).
query evinj:endRparam(x) ==> evinj:beginRparam(x).
query evinj:endRfull(x1,x2,x3,x4) ==> evinj:beginRfull(x1,x2,x3,x4).
```

Then by adding the following queries to the protocol model the ProVerif will check for the results, to see if they are true or false.

After getting the true results for the above queries, we can say that the secrecy and mutual authentication properties are satisfied by Feldhofer Protocol. Now in order to check our untraceability we use choice query to make sure that an adversary cannot distinguish P from Q where the processes P and Q both have the same structure and differ only in the second tag's id.

The verification results by ProVerif also shows that this protocol which is said to be untraceable by Deursen, only satisfies our low untraceability definition. This is due to the fact that in low untraceability the attacker should distinguish the two given session of the protocol which does not include the complete outputs of the protocol. Because in low untraceability we only investigate two specified sessions of tags which provided by the main process.

Now, in order to check whether it satisfies Arapinis strong definition we should model the equivalence between a system in which the protocol executed by each tag at most once, and a system in which tags can execute the protocol more than once, modelled as follows:

$$Q \;=\; !R|!T \;\approx\; !R|!T \;=\; Q'$$

$$\begin{aligned} T &= \nu id.!(\nu n_t.in(c,nr). \\ &\quad out(c, senc((nt,nr),sk)).in(c,msg)) \\ \\ T' &= \nu id.\nu n_t.in(c,nr). \\ &\quad out(c, senc((nt,nr),sk)).in(c,msg) \end{aligned}$$

where T and T' are tag processes and n_t is the random nonce and id is the tag identity which is modelled by the tag's secret key. The process T represents a tag with identity id which can execute the protocol an unbounded number of times, while the process T' can execute the protocol at most once. We modelled the above equivalence using ProVerif. As we expected the strong untraceability could not be satisfied for this protocol.

Protocol	Untraceability level						
	Deursen	Bruso	Weak	Low	Mid	High	Strong
Feldhofer	✓	✓	✓	✓	✗	✗	✗

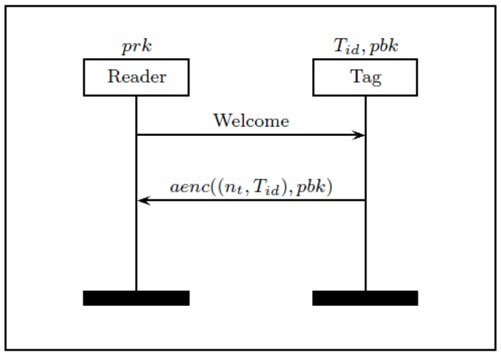


Figure 5.3 Arapinis's Sample Protocol

5.3 Arapinis Toy Protocol

In this section we model the toy protocol used by Arapinis in (Arapinis *et al.*, 2009) in order to analyse its untraceability property. The protocol is depicted as a message sequence chart in Figure 5.3.

5.3.1 Description

Within this sample protocol the reader sends a welcome message to the tag. The tag responds to reader with its identity T_id paired with a nonce n_t, asymmetrically encrypted with public key k. It is worth mentioning here that this sample protocol does not satisfy authentication, since it is a sample program.

5.3.2 The Model in Applied Pi

The protocol is defined with respect to the signature $\Sigma = \{aenc, adec, pk\}$. The cryptographic primitives are modelled as an equational theory. The encryption function is modelled by the equation: $adec(aenc(m, pk(skey)), skey) = m$; where the unary constructor pk constructs a key pair, it takes a private key and returns a public key. $aenc$ and $adec$ are binary functions representing asymmetric key encryption and decryption. Encryption and decryption are in a similar manner to symmetric cryptography with a public/private key pair used instead of a symmetric key. The ProVerif code used to model the protocol is shown in table 5.4

5.3.3 Automatic Verification Results

We modelled different properties of untraceability. We verified all the untraceability definitions but the strong one.

Protocol	Untraceability level						
	Deursen	Bruso	Weak	Low	Mid	High	Strong
Arapinis toy prot.	✓	✓	✓	✓	✓	✓	✗

5.4 KIM Secure and Private Protocol

In this section we study a protocol proposed by Kim et al. (Kim *et al.*, 2007). This protocol aims to protect personal privacy and keep the tag untraceable. First we give an informal definition. The protocol is depicted as a message sequence chart in Figure 5.4. Then, we model the protocol in the applied pi calculus. Lastly, we show that this protocol is not private due to our untraceability definitions.

Table 5.4 Arapinis's sample protocol

```
(* a generic asymmetric encryption function *)
fun pk/1.
fun adec/2.
fun aenc/2.
reduc adec(aenc(m, pk(prk)),prk) = m.


(* Key Distributor Process *)
let Key =
new prk;
out(privcR,prk);
out(pubch,pk(prk)).


(* Tag Process *)
let Tag =
in(pubcT,pbk);
in(c,x);
out(c,aenc((nt,id),pbk)).


(* Reader Process *)
let Reader =
in(privcR,sk);
new x;
out(c,x)
in(c,msg).


(* Main process *)
process new id; ( Key | Reader |
(let pubch = pubcR in let id = id in Tag) )
```

We also demonstrate the man-in-the-middle attack which gives the adversary the knowledge to distinguish between tags. Then we propose a correction on the protocol and show that the new fixed model satisfies upto high untraceability and not Arapinis's, strong untraceability.

5.4.1 Description

This protocol consists of three phases, identification, initial setup and privacy protection phase. In the initial phase both the mobile reader and the tag receive a key k from the server. The protocol starts with the reader challenging the tag with a freshly generated nonce n_r. Upon receiving the request, the tag generates a nonce n_t. Then it computes $h_k(n_r)$ and responds with $ID \oplus n_t$ and $h_k(n_r) \oplus n_t$.

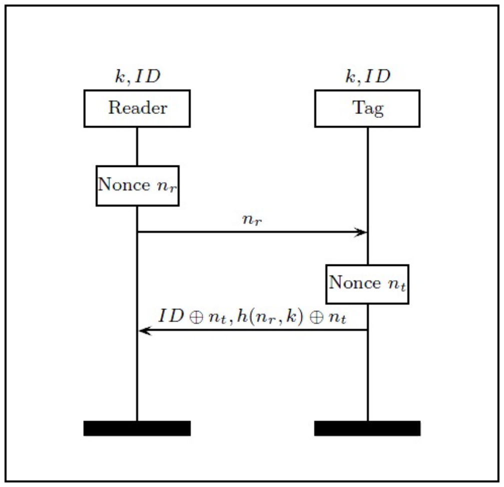


Figure 5.4 The Kim Secure and Private Protocol

5.4.2 The Model in Applied Pi

The algebraic properties of the xor function are not supported by ProVerif. Hence, we modelled a simplified version as: $xor(xor(x, y), y) = x$. However, this abstraction gives less deduction power to the attacker and thus could result in the loss of possible attacks. We use a signature in which $xor(x, y)$ denotes the simplified xor function and h denotes the keyed hash function. The keyed hash function is modelled using a binary function without any equation that represent the one-wayness of the hash function. The ProVerif code used to model the protocol is shown in table 5.5.

Main Process. The main process generates a new session identifier represented by tag's nonce and a tag identifier represented by hash function's key, to instantiate a copy of a tag process executing in parallel with the reader process.

Tag Process. In the tag process, first it obtains its secret key. Then as it is mentioned in the informal description, after receiving the nonce generated by the reader, it sends a message to the reader.

Reader Process. The reader is willing to run the protocol with any other principal. On request from a tag, the reader starts the protocol by selecting a fresh nonce n_r and outputting it on the public channel.

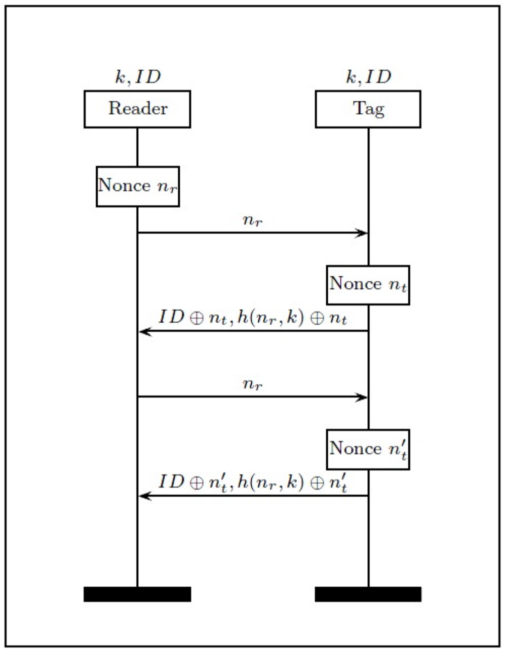
Table 5.5 Kim Protocol ProVerif Code

```
(* a one-way keyed hash function *)
fun h/2.


(* simplified xor function *)
fun xor/2.
equation xor(xor(x,y),y)=x.


(* Tag Process *)
let Tag =
in(privch,k);
in(c,nr);
out(c,(xor(id,nt),xor(h(nr,k),nt))).


(* Reader Process *)
let Reader =
new nr;
out(c,nr)
in(c,msg).


(* Main process *)
process new sk; new nt1; ( Reader |
(let k = sk in let nt = nt1 in Tag) )
```

5.4.3 Attack on untraceability

The authors claimed that their protocol provides untraceability, because the tag never sends the same response twice and it is refreshed by both sides on each session. ProVerif could not prove the untraceability property of this protocol. We show that the protocol is not untraceable by providing an algorithm that gives the adversary a non-negligible advantage of guessing the selected tag.

To attack untraceability, the adversary challenges the tag twice with the same nonce. He can then calculate the xor of the two parts $ID \oplus n_t$ and $h_k(n_r) \oplus n_t$ of the responses. This equation does not depend on n_t and it always gives the result of $ID \oplus h(n_r, k)$. Therefore the adversary twice obtains $ID \oplus h(n_r, k)$, if and only if it was twice the same tag that he challenged. The attack is depicted in Figure 5.5.



k, ID
Reader
k, ID
Tag
Nonce n_r
n_r
Nonce n_t
$ID \oplus n_t, h(n_r, k) \oplus n_t$
n_r
Nonce n'_t
$ID \oplus n'_t, h(n_r, k) \oplus n'_t$

Figure 5.5 Man-in-the-middle Attack on Kim et al. Protocol

5.4.4 Fixing the Protocol

The previously described protocol problem is based on the fact that the adversary can guess if she is communicating with the same tag or not, by receiving the same messages after sending the same nonce twice. The solution for this problems should not require heavy additional overload for the tag.

Therefore it is feasible to change the message $h_k(n_r) \oplus n_t$ to $h_k(n_r) \oplus h_k(n_t)$. As you could see, we did not add additional cryptographic capabilities in the tag, but rather use what already is available.

5.4.5 Automatic Verification Results

We modelled the fixed system in applied pi calculus, first we verified the confidentiality of the secret values like ID, k and n_t. After that we went through the mutual authentication properties. ProVerif verified both secrecy and authentication properties. This part is done in the similar was as the first case study by adding begin and end events, in both tag and reader processes. Then by querying the correspondence property mutual authentication property could be concluded.

After satisfying the correctness of the protocol, we analysed the untraceability properties. Finally, it is observed that, all untraceability properties, but the strong, are successfully verified on fixed model of this protocol.

Protocol	Untraceability level						
	Deursen	Bruso	Weak	Low	Mid	High	Strong
Kim	✗	✗	✗	✗	✗	✗	✗
Kim modified	✓	✓	✓	✓	✓	✓	✗

5.5 OSK protocol

OSK, the protocol presented by Ohkubo et al. (Ohkubo *et al.*, 2003), belongs to the class of single step protocols. In this class of protocols the tag is activated by the reader without receiving any message. Then the tag generate the message due to its current state, which mostly contains a fresh nonce and sends it to the reader. This is the only message sent within this protocol. After that the tag updates its current state and the session ends. This class of protocols mostly used in the supply chain systems. As it is mentioned in the chapter three, there is a security scheme for tags based on one-way hash functions. It only requires implementing a hash function on the tag and the key management is done on the back-end. This makes the tags light and cheap for mass production.

The problem here, is raised because the tags may still function as object identifiers while in the locked state by using the ID for database lookups. Since the ID acts as identifier, it allows the adversary to take advantage of the tag functionality and track individuals. This shows the importance of verifying the untraceability for this kind of protocols.

5.5.1 Description

This protocol is defined with respect to the signature $\Sigma = \{f, g\}$ where both f and g are unary one-way hash functions. It is depicted in Figure 5.6.

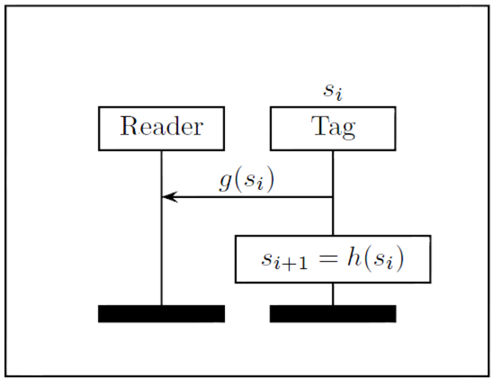


Figure 5.6 The OSK protocol

It assumes that a tag can compute two distinct one-way hash functions f and g. Each tag initially stores a secret id s which is shared with the backend database. The hash function h

is used to update the secret at each run of the protocol while g is used to encrypt the output; the tag sends $g(s_i)$, where s_i is its current identifier and then updates its secret $s_{i+1} = h(s_i)$.

5.5.2 The model in Applied Pi

In this protocol, similar to all other single step protocols, the readers are completely passive, in contrast with above presented protocols. So, in this protocol for an RFID system we could set Reader as a null process. The tag also does not need any interaction with other participants to update its secret, and it is modelled with a simple hash function. Therefore we have the following simple RFID system:

Table 5.6 OSK Protocol ProVerif Code

```
(* two one-way hash functions *)
fun h/1.
fun g/1.


(* Tag Process *)
let Tag =
out(c,g(s));
let s = h(s) in 0.


(* Reader Process *)
let Reader =
in(c,msg).


(* Main process *)
process ( Reader | Tag )
```

5.5.3 Automatic Verification Result

ProVerif proved that all untraceability properties are satisfied by OSK protocol. Actually we modelled different single step protocols and we could not find any that does not satisfy untraceability properties. We find single step protocols the only group which could achieve Arapinis Strong untraceability, and this is due to the fact that these protocols only send one message by the public channel and each time the secret changes, which makes it impossible for the attacker to get any knowledge of messages or agents secrets.

Protocol	Untraceability level						
	Deursen	Bruso	Weak	Low	Mid	High	Strong
OSK	✓	✓	✓	✓	✓	✓	✓

5.6 Results and Discussions

Table 5.7 depicts the results of our case studies in brief.

Table 5.7 Untraceability results of case studies

Protocol	Untraceability level						
	Deursen	Bruso	Weak	Low	Mid	High	Strong
Feldhofer	✓	✓	✓	✓	✗	✗	✗
Arapinis toy prot.	✓	✓	✓	✓	✓	✓	✗
Kim	✗	✗	✗	✗	✗	✗	✗
Kim modified	✓	✓	✓	✓	✓	✓	✗
OSK	✓	✓	✓	✓	✓	✓	✓

- As shown in the table 5.7, weak and low untraceability indicate the same outputs for all studied protocols, corresponding to the discussion held in section 4.4.
- Although we showed the difference between mid and high untraceability in an example, we could not find any protocol, among the existing protocols, which satisfies mid untraceability but not high untraceability. Consequently in the above results, it is seen that any protocol, satisfies mid untraceability, also satisfies high untraceability.
- The result reveals that, the OSK is the only protocol which satisfies strong untraceability. Not satisfying the strong untraceability by other protocols than OSK, is due to the fact that, the adversary observes that there are two tags which executed multiple sessions each. Since it violates the case that each tag executes itself at most once the actual protocols can be distinguished from the ideal version. This does not let the ProVerif to satisfy strong untraceability, however, this information does not allow the tag to be really traced.
- ProVerif showed that, the untraceability property is not satisfied by the Kim's protocol, and the attack algorithm is provided. We modified the protocol and demonstrated that the modified version meets all our untraceability levels.

Table 5.8 Comparing time to verify high untraceability

Protocol	Time (ms)	
	Bruso UNT	High UNT
OSK	4	4
Kim modified version	15476	7867

ProVerif also enables us to measure the time that takes to get the results. The results showed that our definition works even more efficient in the manner of time among others.

Although the time is not an important factor comparing to the properties. As we measured, in simple protocols the difference is not too much, but in more complicated protocols you could see an impressive difference. The table 5.8 depicts a comparison between our high untraceability with a weaker model proposed by Bruso, which you could see it works faster. We could not compare our high definition with Arapinis strong definition since we could not find a suitable protocol which verifies both properties.

CHAPTER 6

CONCLUSION

Technically, in this thesis, we improved the capabilities of the formal verification of RFID identification protocols, by defining explicit definition of security properties, presenting a procedure for the evaluation of security properties. This final chapter presents a brief conclusion of our work and the possibilities for the future work.

6.1 Review of the Research

This work provided the design of a secure and a private identification protocols, improving the evaluation of existing schemes by using proper verification. We classified different strengths of untraceability, based on three presented game types. Subsequently, a formal methodology is suggested to verify such property.

After collecting all the privacy games in the literature, it is concluded that the difference between definitions, is derived from the indistinguishability games used in the definition of untraceability. Then we divided this property into three levels related to the type of games. It is assumed as the first classification of untraceability, where it covers all previous definitions. In brief we divided untraceability into following levels:

- Low Untraceability as ensuring that the adversary can not be able to distinguish whether two given sessions in any trace initiated from the same tag or two different tags.
- Mid Untraceability as ensuring that for a given session the adversary must not be able to infer that there exist another session initiated from the same tag as the given session is initiated from.
- High Untraceability as ensuring that the adversary must not be able to infer whether there exist two sessions in any trace that initiated from the same tag.

We compared our levels of untraceability with Arapinis's weak and strong definition. We depicted that the weak untraceability and our low untraceability are coincides. We also showed that the strong definition can only be used for a small group of protocols, and because of some assumptions it is impossible to satisfy this property for most protocols. We also transformed the trace-based model of untraceability, presented by Deursen, into processes and demonstrated that it is weaker than our notion of low untraceability. Finally we proved that Bruso's definition of untraceability implies our low untraceability.

In order to specification and verification, we have formalized an RFID protocol model and three untraceability levels, using applied pi calculus. It is a process calculus which is able to model security protocols by having a rich term algebra, value passing, function symbols and a equational theory to model the cryptographic operations. Applied pi calculus enables us to model both reachability, correspondence and observational equivalence properties. We formalized untraceability as some kind of observational equivalence.

We have demonstrated our results in some case study protocols. We modelled four different protocols in the literature, where three of them was chosen among the previously analysed protocols and one of them has not been automatically analysed before. We demonstrated that, the protocol proposed by Feldhofer, which is assumed to be untraceable can only satisfy our notion of low untraceability. We modelled another protocol which as authors claimed, it supposed to be secure and private. After analysing this protocol, we got a trace which resulted in tracing a tag by the adversary. Then by providing the attack algorithm on untraceability property, we proposed a fix for this protocol. We have used the existing cryptographic capabilities of the tag in order to not any additional overhead to the tag. We showed that the modified protocol satisfies all our notions of untraceability. Finally we modelled a protocol belonging to the group of single step protocols, where we found the only group that satisfies all of the mentioned untraceability levels including Arapinis strong definition.

Extensive case studies show the effectiveness and efficiency of our taxonomy in verifying untraceability of RFID protocols. The results also showed that our definitions works more efficient in the manner of time, among the others. Moreover, our approach and framework are generic and could be adapted into different systems.

The principle limitation of our approach is derived from the algebraic properties of logical operations used in some RFID protocols, which do not allow the verifier tools to model such systems. There are some investigation to illustrate how to convert these sort of operators in a language, processed by some tools. However, they can solely verify authentication properties and can not be used in analysis of the observational equivalence.

6.2 Possible Improvements and Future Work

Concerning future possible research related to verification of security properties, it seems promising to model other privacy properties like forward or backward security using the same model presented for untraceability.

Another potential area for future work is enriching ProVerif to consider logical operators such as Exclusive-Or, or to develop an interface to convert the protocol including logical operators into applied pi. It is worth mentioning that there is an add-on for proverif which enables us to verify secrecy and correspondence properties of protocols with logical operators but it is not able to analyse the observational equivalence. Therefore since logical operators acts the main role in light identification protocols, it will be very useful to automatize the verification procedure.

Although untraceability properties are mostly used in the context of RFID systems, they are issues for any protocol, which can be used with a mobile device, so it is possible to extend it to other protocols like VANET and etc. So there is also a possibility of study of untraceability in these environments.

REFERENCES

ABADI, M. et FOURNET, C. (2001). Mobile values, new names, and secure communication. *ACM SIGPLAN Notices*. ACM, vol. 36, 104–115.

ABADI, M. et GORDON, A. (1997). A calculus for cryptographic protocols: The spi calculus. *Proceedings of the 4th ACM Conference on Computer and Communications Security*. ACM, 36–47.

AHAMAD, S. S., UDGATA, S. K. et SASTRY, V. (2012). A new mobile payment system with formal verification. *International Journal of Internet Technology and Secured Transactions*, 4, 71–103.

ARAPINIS, M., CHOTHIA, T., RITTER, E. et RYAN, M. (2009). Untraceability in the applied pi-calculus. *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*. IEEE, 1–6.

ARMANDO, A., COMPAGNA, L. et GANTY, P. (2003). Sat-based model-checking of security protocols using planning graph analysis. *FME 2003: Formal Methods*, 875–893.

AVOINE, G. (2005). Adversarial model for radio frequency identification. *Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC*. Citeseer.

AVOINE, G. et OECHSLIN, P. (2005). Rfid traceability: A multilayer problem. *Financial Cryptography and Data Security*, 125–140.

BASSIL, R., EL-BEAINO, W., ITANI, W., KAYSSI, A. et CHEHAB, A. (2012). Pumap: A puf-based ultra-lightweight mutual-authentication rfid protocol. *International Journal of RFID Security and Cryptography*, 1, 58–66.

BIEBER, P. (1990). A logic of communication in hostile environment. *Computer Security Foundations Workshop III, 1990. Proceedings*. IEEE, 14–22.

BLANCHET, B. *ET AL.* (2001). An efficient cryptographic protocol verifier based on prolog rules. *14th IEEE Computer Security Foundations Workshop (CSFW-14)*. vol. 96.

BREU, D. F. (2011). Rfid security and privacy. *Future Internet*, 81.

BRUSÓ, M., CHATZIKOKOLAKIS, K. et DEN HARTOG, J. (2010). Formal verification of privacy for rfid systems. *2010 23rd IEEE Computer Security Foundations Symposium*. IEEE, 75–88.

BURROWS, M., ABAD, M. et NEEDHAM, R. (1989). A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 426, 233.

CAI, S., DENG, R., LI, Y. et ZHAO, Y. (2012). A new framework for privacy of rfid path authentication. *Applied Cryptography and Network Security*. Springer, 473–488.

CANARD, S., COISEL, I., ETROG, J. et GIRAULT, M. (2010). Privacy-preserving rfid systems: Model and constructions. *Eprint: IACR*, 21, 22.

CHATMON, C., VAN LE, T. et BURMESTER, M. (2006). Secure anonymous rfid authentication protocols. *Florida State University, Department of Computer Science, Tech. Rep.*

CHOTHIA, T. et SMIRNOV, V. (2010). A traceability attack against e-passports. *Financial Cryptography and Data Security*, 20–34.

CLARKE, E. (1997). Model checking. *Foundations of software technology and theoretical computer science*. Springer, 54–56.

CREMERS, C. et MAUW, S. (2005). Operational semantics of security protocols. *Scenarios: Models, Transformations and Tools*, 576–576.

DATTA, A., DEREK, A., MITCHELL, J. C. et PAVLOVIC, D. (2005). A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13, 423–482.

DENKER, G., MESEGUER, J. et TALCOTT, C. (1998). Protocol specification and analysis in maude. *Proc. of Workshop on Formal Methods and Security Protocols*. Indianapolis, vol. 25.

DIMITRIOU, T. (2006). A secure and efficient rfid protocol that could make big brother (partially) obsolete. *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*. IEEE, 6–pp.

DOLEV, D. et YAO, A. (1983). On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29, 198–208.

FELDHOFER, M., DOMINIKUS, S. et WOLKERSTORFER, J. (2004). Strong authentication for rfid systems using the aes algorithm. *Cryptographic Hardware and Embedded Systems-CHES 2004*, 85–140.

FOSSO, S. (2012). Rfid-enabled healthcare applications, issues and benefits: An archival analysis. *Journal of Medical Systems*, 1–6.

GARCIA, F., HASUO, I., PIETERS, W. et VAN ROSSUM, P. (2005). Provable anonymity. *Proceedings of the 2005 ACM workshop on Formal methods in security engineering*. ACM, 63–72.

HA, J., MOON, S., NIETO, J. et BOYD, C. (2007). Low-cost and strong-security rfid authentication protocol. *Emerging Directions in Embedded and Ubiquitous Computing*, 795–807.

HENRICI, D. et MÜLLER, P. (2004). Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers.

JIA, H. et WEN, J. (2012). A novel rfid authentication protocol with ownership transfer. *Advances in Automation and Robotics, Vol. 1*, 599–606.

JUELS, A. (2006). Rfid security and privacy: A research survey. *Selected Areas in Communications, IEEE Journal on*, 24, 381–394.

JUELS, A. et WEIS, S. A. (2007). Defining strong privacy for rfid. *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*. 342 –347.

KHASHKHASHI MOGHADDAM, S., NAKHAEIZADEH, G. et KAKHKI, E. (2012). An improved data warehouse model for rfid data in supply chain. *Intelligent Information and Database Systems*, 488–497.

KIM, I., CHOI, E. et LEE, D. (2007). Secure mobile rfid system against privacy and security problems. *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2007. SECPerU 2007. Third International Workshop on*. IEEE, 67–72.

KUMAR, V. N. et SRINIVASAN, B. (2012). Design and implementation of epassport scheme using cryptographic algorithm along with multimodal biometrics technology. *International Journal*.

KUSHAL, K., MUTTANNA KADAL, H. et CHETAN, S. (2012). Design and implementation of a rfid based prototype smart library (salary) system using wireless sensor networks. *Advances in Computer Science, Engineering & Applications*, 499–505.

KÜSTERS, R. et TRUDERUNG, T. (2011). Reducing protocol analysis with xor to the xor-free case in the horn theory based approach. *Journal of Automated Reasoning*, 46, 325–352.

LEE, S., ASANO, T. et KIM, K. (2006). Rfid mutual authentication scheme based on synchronized secret information. *Symposium on Cryptography and Information Security*.

LEE, Y.-C. (2012). Two ultralightweight authentication protocols for low-cost rfid tags. *Applied Mathematics and Information Sciences*, 6, 425–431.

LOWE, G. (1996). Breaking and fixing the needham-schroeder public-key protocol using fdr. *Tools and Algorithms for the Construction and Analysis of Systems*, 147–166.

MARCELLA, A. et STUCKI, C. (2003). *Privacy handbook: guidelines, exposures, policy implementation, and international issues*. Wiley.

MEADOWS, C. (1996). The nrl protocol analyzer: An overview. *The Journal of Logic Programming*, 26, 113–131.

MEADOWS, C. (2003). Formal methods for cryptographic protocol analysis: Emerging issues and trends. *Selected Areas in Communications, IEEE Journal on*, 21, 44–54.

MEADOWS, C. et MEADOWS, C. (1995). Formal verification of cryptographic protocols: A survey. *Advances in Cryptology—ASIACRYPT'94*, 133–150.

MENEZES, A., VAN OORSCHOT, P. et VANSTONE, S. (1997). *Handbook of applied cryptography*. CRC.

MILLEN, J., CLARK, S. et FREEDMAN, S. (1987). The interrogator: Protocol secuity analysis. *Software Engineering, IEEE Transactions on*, 274–288.

MILNER, R. (1991). The polyadic n-calculus: a tutorial. *Logic and algebra of specification*, 94.

MILNER, R. (1999). *Communicating and mobile systems: the pi-calculus*. Cambridge Univ Pr.

MITCHELL, J., MITCHELL, M. et STERN, U. (1997). Automated analysis of cryptographic protocols using murφ. *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*. IEEE, 141–151.

NAOR, M. et YUNG, M. (1990). Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 427–437.

NEEDHAM, R. et SCHROEDER, M. (1978). Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21, 993–999.

OHKUBO, M., SUZUKI, K., KINOSHITA, S. *ET AL.* (2003). Cryptographic approach to privacy-friendly tags. *RFID Privacy Workshop*. Citeseer, vol. 82.

OUAFI, K. et PHAN, R. (2008). Privacy of recent rfid authentication protocols. *Information Security Practice and Experience*, 263–277.

PAULSON, L. (1994). *Isabelle: A generic theorem prover*, vol. 828. Springer.

PRADHAN, D. et HARRIS, I. (2009). *Practical design verification*. Cambridge University Press.

QI, S., LU, L., LI, Z. et LI, M. (2012). Best: A bidirectional efficiency-privacy transferable authentication protocol for rfid-enabled supply chain. *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*. IEEE, 424–431.

RACKOFF, C. et SIMON, D. (1992). Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology—CRYPTO'91*. Springer, 433–444.

SARAC, A., ABSI, N. et DAUZÈRE-PÉRÈS, S. (2010). A literature review on the impact of rfid technologies on supply chain management. *International Journal of Production Economics*, 128, 77–95.

SIMINICEANU, R. I. et CIARDO, G. (2012). Symbolic model checking for avionics. *Formal Methods for Industrial Critical Systems: A Survey of Applications*, 85.

SOUYRIS, J., WIELS, V., DELMAS, D. et DELSENY, H. (2009). Formal verification of avionics software products. *FM 2009: Formal Methods*, Springer. 532–546.

SUN, D.-Z. et ZHONG, J.-D. (2012). A hash-based rfid security protocol for strong privacy protection. *Consumer Electronics, IEEE Transactions on*, 58, 1246–1252.

SYVERSON, P. (1990). Formal semantics for logics of cryptographic protocols. *Computer Security Foundations Workshop III, 1990. Proceedings.* IEEE, 32–41.

TSUDIK, G. (2006). Ya-trap: Yet another trivial rfid authentication protocol.

VAN DEURSEN, T., MAUW, S. et RADOMIROVIĆ, S. (2008). Untraceability of rfid protocols. *Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, 1–15.

VAN DEURSEN, T. et RADOMIROVIC, S. (2009). Security of rfid protocols-a case study. *Electronic Notes in Theoretical Computer Science*, 244, 41–52.

VAUDENAY, S. (2007). On privacy models for rfid. *Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security.* Springer-Verlag, 68–87.

WAMBA, S. (2012). Achieving supply chain integration using rfid technology: The case of emerging intelligent b-to-b e-commerce processes in a living laboratory. *Business Process Management Journal*, 18, 58–81.

WANG, M., TANG, Y., SHI, F. et PAN, J. (2012). An effective rfid authentication protocol. *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on.* IEEE, 141–144.

XIANG, P. (2012). Research and implementation of access control system based on rfid and fnn-face recognition. *Intelligent System Design and Engineering Application (ISDEA).* IEEE, 716–719.