Eindhoven University of Technology
Department of Mathematics and Computer Science

Master's Thesis

Secure Ownership and Ownership Transfer in RFID Systems

by

Pim Vullers

Supervisors:

Dr. Jerry den Hartog (Eindhoven University of Technology)
Prof. Dr. Sjouke Mauw (University of Luxembourg)

Tutors:

Dr. Saša Radomirović (University of Luxembourg)
Ir. Ton van Deursen (University of Luxembourg)

Luxembourg, July 2009

Nothing that you have not given away will ever be really yours.
– C.S. Lewis

Abstract

Radio frequency identification (RFID) is an upcoming technology with a large potential, but it also introduces a number of threats. Ownership transfer has been introduced to mitigate the risks which are caused by changes in ownership of RFID tags.

We present formal definitions for ownership and ownership transfer in RFID systems and their secure variants. These definitions can be used to verify the correctness of ownership transfer protocols. We apply our definitions to existing RFID protocols, exhibiting attacks on desynchronisation resistance, secure ownership, exclusive ownership and secure transfer.

We design an ownership transfer protocol using public-key cryptography which satisfies all our requirements. We also design a second protocol using shared-key cryptography which has led to an interesting observation concerning exclusivity of ownership.

Preface

Silent gratitude isn't much use to anyone.
– G.B. Stern

This Master's Thesis concludes my study in Information Security Technology at the Department of Mathematics and Computer Science at the Eindhoven University of Technology. The research for this thesis, as well as the writing, has been done at the University of Luxembourg. This research focuses on the security aspects of ownership and ownership transfer in RFID systems.

I am grateful to Sjouke Mauw for providing me the opportunity to carry out my Master project at the University of Luxembourg, as well as for his supervision of this project. Special thanks go to my tutors Saša Radomirović and Ton van Deursen for the many interesting discussions and their help and guidance during the execution of this project. The results from this project are published in [DMRV09]. This would not have been accepted at ESORICS without their effort as co-authors of this paper.

Furthermore, I would like to thank the other members of the SaToSS group for their helpful comments and the pleasant working environment.

A word of thanks towards Sandro Etalle and Jerry den Hartog for their support from the Eindhoven University of Technology. Also a word of thanks for the members of my Assessment Committee for the time they spent reading and judging my work.

Finally, I would like to thank my girlfriend and family for their support during this period of my life in Luxembourg.

Pim Vullers
Luxembourg, July 2009

Contents

Chapter 1

Introduction

With the possible exception of the equator, everything begins somewhere.
– C.S. Lewis

Radio frequency identification (RFID) is expected to become a key technology in supply chain management, because it has a large potential to save costs. Two of the cost-saving advantages of this technology are the improved efficiency of inventory tracking [GJP05] and the reduction of counterfeit products [STF05]. The former is caused by the contactless nature of this technology which requires no line of sight between the RFID reader and the RFID tag attached to a product. The latter is because RFID tags can store and process information as well as execute simple communication protocols. Because of these advantages and their small size, RFID tags are expected to replace traditional identification methods such as bar codes.

This technology is furthermore of interest to retailers since in addition to tracking and management benefits, the customer's checkout time can be significantly reduced. This is possible since multiple RFID tags can be read almost simultaneously which allows for fast processing of batches, in this case the products in a shopping cart. Another benefit of using RFID tags instead of bar codes is the possibility to store additional information on a tag. This means that, for example, the expiration date can be stored such that a smart fridge can keep track of it [RFI03]. These additional uses, besides identification, make it interesting for consumers to have products tagged.

However, the widespread use of RFID technology also introduces security and privacy issues, ranging from corporate espionage threats to tracking of individuals [GJP05]. Simple solutions such as killing a tag are not desired since they will render the tag useless. Hence we will have to look at other solutions to solve these problems.

When tagged products change owners the situation becomes even more difficult. One would like that the ownership of a tag changes along with the ownership of the product the tag is attached to. This involves, for example, exchanging the key material, used to achieve authentication or untraceability, between the previous and the new owner. Ownership transfer for RFID tags has been introduced by Molnar et al. [MSW05] to securely transfer a tagged item from one owner to the other.

1.1 Motivation

The ownership transfer approach has been adopted by Osaka et al. [OTYT06] who are among the first to propose a two-party ownership transfer protocol, in contrast to the solution of Molnar et al. which requires a trusted third party. Lei and Cao [LC07] are the first to report flaws in the protocol by Osaka et al. In their work they propose a protocol which solves these vulnerabilities. They do, however, not meet their security requirements as shown in Section 4.3.4 of this thesis. Other solutions which attempt to repair the flaws in the protocol by Osaka et al. have been proposed by Jäppinen and Hämäläinen [JH08], and Yoon and Yoo [YY08]. These solutions are also not secure, as shown in Sections 3.5.3 and 4.3.3 respectively.

Similar to the protocol proposed by Li and Ding [LD07], which has been reported broken by Van Deursen and Radomirović [DR08b], the proposed solutions in literature do not provide proper proofs of their security requirements. This is mainly caused by the fact that there is no formal definition of security on which such proofs can be based. Furthermore, ownership, as well as ownership transfer, have not yet been formally defined.

1.2 Goal

In order to design a secure ownership transfer protocol we need to determine what it means for ownership transfer to be secure. This leads to the following goal for this master project.

> *Define security for ownership transfer and design and verify a secure ownership transfer protocol using this definition.*

1.3 Contributions

We propose a number of definitions which, together, allow the verification of security for ownership transfer. First, we provide a definition for ownership. This definition serves as a basis for secure ownership, exclusive ownership, and desynchronisation resistance. As well as for all other definitions in this thesis.

Next, we describe the notion of ownership transfer, and its security requirements secure and exclusive transfer. Together with the security requirements for ownership these definitions provide a framework to verify ownership related properties. This framework, as described in Chapters 3 and 4 is published in [DMRV09]. This paper has been written in cooperation with Ton van Deursen, Sjouke Mauw and Saša Radomirović.

Finally, we design a protocol which achieves secure exclusive ownership transfer. This protocol is refined to use shared-key cryptography instead of public-key cryptography. This refinement achieves secure ownership transfer, but it does not satisfy our exclusivity requirements. It has, however, resulted in an interesting observation concerning these requirements.

Chapter 2

Background

The answers you get from literature depend on the questions you pose.
– M. Atwood

Before defining a verification framework or designing any protocols, we introduce the context of this thesis. First, we discuss a number of scenarios for ownership transfer. Next, we consider related work which mainly consists of proposed ownership transfer protocols. Finally, an introduction to the formal model, used to formulate our definitions, is given.

2.1 Scenarios

The basic scenario for ownership transfer envisioned in literature concerns a longer lifetime of RFID tags in which ownership of the tags changes. A concrete ownership transfer scenario, which concerns tagged products in a supply chain, is given by Li and Ding [LD07]. As products flow through a supply chain, their ownership is transferred from one partner to the next. This transfer of ownership can be extended to the RFID tags attached to these products.

Possible threats in this scenario are corporate espionage and disruption of the production process. To prevent the former only the partner currently owning the product should be able to read the tag. In some cases even untraceability of tags [DMR08] is required to provide protection against tracing of the internal product flow. To prevent against the latter the RFID system should protect against denial-of-service attacks. Hence it should at least be desynchronisation resistant (Section 3.6.1).

Once a product has reached the end of a supply chain it will end up at a retail store. The tags on the products allow easy inventory management and fast product registration at the checkout, where they are transferred to the customer. Engberg et al. [EHJ04], Fouladgar and Afifi [FA07], as well as Dimitriou [Dim08] mention the situation that a tag can also be used at after sales services to retrieve additional information concerning the product. The tags might, however, also contain information which can be used by consumers, such as an expiration date which can be used by smart fridges [RFI03].

This scenario differs from the previous scenario since it now involves consumers as the new owners instead of businesses. Garfinkel et al. [GJP05] describe a number of privacy threats which arise from the fact that tags can easily be

associated with a person's identity. Hence for a consumer it is important that the tags are untraceable, while they should also be usable.

A related scenario, in which we consider a large supermarket with various small shops, uses tags to record which products have been paid for, and which not. When a customer pays for a product, for example in one of the small shops, ownership will be transferred to the customer. At the final checkout the ownership of the tags is verified. For those which still belong to the supermarket, the customer has to pay.

For this scenario it is important that after a transfer the supermarket actually loses ownership to prevent that a customer has to pay twice. Furthermore, to prevent stealing, it should not be possible that the supermarket loses ownership while the product has not been paid for.

Finally, another area in which ownership transfer might play a role is parcel delivery. In this scenario ownership is transferred to the messenger when the parcel is picked up, and transferred to the recipient when delivered. In combination with non-repudiation this offers the opportunity for the messenger to prove to the sender that the parcel has actually been delivered at the correct recipient.

2.2 Related Work

Work on ownership transfer in RFID systems has thus far mostly focused on designing ownership transfer protocols, but not on their security requirements. A notable exception is the work by Song [Son08]. This work provides a first survey of security requirements related to ownership transfer. Besides a list of basic RFID security requirements, new and old owner privacy as well as authorisation recovery are mentioned. Song also proposes a set of protocols for secure ownership transfer which has been designed to meet the requirements which came up from the survey. One of these protocols is based on an authentication protocol by Song and Mitchell [SM08]. This earlier work, however, suffers from a number of flaws as described in Section 3.6.2 and by Van Deursen and Radomirović [DR08a]. These flaws are also present in the transfer protocol by Song [Son08]. In Section B.1 we describe an additional flaw which has been discovered during our research.

The first treatment of ownership transfer in RFID systems is due to Molnar et al. [MSW05]. They describe a protocol that relies on a trusted centre. In their protocol tags respond with a pseudonym instead of their identity. Readers send these tag pseudonyms to the centre requesting the real identity of a tag. If the reader is the owner of the tag it receives the identity, otherwise the request is denied. Owners of tags can request the trusted centre to transfer the ownership of a tag to a new owner. The trusted centre subsequently refuses identity requests from the old owner, and accepts them from the new owner.

A trusted party is also used by the protocol proposed by Saito et al. [SIS05]. In this case, the trusted party shares a key with the tag which is used to update the owner's key. Hence an ownership transfer consists of a request to the trusted party to encrypt the new owner's key for the tag.

Osaka et al. [OTYT06] are among the first to propose a two-party ownership transfer protocol. Their protocol consists of three phases in which first the tag is updated with a fresh key. This key is transferred in a secure fashion to the

new owner who then updates the tag again using its own key. This structure is used by almost all subsequent solutions.

Lei and Cao [LC07] are the first to report flaws in the protocol by Osaka et al. In their work they also propose a protocol which solves these vulnerabilities. They do, however, not meet their security requirements as shown in Section 4.3.4 of this thesis. Other solutions which attempt to repair the flaws in the protocol by Osaka et al. have been proposed by Jäppinen and Hämäläinen [JH08], and Yoon and Yoo [YY08]. Again their solutions are also not secure. We discuss these protocols in detail, including descriptions of attacks, in Sections 3.5.3 and 4.3.3 respectively. Like the protocol proposed by Li and Ding [LD07], which has been reported broken by Van Deursen and Radomirović [DR08b], they do not provide proper proofs of their security requirements.

Lim and Kwon [LK06] propose a protocol which, compared to other solutions, uses a more computationally intensive mutual authentication method based on key chains. Fouladgar and Afifi [FA07] propose an ownership transfer protocol with two implementations. Their hash-based implementation, which is flawed, is discussed in Section B.2. The other implementation, in contrast to the earlier mentioned solutions based on hashing, is based on symmetric encryption. Another solution based on symmetric encryption has been proposed by Koralalage et al. [KRM+07]. Finally, one of the most recent protocols in this area is due to Dimitriou [Dim08]. Its distinguishing feature is that it enables the owner of a tag to revert the tag to its original state. This is useful for after-sales services, since it makes it possible for the tag's new owner to let a retailer recognise a sold tag.

The research concerning ownership transfer has thus far resulted in a lot of protocols which claim to achieve secure ownership transfer. However, a large number of these protocols do not match their security claims as shown throughout this thesis. This can be accounted to the fact that only informal descriptions of the security requirements are available. There are no formal definitions which can be used to actually prove the security of these protocols.

2.3 Stateful Security Protocols

In this section we introduce basic notation and definitions concerning security protocols. Rather than providing a full description of security protocol syntax and semantics, we only present the essentials needed for defining and analysing ownership and related notions. A more extensive description can be found in Appendix A. The model presented, which has been developed by Van Deursen and included in [DMRV09], is based on the model for stateless protocols by Cremers and Mauw [CM05]. Their model has been extended by adding support for stateful protocols. While stateless protocols start in the same state for every execution, stateful protocols may use information from previous and parallel protocol executions.

A *protocol* is defined as a map from an n-tuple of distinct *roles* to an n-tuple of *role specifications*. A role specification defines the behaviour of an *honest agent* executing the role. Typical roles in an RFID system are the reader and tag roles to be executed by actual RFID readers and RFID tags. A particular execution of a protocol role by an agent is called a *run*.

The specification consists of a composition of events and the declaration

of all nonces and variables appearing in the composition. An *event* is either the sending or the receiving of a message and both can be accompanied by assignments to variables. The receiving of messages is referred to as a *read* event. Inspired by Ryan et al. [RSG+00], we use *signals* to indicate that a certain point in the protocol has been reached.

The exchanged messages between roles consist of *terms*. These terms are built from basic terms such as nonces, constants, and agent names. Complex terms can be constructed using functions such as pairing (denoted by $(\_,\_)$), encryption ($\{\_\}_\_$), hashing ($h(\_)$), and exclusive or ($\_ \oplus \_$). When an agent executes a role, nonces are freshly generated and variables receive their actual values through read events and assignments. We separate two kinds of variables. *Local variables* model the stateless part of protocols. Their values are assigned through read events and they are reassigned every run. Once assigned, their value does not change. The stateful part of protocols is modelled by *global variables*. They receive their value through explicit assignments and their values are maintained across different runs.

We study the possible behaviour of a system in which a collection of agents executes a set of protocols Π through so-called *traces*, denoted by $\mathsf{traces}(\Pi)$. Informally, a trace is a list of events occurring in the interleaved execution of protocol runs. The precise construction of traces is dictated by the semantic rules of the system (given in Appendix A). Thus, formally, a valid derivation $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} s_n$ of system states $s_0 \ldots s_n$ is provided by the events $t_0 \ldots t_{n-1}$ of a trace $t = t_0 \ldots t_{n-1}$. In the following we use $\Sigma(t)$ to denote the states $s_0 \ldots s_n$ of this derivation and $|t|$ to denote the length of the trace.

A system state $\langle A, G, SB, RB, I \rangle$ in our model consists of several parts. First of all, the set A is used to record all active runs. For each run we store an identifier, the name of the executing agent, the list of events that still have to be executed in the run, and the local variable assignments. The list of events can be used to derive the execution state of a run. A run r has been successfully completed in state s, denoted by $\mathsf{success}(r, s)$, if its event list is empty. Otherwise the run is still active or it has been terminated unsuccessfully.

Second, the global knowledge of the agents G is stored in the system state to preserve this knowledge across runs. Besides the agent knowledge the knowledge of the intruder I is part of the state. Finally, we consider communication to be asynchronous. Messages sent by agents are placed in the send buffer SB. Similarly, agents read messages from the read buffer RB.

We assume that a standard Dolev-Yao intruder [DY83] controls the network, in our case the communication buffers. The intruder delivers a sent message by moving it from the send buffer to the read buffer. He eavesdrops on messages by adding them to his knowledge. The intruder can construct any message from his knowledge and place it in the read buffer, thus faking sent messages. If the intruder does not move a message from SB to RB the message does not get delivered. This allows the intruder to delay or completely block messages. Finally, a message can be modified by faking a message and blocking the original one. As usual in Dolev-Yao intruder models, the intruder is bound by the assumption of perfect cryptography. This means that he cannot reverse hash functions and that he is not able to learn the contents of an encrypted term, unless he knows the decryption key. We assume that there is one agent E which is under full control of the intruder.

2.4 Message Sequence Charts

Message sequence charts [RGG96] are used to provide a graphical representation of protocol specifications. Every message sequence chart shows the role names, framed, near the top of the chart. Above a role name, the role's secret terms and persistent knowledge are shown. Actions, such as nonce generation, computation, verification of terms, 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. Finally, signals emitted by a role are indicated by a labelled dot.

Chapter 3

Ownership

"It is mine, I tell you. My own. My precious. Yes, my precious."
– Bilbo Baggins in The Fellowship of the Ring by J.R.R. Tolkien

We consider two views on tag ownership. The first view, which we call the system view, stems from the ability to perform a given action on the tag in a predefined manner. Ownership of a tag can, for instance, be defined as an agent's ability to inspect the tag's ID.

The second view is called the agent's view. In this view we assume that an agent records, in some local data structure, of which tags he believes to be the owner. A correspondence relation between these two views can then be considered as a security requirement.

As an application of these definitions we consider them in the context of a related security notion called desynchronisation resistance. We will use our notion of ownership to give a formal definition of desynchronisation resistance.

3.1 Ownership Test Protocol

We define ownership of a tag as the ability to execute a designated protocol with the tag. This could, for example, be a mutual authentication protocol or a tag identification protocol. We call this protocol the *ownership test protocol*, or just test protocol when it is clear that we are testing for ownership.

This approach has been chosen over a knowledge-based solution, in which knowledge of a secret on the tag indicates ownership, because it is more general. It allows, for example, to include trusted or other third parties in the decision of ownership.

It is not necessarily the case that this test protocol is actually implemented on the tag. It might be a virtual protocol, merely used to define ownership-dependent security properties. Consequently, in every system state the ownership relation between tags and agents is precisely defined, while the (hypothetical) executions of the ownership test protocol do not occur in the traces of the system.

The ability to execute the test protocol proves ownership of a tag. The test protocol must therefore test whether the reader has all necessary knowledge and can perform all necessary steps. In some contexts the knowledge of a key may be the defining notion of ownership, while in others it may be the ability to

execute some or all protocols implemented on a tag. In the former setting, a simple proof-of-knowledge protocol would be a suitable test protocol, while in the latter setting it would be the collection of protocols implemented on the tag.

A consequence of our approach to define ownership relative to a test protocol is that all notions based on this definition, such as *ownership transfer*, are also relative to the chosen test protocol. The choice of a proper test protocol is therefore an important step in all verification efforts. Choosing an insufficient test protocol may lead to ownership-related vulnerabilities being overlooked. A trivial example is the test protocol that can be successfully executed by any agent and which thus declares everyone as the owner of a tag. This problem is, however, mitigated by the fact that an intuitive notion of ownership frequently coincides with the ability to complete a mutual authentication protocol with a tag. In these cases, such an authentication protocol can simply be taken as the test protocol.

3.2 Micro Traces

Testing for ownership of a tag in a certain state s amounts to verifying whether the test protocol can be successfully executed in a virtual environment whose initial state is s. This way we can verify whether the system trace can be extended, in such a way that the test protocol is successfully executed, without disturbing the system's traces. In order to model this, we introduce the notion of *micro traces*. These can be defined in a similar manner to the traces described in Section 2.3 by allowing only one run for each of the parties involved and disallowing intruder activities.

We denote by $\mu\mathsf{traces}_{P(a_1,\ldots,a_n)}(s)$ the micro traces for protocol P when executed by agents $a_1 \ldots a_n$, starting from initial state s. For every role, we allow exactly one run creation. Since we will not verify security claims against micro traces, no intruder has to be modelled. Therefore all messages sent between agents will be delivered eventually.

This results in a restriction, compared to the traces given in Section 2.3, on the semantics as defined in Section A.2. In particular the condition of the *create* rule, as given in Section A.2.1, is extended with the following requirements:

$$n \in \{a_1 \ldots a_n\} \quad n \notin agents(A)$$

where we use $agents(A)$ to denote the set of agents in the active runs A. This restricts the agents to a single run creation. Furthermore, the *deliver* rule from Section A.2.3 is required in order to have communication between the agents. Finally, the *eavesdrop*, *block*, and *inject* rules from Section A.2.3 are not allowed, such that the intruder is not capable to interfere with the protocol execution.

3.3 System View

Using the micro traces we can give a formal definition for ownership in RFID systems. We stress that this definition of ownership is not the definition of a security requirement. Instead it forms the basis to define security requirements, in particular secure ownership (Section 3.5.1), exclusive ownership (Section 3.5.2), and secure transfer (Section 4.3.1).

Definition 3.1 (Tag Owner)**.** Let $\mathcal{A}$ be a projection from system states to active runs. An agent R is owner of tag T with respect to test protocol P in system state s, denoted by $\mathsf{owns}_P(R, T, s)$, if and only if

$$\exists_{t \in \mu\mathsf{traces}_{P(R,T)}(s)} \forall_{r \in \mathcal{A}(\Sigma(t)_{|t|})} \mathsf{success}(r, \Sigma(t)_{|t|}).$$

Informally, an agent R *owns* a tag T *with respect to a test protocol* P, if in absence of all adversarial activity, all participating agents can successfully terminate their runs. Hence together they can successfully execute the test protocol P. In this context, R is called the *owner* of T with respect to P and T is called R's *property* with respect to P.

3.4 Agent View

The definition of tag ownership, as given in the previous section, allows one, in every state of the system, to verify whether a reader owns a tag. It misses, however, the owner's point of view. This view is, for example, important when discussing the intention of an owner to transfer his ownership, that is, the fact that the owner engages in an ownership transfer protocol. Merely based on this definition, it is not possible to define any meaningful security properties. This is due to the fact that, solely based on the definition of a tag owner, an owner is not aware of which tags he owns. Thus we introduce the agent's view regarding ownership of a tag by defining tag *holders*.

A tag holder is an agent which, based on its protocol executions and local data structure, believes it is the owner of a tag. We model whether a reader holds a tag T with respect to test protocol P by a global variable $\mathit{holds}(P, T)$. A mapping G from agents to variable assignments is part of the system state, as described in Section 2.3. We use σ to denote a variable assignment, which is a mapping from variable names to their value. This approach allows for a quick and easy way for an agent to determine ownership.

Definition 3.2 (Tag Holder)**.** Let R be a reader and s be a system state $\langle A, G, SB, RB, I\rangle$ such that $G(R) = \sigma$ is the global variable assignment for R. We call R a holder of tag T with respect to test protocol P in system state s, denoted by $\mathsf{holds}_P(R, T, s)$, if and only if

$$\sigma(\mathit{holds}(P, T)) = \mathit{true}.$$

By modelling tag holding explicitly we can let the protocol execution depend on the value of the *holds* variable. This allows us, for instance, to specify that an agent shall not transfer ownership of a tag, unless it actually holds the tag.

3.5 Security Requirements

In an ideal world, the notions of tag owner and tag holder always coincide. It is, however, immediate that this is impossible to achieve in an asynchronous communication model. Tag ownership changes when a tag updates its knowledge. Due to asynchronicity, it is in general not possible for an agent to update its *holds* variable simultaneously with the ownership change.

We define two security requirements for ownership. One which maintains consistency between these two views, and another which disallows multiple owners while an agent is holding a tag.

3.5.1 Secure Ownership

We define *secure ownership* as a consistency requirement on all states. We say that a set of protocols provides secure ownership, if, whenever an agent is holder of a tag, it must also be the owner of that tag. Phrased differently, a tag holder never loses his ownership unintentionally.

Definition 3.3 (Secure Ownership)**.** A set of protocols Π provides secure ownership with respect to test protocol P if in all states holding a tag implies owning that tag.

$$\forall_{t \in \mathsf{traces}(\Pi)}\ \forall_{0 \leq i \leq |t|}\ \forall_{R,T \in \mathsf{Agent}}\ \mathsf{holds}_P(R, T, \Sigma(t)_i) \Rightarrow \mathsf{owns}_P(R, T, \Sigma(t)_i).$$

Secure ownership provides a guarantee to the owner that it cannot be "disowned" as long as it holds a tag. However other agents might have simultaneous ownership of the tag. This is possible with our current definitions since there might be multiple agents which are able to execute the test protocol successfully. Therefore we introduce the notion of *exclusive ownership* which disallows simultaneous ownership.

3.5.2 Exclusive Ownership

Exclusive ownership guarantees that the holder of a tag is the sole owner of the tag. This is important, for instance, when nobody (and in particular no previous owner) but the holder of a tag is supposed to be able to identify or trace a tag. We define *exclusive ownership* as the requirement that if an agent holds a tag, no other agent is owner of the tag.

Definition 3.4 (Exclusive Ownership)**.** A set of protocols Π provides exclusive ownership with respect to test protocol P if and only if

$$\forall_{t \in \mathsf{traces}(\Pi)}\ \forall_{0 \leq i \leq |t|}\ \forall_{R,T \in \mathsf{Agent}} \\ \mathsf{holds}_P(R, T, \Sigma(t)_i) \Rightarrow \neg \exists_{R' \in \mathsf{Agent} \setminus \{R\}}\ \mathsf{owns}_P(R', T, \Sigma(t)_i).$$

It is clear that when a protocol achieves untraceability, the tag owners are the only agents which can trace the tags. Exclusive ownership is thus a necessary condition for ownership transfer protocols to satisfy untraceability against previous and future owners of tags. This requirement is therefore verified when analysing protocols which are claimed to satisfy untraceability.

3.5.3 The Jäppinen and Hämäläinen Protocol

As an example we will show an attack on secure ownership on an ownership transfer protocol recently proposed by Jäppinen and Hämäläinen [JH08]. To the best of our knowledge, this flaw has not been reported before.

This protocol is proposed as an enhanced version of the protocol proposed by Osaka et al. [OTYT06], which has first been reported broken by Lei and

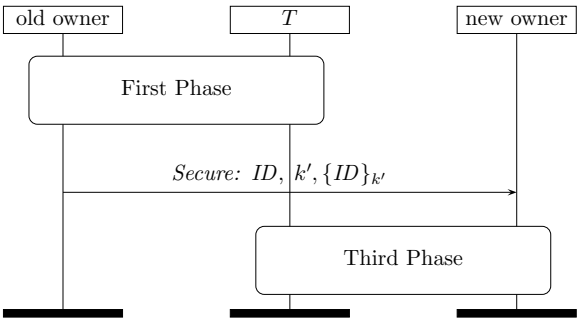

Figure 3.1: Protocol structure of an efficient and secure RFID security method with ownership transfer [OTYT06]

Cao [LC07]. Like the original protocol it relies on a shared secret $p = \{ID\}_k$ between owner and tag, called a *pseudonym* since the real *ID* is hidden.

The ownership transfer protocol consists of three phases as shown in Figure 3.1. The first and third phase consist of executions of the protocol depicted in Figure 3.2. In the first phase, the old owner updates the pseudonym p, using a fresh key k'. While in the second phase this key, together with the real identity *ID* and the pseudonym, is send to the new owner over a secure channel. Finally, in the third phase, the new owner updates the pseudonym again using its own fresh key. This structure is common for all protocols based on the work by Osaka et al. The differences between the various proposals are in the update protocol.

The only way, which makes any sense, to indicate the owner is by means of the pseudonym. Therefore we use a proof-of-knowledge protocol for p as the ownership test protocol. This protocol is depicted in Figure 3.3. We can now analyse the protocol with respect to secure ownership.

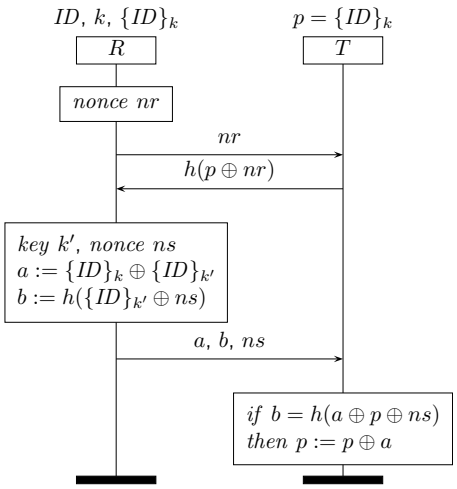

Figure 3.2: Enhanced RFID security method with ownership transfer [JH08]

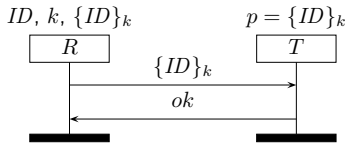

Figure 3.3: Ownership test protocol for the pseudonym p

An adversary can attack this protocol without the presence of a genuine reader, as shown in Figure 3.4. The attack is based on updating the pseudonym using some random value x, which is not known by the owner. The tag verifies the structure of the message, but this does not provide reader authentication. Therefore the attacker can easily construct a message $a = nr \oplus x, b = h(p \oplus nr), x$ which is accepted by the tag.

After the attack the owner has no knowledge of the new pseudonym $\{ID\}_k \oplus nr \oplus x$. Hence the owner will no longer be able to successfully execute the test protocol. This means that the original owner has lost its ownership of the tag. Assuming that the original owner was holder of the tag, secure ownership is violated.

The general idea behind this attack is to desynchronise the tag and reader. This special class of denial-of-service attacks is known as desynchronisation attacks. In Section 3.6 we study the relation between ownership, as defined in Section 3.3, and desynchronisation attacks.

Remark 3.1. For this protocol the intruder has the capability to synchronise reader and tag again. This can be achieved by performing the attack, as given in Figure 3.4, again with the same values for nr and x. Because of the algebraic properties of the $\oplus$-operator these values will be cancelled and the pseudonym becomes $\{ID\}_k$ again.

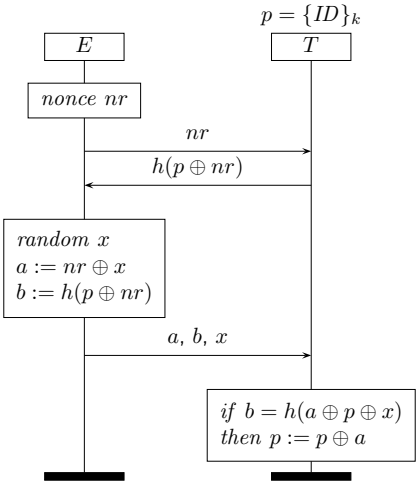

Figure 3.4: Attack on secure ownership

3.5.4 The Lei and Cao Protocol

In contrast to the previous example Lei and Cao [LC07] improve the protocol proposed by Osaka et al. [OTYT06] such that it does provide secure ownership. We discuss their protocol in Section 4.3.4, where we also address other security requirements besides secure ownership.

3.6 Desynchronisation

As a first application of the ownership definition we study desynchronisation attacks on stateful protocols. Although it is easy to characterise desynchronisation for some given protocol (by inspection of the values of the involved variables), it is not straightforward to transform this into a generic definition of desynchronisation. In this section we will study how the notion of ownership, as introduced before, can be used to define desynchronisation.

The execution of a stateful RFID protocol frequently ends with both reader and tag updating the shared identifier or key. It is clear that both reader and tag need to perform the update in an identical manner.

An incorrect protocol may allow the adversary to disrupt the communication such that one of the two agents does not carry out its update. Alternatively, the adversary may be able to force the tag and reader to carry out a different update, for example as shown in the previous section. A flawed protocol will not allow the agents to recover from this disruption and the reader and tag will be in a state of *desynchronisation*: they will no longer be able to successfully communicate with each other. We call a protocol that is not vulnerable to this type of attack *desynchronisation resistant*.

3.6.1 Desynchronisation Resistance

In general, stateful RFID authentication protocols do not need to verify ownership requirements, since the owner of a tag never changes. We argue, however, that our notion of ownership is closely related to desynchronisation resistance. Indeed, a tag that has no owners is desynchronised from all readers. This means that there does not exist a reader that can successfully communicate with the tag.

We can now define desynchronisation resistance using our ownership definition. We say that a protocol P is desynchronisation resistant, if a tag never loses all its owners with respect to P.

Definition 3.5 (Desynchronisation Resistance)**.** A protocol $P \in \Pi$ is desynchronisation resistant if and only if

$$\forall_{t \in \mathsf{traces}(\Pi)}\ \forall_{0 \leq i < |t|}\ \forall_{T \in \mathsf{Agent}} \\ (\exists_{R \in \mathsf{Agent}}\ \mathsf{owns}_P(R, T, \Sigma(t)_i)) \Rightarrow \exists_{R' \in \mathsf{Agent}}\ \mathsf{owns}_P(R', T, \Sigma(t)_{i+1}).$$

It is interesting to note that desynchronisation resistance together with exclusive ownership can imply secure ownership. Therefore in order to prove secure ownership with respect to a test protocol P it is sufficient, under the conditions stated in the following theorem, to prove desynchronisation resistance of P and exclusive ownership with respect to P. Note that the second

condition in the theorem corresponds to putting assignments of *true* to *holds* at a point in which an agent is sure to have become owner of a tag. This is discussed more extensively in Section 4.2.

Theorem 3.1. Let Π be a set of protocols containing the test protocol P. Suppose that Π provides exclusive ownership with respect to P and that P is desynchronisation resistant. Then Π provides secure ownership with respect to P for every trace which satisfies the following two conditions.

(1) In the initial state every holder of a tag is owner of the tag.

(2) An agent only becomes holder of a tag if it owns the tag.

Proof. Suppose towards a contradiction that there is a trace $t \in \mathsf{traces}(\Pi)$ such that in a state $\Sigma(t)_i$ an agent R holds a tag T, but does not own the tag. By condition (2) the agent has not become holder of T in state $\Sigma(t)_i$. Thus there must be a state $\Sigma(t)_j$, $1 \leq j < i$, in which the agent became holder of the tag. By exclusive ownership, no other agent owns the tag in state $\Sigma(t)_i$. Desynchronisation resistance implies that if no agent owns T in a state $\Sigma(t)_i$, then no agent could have owned T in state $\Sigma(t)_{i-1}$. By condition (2) no agent could have become holder in state $\Sigma(t)_{i-1}$. This argument can be repeated to conclude that no agent could have owned T in the initial state and no agent could become holder in the states $\Sigma(t)_1, \ldots, \Sigma(t)_i$. Thus R must have been the holder in the initial state. This contradicts condition (1). □

3.6.2 The Song and Mitchell Protocol

Song and Mitchell [SM08] propose a stateful RFID protocol that relies on a shared secret for authentication. Their protocol achieves identification and authentication of the tag and can therefore be used in scenarios such as supply

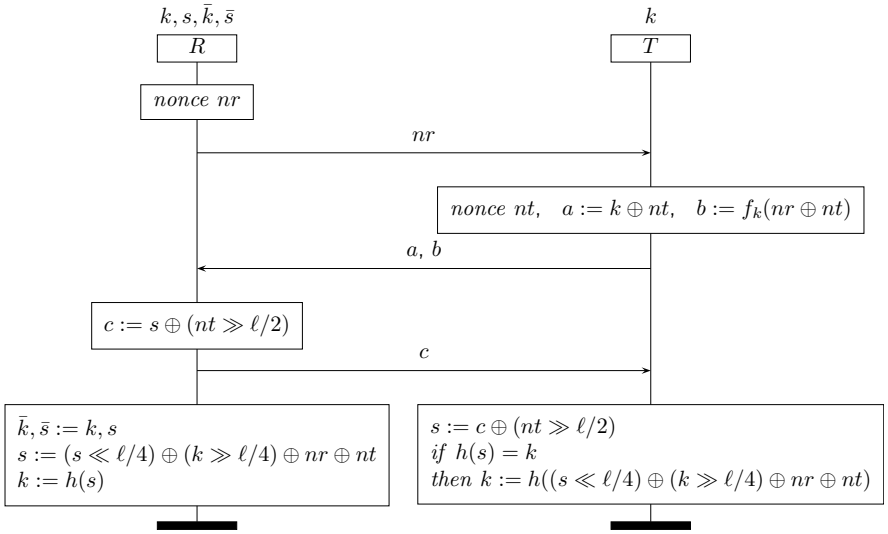

Figure 3.5: RFID authentication protocol for low-cost tags [SM08]

chain management or access control. They notice that in many proposed protocols tags and readers can be desynchronised by blocking certain messages from reader to tag. They attempt to prevent desynchronisation attacks by storing additional information, allowing the reader to re-synchronise with a tag in case messages are blocked. In this section we show that this mechanism is insufficient to provide desynchronisation resistance by describing an attack that has previously gone unnoticed. This is, however, not the first attack on this protocol. Van Deursen and Radomirović [DR08a] describe an attack on tag authentication.

We demonstrate that by modifying and blocking certain messages an attacker can force a tag and reader to carry out different updates of their shared secret. As a result, the reader loses ownership of the tag.

The protocol specification is given in Figure 3.5. We use $f_k(\_)$ to denote a keyed hash function. A cyclic right (or left) shift of a over b bits, is denoted by $a \gg b$ (or $a \ll b$). The length of the value to be shifted is denoted by ℓ.

We assume that the attacker does not know the shared secret between the tag and reader. To attack desynchronisation, the attacker proceeds as follows.

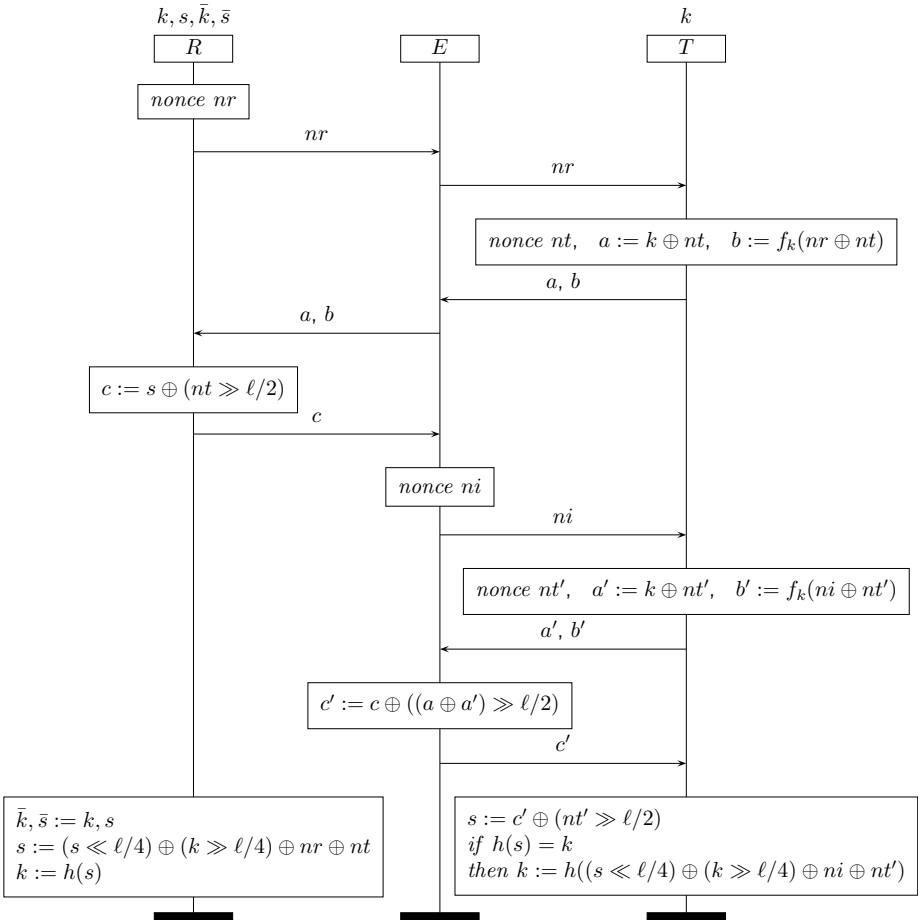

Figure 3.6: Desynchronisation attack

The attacker first eavesdrops on a genuine protocol execution between a reader and a tag. He obtains the first two messages (nr and a, b) and then aborts the protocol by blocking the third message (c). Note that the tag has not successfully completed its run and therefore does not carry out its update.

As shown in Figure 3.6, the attacker then challenges the same tag with his own, freshly generated nonce ni. The tag responds with a', b', where $a' = k \oplus nt'$ and $b' = f_k(ni \oplus nt')$. Using distributivity of $\oplus$ over $\gg$, the attacker can now construct a valid reader response $c' = c \oplus ((a \oplus a') \gg \ell/2) = s \oplus (nt' \gg \ell/2)$. The tag accepts the message and updates its k to $h((s \ll \ell/4) \oplus (k \gg \ell/4) \oplus ni \oplus nt')$.

As soon as the tag carries out its update the genuine reader loses its ownership. Indeed, it cannot successfully execute the test protocol anymore, since the key k of the tag is not known by the reader. Furthermore, since the attacker never learns the secret k, nobody has ownership over the tag. Thus, the protocol is not desynchronisation resistant.

Chapter 4

Ownership Transfer

"I did not 'get hold of it', I was given it," said the wizard.
– Gandalf in The Hobbit by J.R.R. Tolkien

In the previous chapter we have defined ownership and related security requirements. We have also shown an application of this definition. However, such ownership relations only consider system states. In this chapter we study the process of ownership transfer. We introduce security requirements for executions of ownership transfer protocols.

4.1 Ownership Transfer Protocol

We call a protocol Q an *ownership transfer protocol* if it satisfies the following functional requirement. By executing Q an agent can become the owner of a tag, if it has not been the owner of that tag.

Definition 4.1 (Ownership Transfer Protocol)**.** Let P be an ownership test protocol. We say that $Q \in \Pi$ is an ownership transfer protocol with respect to P if and only if

$$\exists_{t \in \mathsf{traces}(\Pi)}\ \exists_{0 \leq i < |t|}\ \exists_{R,T \in \mathsf{Agent}}\ \neg \mathsf{owns}_P(R, T, \Sigma(t)_i) \wedge \mathsf{owns}_{Q \cdot P}(R, T, \Sigma(t)_i),$$

where $Q \cdot P$ is used to denote sequential protocol composition.

Informally, the definition states that Q is an ownership transfer protocol, if there exists an agent R for whom the following two conditions are met. First, R is not an owner of T and hence cannot successfully complete the test protocol P with T. Second, R is able to successfully complete the sequential composition of Q and P with a tag T. Hence R is an owner of T after executing Q, since he can successfully complete P.

Remark 4.1. This definition only specifies the provided functionality to the new owner, that is, the reader R can become owner of the tag T. This is the minimal functional requirement for an ownership transfer protocol. Other requirements, which might for example put restrictions on the original owner, can be defined separately. This "modular" approach allows for flexible usage such that it can fit various situations.

4.2 Signals

In order to reason about the agent's view of ownership in a transfer protocol, we need to keep track of the events in a trace in which an agent *changes* the value of the *holds* variable. For this purpose we decorate protocols with *obtain* and *release* signals as follows. Events in which *true* is assigned to the *holds* variable will be accompanied by an obtain signal, whereas events with an assignment of *false* will be accompanied by a release signal.

Note that a signal is only emitted if value of the *holds* variable changes. Therefore no signal will be emitted if *true* or *false* is assigned while the *holds* variable already has this value. This also implies that only a tag holder can release a tag.

For a trace $t = t_0 \dots t_{n-1}$, $0 \leq i < n$, we write $t_i = \mathsf{obtain}_P(B, T, A)$ to denote an event accompanied by an obtain signal. We call such an event t_i an obtain event. We then say that agent B obtained tag T, apparently from agent A, in state $\Sigma(t)_{i+1}$. Similarly, $t_i = \mathsf{release}_P(A, T, B)$ denotes an event accompanied by a release signal in which agent A releases tag T, apparently to agent B. We call such an event t_i a release event.

For secure and exclusive ownership it is important that the release and obtain signals occur in the right places in the protocol execution. Therefore the assignments to the *holds* variable should be placed in the correct positions in the protocol specification. The release signal should occur at a point causally preceding a tag's ownership update, typically at the start of the role for the current owner of the tag. The obtain signal should occur at a point causally following a tag's confirmed ownership update, thus typically at the end of the role for the new owner.

It is easy to see that if a release signal appears too late or an obtain signal appears too early, an agent may be holder of a tag while not owning the tag, thus violating secure ownership. This holds in a similar fashion for exclusive ownership.

4.3 Security Requirements

The security requirements for ownership only provide protection as long as there is a tag holder. To extend this protection during a transfer we introduce a security requirement for transfers. Furthermore, we discuss exclusivity from a protocol point of view.

4.3.1 Secure Transfer

We say that a set of protocols provides *secure transfer*, if, whenever an agent R becomes owner of a tag, it must be as a result of an execution of an ownership transfer protocol. That is, the ownership change must be intentional.

To capture an agent's intention to give up ownership, we require that every change in ownership, making R owner of T, must be preceded by a release signal.

We restrict the relation between ownership changes and release signals in two ways. First, the ownership change must be in a one-to-one correspondence with the release signals. Hence one release signal must not be the source of two or more ownership changes. An ownership change which is the result of the

execution of an ownership transfer protocol is the natural one corresponding to a release signal. Any other changes which might be related to the same signal do not correspond to the execution of the protocol and are therefore undesired.

Second, no corresponding release and ownership-change events related to T may interleave other corresponding release and ownership-change events of T. That is, the one-to-one map must be such that the ownership change for T is mapped to the latest preceding release signal for T. For changes due to the execution of an ownership transfer protocol this holds immediately. However, the release signal of a failed transfer might be abused to validate an unauthorised change in ownership, which is to be prevented.

For tags owned by the intruder, these requirements cannot be enforced. Therefore, an agent R can become owner of a tag, either as a consequence of the tag being intentionally released to R or as a consequence of the tag being released to the agent E controlled by the intruder. In the latter case the intruder must have made R the new owner without properly releasing the tag.

Definition 4.2 (Secure Transfer)**.** Let Event denote the set of all possible events and let $E \in \mathsf{Agent}$ be the agent controlled by the intruder. A set of protocols Π provides secure transfer with respect to P if and only if

$$\begin{aligned}\forall_{t\in\mathsf{traces}(\Pi)}\ \exists_{f:\mathsf{Event}\to\mathsf{Event},injective}\ \forall_{0\le k<|t|}\ \forall_{R,T\in\mathsf{Agent}}\\ \neg\mathsf{owns}_P(R,T,\Sigma(t)_k) \wedge \mathsf{owns}_P(R,T,\Sigma(t)_{k+1}) \Rightarrow\\ \exists_{0\le i\le k}\ f(t_k) = t_i \wedge \neg\exists_{i<j\le k}\ t_j = \mathsf{release}_P(*,T,*) \wedge\\ (t_i = \mathsf{release}_P(*,T,R) \vee t_i = \mathsf{release}_P(*,T,E)),\end{aligned}$$

where $*$ is used to represent any agent.

4.3.2 Exclusive Transfer

As already noted in Remark 4.1 the definition of an ownership transfer protocol only describes the minimal requirement. The result of this definition is that at least a new owner can be introduced. This can be extended such that the protocol achieves a *strict transfer*, that is, the previous owner loses ownership while transferring to the new owner.

In order to model this we use the information from the obtain signal. We require that the agent R' from whom the new owner R obtained the tag no longer owns the tag when the obtain signal is emitted.

Definition 4.3 (Strict Transfer)**.** Let $Q \in \Pi$ be a protocol, T a tag, and R an honest reader. We say Q is a strict ownership transfer protocol with respect to test protocol P, if and only if

$$\begin{aligned}\forall_{t\in\mathsf{traces}(\Pi)}\ \forall_{0\le i<|t|}\ \forall_{R'\in\mathsf{Agent}\setminus\{R\}}\\ t_i = \mathsf{obtain}_P(R,T,R') \Rightarrow \neg\mathsf{owns}_P(R',T,\Sigma(t)_{i+1}).\end{aligned}$$

Secure transfer provides the guarantee to anybody releasing a tag that the only possible new owner is the intended one. Assuming *secure ownership*, the only guarantee provided to the new owner, is that he actually is the owner of the tag. If we extend this with our definition for *strict ownership* we introduce the guarantee that the previous owner loses ownership. This does, however,

not provide a proper guarantee since it does not exclude others from owning it. Therefore we introduce the notion of *exclusive transfer* in analogy to exclusive ownership.

Definition 4.4 (Exclusive Transfer)**.** Let $Q \in \Pi$ be a protocol, T a tag, and R an honest reader. We say Q is an exclusive ownership transfer protocol with respect to test protocol P, if and only if

$$\forall_{t \in \mathsf{traces}(\Pi)}\ \forall_{0 \leq i < |t|} \\ t_i = \mathsf{obtain}_P(R, T, *) \Rightarrow \neg \exists_{R' \in \mathsf{Agent} \setminus \{R\}}\ \mathsf{owns}_P(R', T, \Sigma(t)_{i+1}),$$

where $*$ is used to represent any agent.

It can easily be seen that exclusive transfer implies strict transfer. Furthermore, exclusive transfer is implied by exclusive ownership. A more interesting observation is that exclusive transfer together with secure transfer can imply exclusive ownership. Therefore in order to prove exclusive ownership with respect to a test protocol P it is sufficient, under the conditions stated in the Theorem 4.3, to prove secure transfer and exclusive transfer with respect to P.

Theorem 4.1. Let $Q \in \Pi$ be an exclusive ownership transfer protocol with respect to test protocol P. Then Q is also a strict ownership transfer protocol with respect to P.

Proof. Exclusive transfer states that when the new owner obtains a tag, then no other agent owns the tag, in particular not the previous owner. □

Theorem 4.2. Let Π be a set of protocols providing exclusive ownership with respect to test protocol P. Then all ownership transfer protocols $Q \in \Pi$ are exclusive ownership transfer protocols with respect to P.

Proof. For exclusive transfer it is required that there is no other owner in the state s in which the tag is obtained. This is the first state in which the agent holds the tag. From exclusive ownership it follows that in every state in which an agent holds a tag, there is no other owner, in particular not in state s. □

Theorem 4.3. Let all ownership transfer protocols $Q \in \Pi$ provide secure and exclusive transfer with respect to test protocol P. Then Π provides exclusive ownership with respect to P for every trace in which in the initial state every holder of a tag is the only owner of the tag.

Proof. Suppose towards a contradiction that there is a trace $t \in \mathsf{traces}(\Pi)$ such that in a state $\Sigma(t)_i$ an agent R holds a tag T, while there is another agent R' which owns the tag. Thus there must be a state $\Sigma(t)_j$, $0 \leq j \leq i$, in which R became holder of the tag. In case $j = 0$, by the condition on the initial state, no other agent owns the tag in state $\Sigma(t)_j$. Similarly, for $j > 0$, no other agent owns the tag in state $\Sigma(t)_j$ by exclusive transfer. Thus R' must have become owner of T in state $\Sigma(t)_k$, $j < k \leq i$. Secure transfer implies that if an agent becomes owner of T in a state $\Sigma(t)_k$, then there must have been a release in state $\Sigma(t)_l$, $l < k$. This contradicts the assumption that R is holding T in state $\Sigma(t)_i$. □

4.3.3 The Yoon and Yoo Protocol

We demonstrate our definitions on the recently published ownership transfer protocol by Yoon and Yoo [YY08]. As with the Jäppinen and Hämäläinen protocol (discussed in Section 3.5.3) this is an attempt to improve the protocol proposed by Osaka et al. [OTYT06]. Therefore it has the same phased structure. The difference is that the first and third phase now consist of executions of the protocol depicted in Figure 4.1.

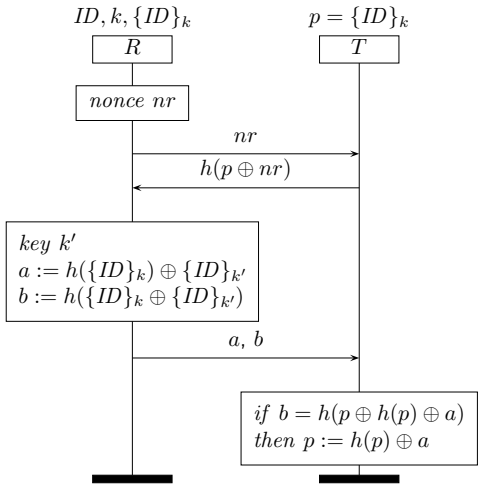

Figure 4.1: Fixed RFID security method with ownership transfer [YY08]

In accordance with Section 4.2, we put the release signal at the start of the first phase, and the obtain signal at the end of the third phase, as shown in Figure 4.2. Since the pseudonym p of the tag is all that is used in communication with the tag, we take as ownership test protocol a proof-of-knowledge protocol of p (Figure 3.3).

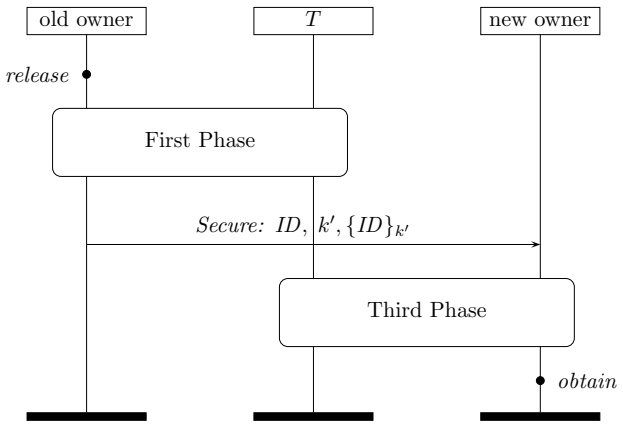

Figure 4.2: Protocol structure including release and obtain signals

Secure Transfer

Consider an execution of the protocol by R, T, and R', where initially R is the owner of the tag T and intends R' to become the new owner. We first show that the protocol does not satisfy secure transfer, because an intruder E can obtain ownership of the tag without being the intended new owner. To achieve this, the intruder queries the target tag T with the constant 0 to which the tag replies with $h(p)$. By eavesdropping on the first phase of the protocol execution, between the owner R and the target tag T, the intruder obtains $a = h(p) \oplus \{ID\}_{k'}$. As soon as the tag updates its pseudonym to $\{ID\}_{k'}$ the intruder becomes owner of the tag. This attack is depicted in Figure 4.3. The intruder can always become owner of a tag in this way even if the old owner of the tag does not release it to him. Therefore, the protocol does not satisfy secure transfer.

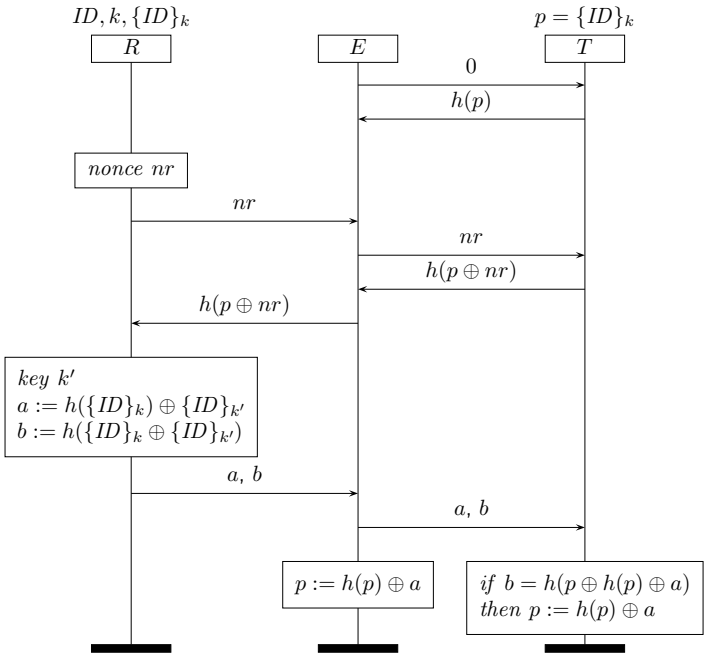

Figure 4.3: Attack on secure transfer

Exclusive Ownership

Next, we show that exclusive ownership can be violated using knowledge of the tag's pseudonym the intruder has gained after the first phase of the protocol through the previous attack. The intruder eavesdrops on the third phase of the transfer, carried out by T and R'. The new owner R' becomes holder of the tag when the third phase finishes. Using the information learnt during this phase the intruder can derive the new pseudonym as he did in the previous attack, hence he remains owner of the tag. This violates exclusive ownership since R' holds the tag, while the intruder owns the tag.

Secure Ownership

After R' has finished the transfer, the intruder executes the pseudonym update protocol to update the tag's pseudonym to a pseudonym the new owner R' does not know. Therefore R' loses ownership while still being holder of the tag which violates secure ownership.

4.3.4 The Lei and Cao Protocol

In this section we demonstrate our definitions on the published ownership transfer protocol by Lei and Cao [LC07]. In contrast to the protocol of the previous section, this protocol improves the protocol proposed by Osaka et al. [OTYT06] such that it provides secure ownership and secure transfer, but not exclusive ownership, although it is claimed to provide untraceability.

This protocol has the same structure as described in Sections 3.5.3 and 4.3.3. We will therefore use the same signal placement (Figure 4.2) and test protocol (Figure 3.3) as in the previous analyses. The update protocol for the first and third phase is given in Figure 4.4.

For secure ownership as well as secure transfer we provide security analyses to show how the protocol achieves these security requirements. These are not proofs, but rather proof sketches. Model checking is necessary to provide full guarantee that this protocol adheres to these requirements. This is part of future research.

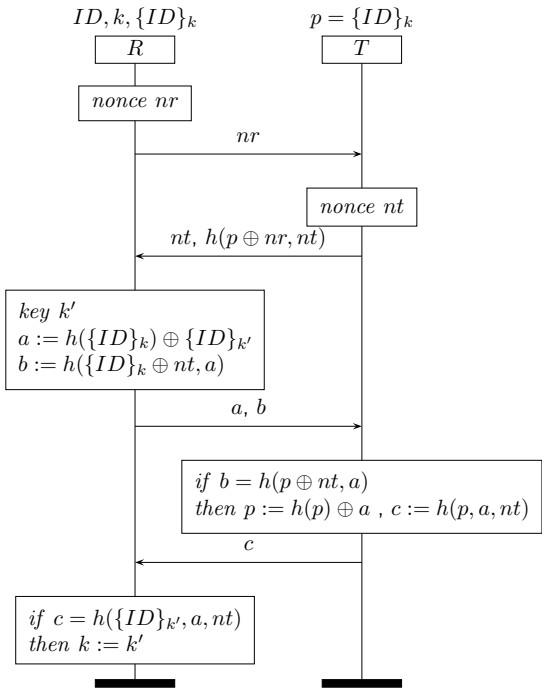

Figure 4.4: RFID protocol enabling ownership transfer to protect against traceability and DoS attacks [LC07]

Secure Ownership

Lei and Cao as well as Osaka et al. do not consider backward security of their protocols. Following this approach we assume the previous owners to be honest.

In order to achieve secure ownership we assume that initially all tag holders are also tag owners. Thus the correspondence relation between holds and owns holds in the initial state. Furthermore we assume that this is the only ownership transfer protocol in the set of protocols Π.

Secure ownership can be violated in two ways. Either a holder loses ownership, or an agent becomes holder while not owning the tag. This first case can be caused by executing the ownership transfer protocol or due to an action of the intruder. When the ownership transfer protocol is executed the tag is released such that the agent is no longer a holder. While intervention of the intruder is prevented by authenticating the update message such that only the owner can send it.

The second case is prevented since the new owner must know the pseudonym p in order to successfully complete the third phase. When the message c has arrived the tag has confirmed that the pseudonym has been set on the tag. Hence, when the tag is obtained the agent is indeed the owner.

Secure Transfer

Secure transfer concerns the situation in which an agent becomes owner. This is only possible by knowing the pseudonym of the tag. By executing the ownership transfer protocol the new owner learns the pseudonym due to the communication with the previous owner in the second phase. In this case the ownership change has been preceded by a release signal.

An intruder can only learn the pseudonym from the communication between reader and tag. To achieve this he should either know the hash of the previous pseudonym, or be able to invert a cryptographic hash function, which we both assume to be infeasible.

We also need to verify that a release signal cannot be abused. In the current setting an old release can be used since the pseudonym is not updated before the new release is signalled. However, if we put the signal just before the second phase this should be fine, since now the old release has been invalidated due to the pseudonym update.

Exclusive Ownership

Lei and Cao as well as Osaka et al. consider indistinguishability (untraceability) of their protocols. In contrast to the analysis for secure ownership, we do not assume the previous owners to be honest. Assuming honesty of the previous owner does not make any sense, since this requirement provides a guarantee for the new owner, concerning all other agents.

It can easily be seen that this protocol does not provide exclusive ownership. By eavesdropping on the communication between the new owner and the tag, the previous owner obtains the message a, b. From this message the new pseudonym can be derived $a \oplus h(\{ID\}_k) = \{ID\}_{k'}$. With this knowledge the previous owner can successfully complete the test protocol, meaning he is still owner, which violates exclusive ownership.

Chapter 5

Design and Analysis

An explanation of cause is not a justification by reason.
– C.S. Lewis

In the previous chapters we have defined a framework for analysing security aspects of ownership transfer protocols. In this chapter we present two protocols designed to meet all of the defined security requirements. The first approach uses asymmetric cryptography and achieves a secure exclusive ownership transfer protocol which satisfies all requirements.

For the second approach, we use only symmetric cryptography, which decreases the computational complexity. The proposed protocol satisfies secure ownership as well as secure transfer. However, exclusive ownership can only be proved under an additional assumption.

5.1 Design Strategy

As a general design strategy we will base our protocols on existing authentication protocols. During the literature study as well as the development of the analysis framework we noticed that most flaws are caused by flawed authentication.

Based on this observation we studied the relation between authentication and ownership transfer. The result from this study is that authentication protocols are ideal building blocks for ownership transfer protocols. The protocols described below use the structure as given in Figure 5.1. Based on related work (Section 2.2), similar to Osaka et al. [OTYT06], we identified three phases to be important for ownership transfer. First, the current owner and the intended new owner *exchange* information about the tag to be transferred. Next the current owner *releases* the tag such that, finally, the new owner can *obtain* the tag in the third phase.

Remark 5.1. Depending on the exact implementation of a protocol the exchange and release phases might need to be switched or interleaved. This depends on the information that is required for these phases. For example, the exchange phase might depend on a fresh key which is generated in the release phase. Then the release has to precede the exchange. This is the case with the previously discussed ownership transfer protocols based on the work by Osaka et al. [OTYT06].

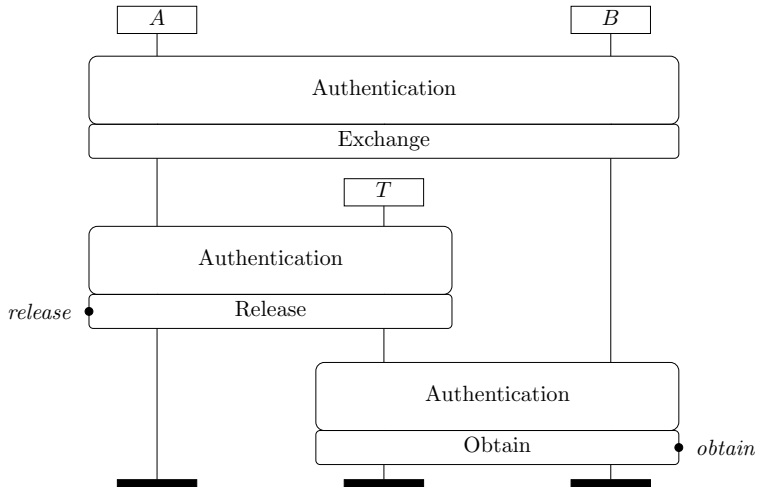

Figure 5.1: General design idea for secure ownership transfer protocols

In order to assure security of these phases each of them is combined with an authentication part. This provides protection against the intruder impersonating one of the agents. It is important to make sure that the authentication which is established by this part is used properly in the functional part such that it offers the required protection.

We now have a recipe to design ownership transfer protocols.

5.2 Using Public-key Authentication

The Needham-Schroeder-Lowe public-key authentication protocol [Low96] is used as the main construction block for our first ownership transfer protocol. Besides its own secret-key $sk(T)$, a tag T only stores a single public-key pkO, which is used to authenticate the owner. Therefore, the agent with knowledge of the corresponding secret-key is the owner. The test protocol $TEST_{NSL}$, as depicted in Figure 5.2, is an instance of the Needham-Schroeder-Lowe protocol with an additional message as explained in Remark 5.2. This protocol can also be implemented on the tag for identification and authentication purposes.

Remark 5.2. For the use in an RFID setting the Needham-Schroeder-Lowe protocol is initiated by the tag. This is preferred since the reader initially does not know the identity of the tag, hence it is unknown which public key to use for the tag.

The tag only has a single key to use such that it can simply construct the first message. Using this message the tag identity can be transferred safely to the reader, since only the owner of the tag will be able to decrypt the message using its secret-key.

Furthermore, the authentication protocol is preceded by an additional message from the reader to the tag. This message is used to initiate the communication with the tag, as well as to select the protocol on the tag. This is necessary since RFID tags are only capable of responding to reader messages.

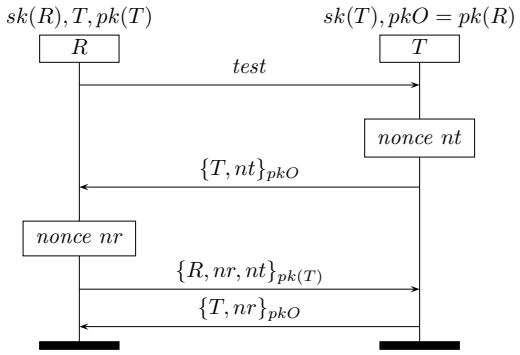

Figure 5.2: $TEST_{NSL}$: public-key ownership test protocol

5.2.1 Protocol Description

The ownership transfer protocol OTP_{NSL}, as specified by Figure 5.3, is designed according to the structure depicted in Figure 5.1. The exchange phase consists of the authentication protocol with two additional terms in the initial message, the tag's identity T as well as its public-key $pk(T)$. With this information the new owner B will be able to identify and communicate with the tag, although this is only possible after the tag has been released by the current owner A.

The first part of the release phase consists of the authentication protocol. After mutual authentication has been achieved between reader and tag the public-key stored on the tag can be updated. Furthermore, upon updating the owner key, the tag sets the *inTransfer* flag, indicating that the tag is involved in a transfer. Finally the tag confirms the update. Note that the confirmation message m is constructed before the key is updated since after the update the tag does no longer know the public-key for the original owner.

The release signal will be placed just before sending the update message to the tag. Until this moment the original owner has not given up ownership, hence there is no need to release earlier. Furthermore, it is certain that the agent owns the tag since the ownership test protocol has been executed as part of the release phase. The release signal cannot be placed any later since the agent loses ownership when the tag updates.

The obtain phase consists solely of an instance of the test protocol. There are only some small modifications on the tag's side. First, when this phase is initiated by the new owner, the tag verifies whether it is actually engaged in a transfer. This is required to avoid multiple executions of the obtain phase. After the new owner has been verified the tag will unset the *inTransfer* flag. This is confirmed to the new owner with a final message after which the obtain signal is emitted.

To avoid possible flaws due to messages being confused, with other messages within this protocol or with those from other protocols, all messages have been tagged. That is, the encrypted terms are extended with an additional term which is unique for each message in the specification. For the exchange phase we use $e_1, \ldots, e_3$, $u_1, \ldots, u_5$ for the update phase, and $v_1, \ldots, v_3$ for the obtain phase.

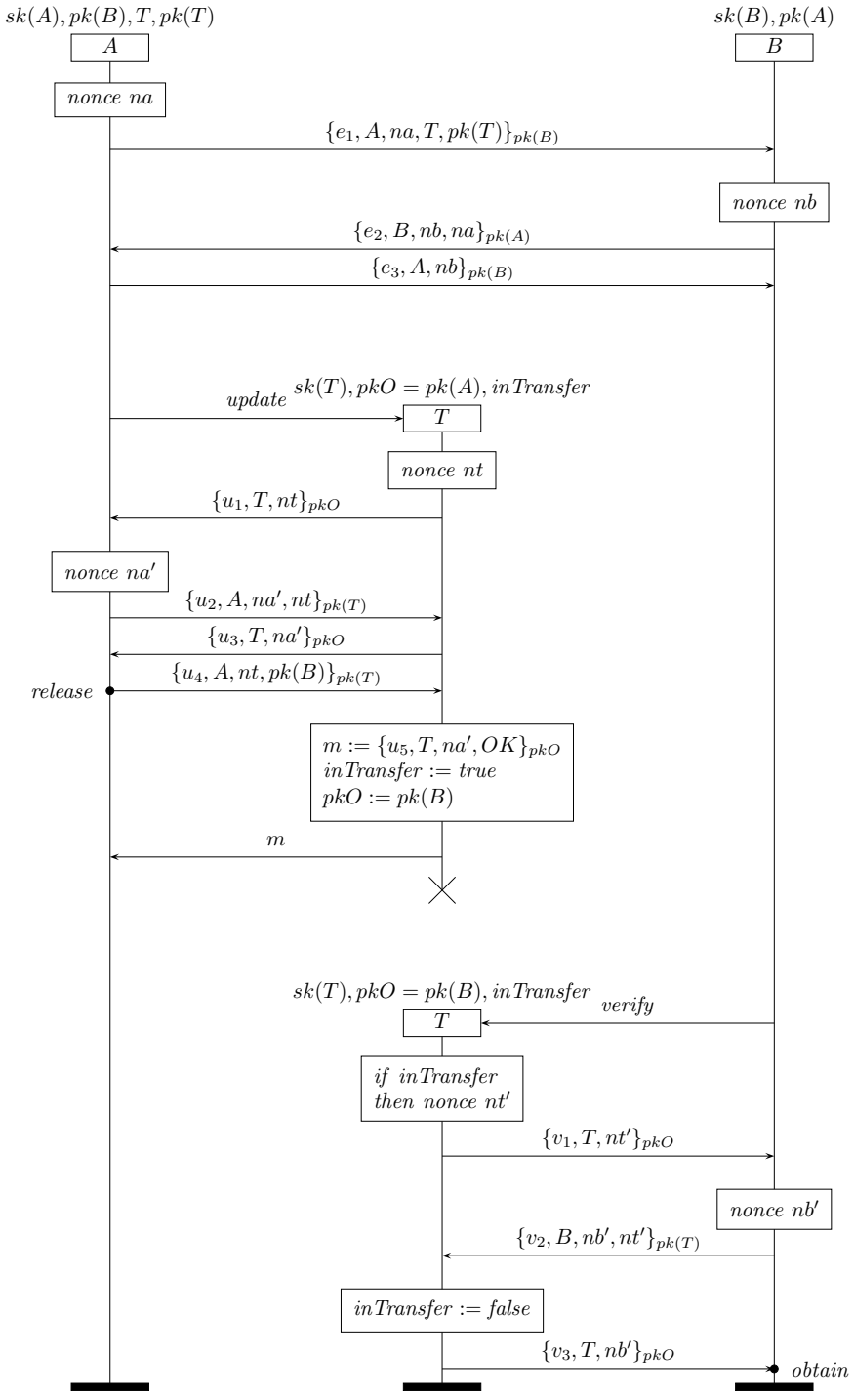

Figure 5.3: OTP_{NSL}: public-key secure exclusive ownership transfer protocol

5.2.2 Security Analysis

This section only contains a security analysis, that is, we do not provide formal proofs. Such proofs can be obtained by using model checking. This is considered as a part of future research. Note that the properties of the Needham-Schroeder-Lowe protocol, which we use in our analysis, have been proved by model checking [Low96].

Secure Ownership

To prevent unintentional loss this protocol has been designed to satisfy secure ownership. This is achieved by properly authenticating the update message in the release phase such that only the actual owner can send it. Furthermore, the obtain phase is designed according to the test protocol in order to prevent agents setting their *holds* variable, while they do not own the tag.

Lemma 5.1. Suppose that all decryption keys are only known to their corresponding agents. Then the public-key stored on the tag can only be updated by the owner executing the release phase of OTP_{NSL}.

Proof. Suppose that an intruder is able to update the key stored on the tag. Then he must have constructed a message which has been accepted by the tag. This message is authenticated by means of the Needham-Schroeder-Lowe protocol. Hence, if the intruder is capable to construct such message, without knowledge of the decryption key of the owner, then he is also able to break the Needham-Schroeder-Lowe protocol, which has been proved secure [Low96]. □

Theorem 5.1. Suppose that the initial state is consistent, that is, all tag holders own their tags, and that all decryption keys are only known to their corresponding agents. Then $\Pi = \{TEST_{NSL}, OTP_{NSL}\}$ satisfies secure ownership with respect to $TEST_{NSL}$.

Proof (Sketch). Suppose towards a contradiction that secure ownership is violated at a certain moment. Then there is an agent which holds a tag while not being the owner. This either means that the *holds* variable is set while the agent did not own it or ownership was lost while this variable was still set.

The *holds* variable is only set at the end of the new owner's run. The new owner B has proved knowledge of the decryption key of the public-key stored on the tag, by successfully executing the Needham-Schroeder-Lowe protocol. Thus B is the owner of the tag at this point. This situation cannot be violated by B already transferring the tag again, since only a holder can release the tag. Furthermore, no other agents can violate this situation since there is only one key on the tag which denotes the owner, and B is the only agent knowing this key.

Losing ownership of the tag while holding it can only be caused by updating the public-key stored on the tag. By Lemma 5.1 only the owner can do this by executing the release phase. In this phase the tag is released hence secure ownership is not violated.

Finally, $TEST_{NSL}$ and OTP_{NSL} are independent of each other because of the message tagging applied to OTP_{NSL}. Thus we can discuss $TEST_{NSL}$ in isolation. This protocol does not modify the key or the *holds* variable. Hence there is no violation possible. Therefore this protocol set satisfies secure ownership. □

Exclusive Ownership

To guarantee that a tag holder is the only owner of that tag, the protocol has been designed to satisfy exclusive ownership. This is achieved by the use of a single public-key to denote the owner of the tag in combination with the assumption that the secret-key is only known to the corresponding agent.

Theorem 5.2. Suppose that the initial state is consistent, that is, all tag holders are the only owners of their tags. Assume that all decryption keys are only known to their corresponding agents. Then $\Pi = \{TEST_{NSL}, OTP_{NSL}\}$ satisfies exclusive ownership with respect to $TEST_{NSL}$.

Proof (Sketch). The tag only stores a single public-key to authenticate the owner. In order to successfully complete the ownership test protocol an agent should know the corresponding secret-key. Since each agent has its own key, and these keys are not shared with other agents there can only be one agent owning the tag. By secure ownership (Theorem 5.1) we have that the holder of a tag must be the owner. Hence we have that the holder is the exclusive owner, since there are no other agents owning the tag. □

Secure Transfer

The previous requirements provide security as long as an agent is holder of a tag. We extend this security further such that also the transfer is protected against malicious activities. Therefore the protocol satisfies secure transfer such that the tag ends up with the intended owner. This is achieved by properly authenticating all phases of the ownership transfer protocol.

Theorem 5.3. Suppose that a release is blocking, that is, an agent does not continue execution when it cannot release a tag (since it does not hold the tag) and that the previous owner is honest. Then $\Pi = \{TEST_{NSL}, OTP_{NSL}\}$ satisfies secure transfer with respect to $TEST_{NSL}$.

Proof (Sketch). Suppose towards a contradiction that an agent achieves to become owner of a tag without the occurrence of a release signal. An agent can only become owner by knowing the decryption key for the public-key stored on the tag. This can be achieved by executing the ownership transfer protocol. The previous owner does not continue if it is not the holder of the tag. Hence if ownership changes, a release signal will be emitted since the previous owner was holder of the tag. This contradicts the assumption that there was no release signal.

The decryption key can also not be derived from the communication since the Needham-Schroeder-Lowe protocol does not leak any information. Furthermore, by Lemma 5.1, the key can only be updated by the owner. Thus an intruder cannot store his own key on the tag.

We also need to verify that a release signal cannot be abused. Injectivity and recentness of the correspondence between ownership changes and release signals follow from the injective synchronisation provided by the Needham-Schroeder-Lowe protocol. Furthermore, a release signal is only emitted if the agent is holder of the tag, hence another release signal will be preceded by an obtain signal. Thus corresponding release signals and ownership changes cannot be interleaved. □

5.2.3 Practical Analysis

The security requirements discussed in the previous section are not the only requirements for RFID systems. There are also practical requirements such as *scalability* and *computational complexity*. The former is required to be able to use the system on a large scale, while the latter is required due to the limited computational capabilities of RFID tags. We will shortly discuss these aspects of our protocol.

Scalability is achieved by having the tag initiate the Needham-Schroeder-Lowe protocol, as noted in Remark 5.2. This allows the reader to immediately identify the tag based on its reply. Therefore the identification time is constant, that it does not depend on the number of tags in the system. Hence it does not matter if the reader owns hundreds, thousands or even more tags.

High computational complexity is the main drawback of this protocol. It requires the tag to be able to perform asymmetric encryption and decryption. Furthermore, a random number generator is required for the nonce generation. With the limited computational powers of a tag this is hardly possible. Research is being done in this area [BGK+07, LSBV08], but tags with these capabilities are still not available in general. In an attempt to satisfy this last requirement another protocol, which does not use asymmetric cryptography, has been designed.

5.3 Using Shared-key Authentication

The ISO Symmetric Key Three-Pass Mutual Authentication protocol [ISO08] serves as the main ingredient in this design. This protocol has been turned into a test protocol $TEST_{ISO}$, as depicted in Figure 5.4, analogous to the previous test protocol $TEST_{NSL}$. The tag stores two symmetric keys. One owner key k_O which is shared by the owner among all its tags, and a tag key k_T which is only shared between the owner and the specific tag. Both keys have to be known in order to successfully execute the given test protocol. Again this protocol can, besides testing for ownership, also be used for tag identification and authentication.

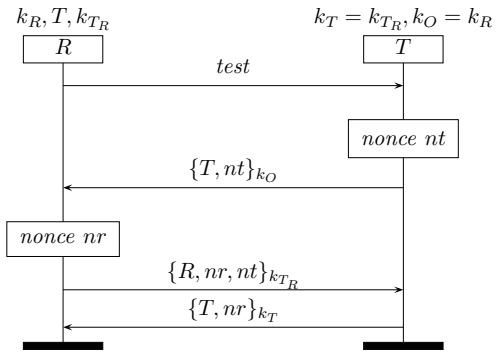

Figure 5.4: $TEST_{ISO}$: shared-key ownership test protocol

5.3.1 Protocol Description

The ownership transfer protocol OTP_{ISO}, as specified by Figure 5.5, is designed to resemble the previous protocol design (Figure 5.3). The main difference with the previous protocol is the use of symmetric encryption instead of asymmetric. For the exchange phase, however, we still use the Needham-Schroeder-Lowe protocol to avoid sharing keys between all agents. A public-key infrastructure is a more efficient solution to retrieve the keys of other agents when necessary. This use of asymmetric cryptography for the exchange phase is not a problem since readers are capable of performing public-key cryptography.

The tag information exchanged during the exchange phase consists of the tag identity T and a fresh symmetric key $k_{T_{AB}}$. This key will be put on the tag during the release phase, such that the new owner is able to communicate with the tag, in a secure fashion, in the obtain phase.

The release phase uses the same messages as the previous protocol. However, this time symmetric encryption is used. The owner key in the first reply of the tag is used to achieve tag identification, similar to the previous protocol. After the tag has been identified the tag key will be used to perform mutual authentication. Finally the tag key is updated with the exchanged new symmetric key.

In the obtain phase, in contrast to the test protocol and release phase, the tag's first reply is encrypted using the tag key. This is necessary to avoid the exchange of owner keys, by which one agent could identify the other agent's tags. After mutual authentication has been established both keys on the tag will be updated. Now the new owner is able to successfully complete the test protocol.

To avoid possible flaws due to messages being confused, with other messages within this protocol or with those from other protocols, all messages have been tagged. That is, the encrypted terms are extended with an additional term which is unique for each message in the specification. For the exchange phase we use $e_1, \ldots, e_3$, $u_1, \ldots, u_5$ for the update phase, and $v_1, \ldots, v_3$ for the obtain phase.

5.3.2 Security Analysis

This section only contains a security analysis, that is, we do not provide formal proofs. Such proofs can be obtained by using model checking. This is considered as a part of future research.

Secure Ownership

Secure ownership is, similar to the previous protocol, achieved by properly authenticating the update message in the release phase such that only the actual owner can send it. The obtain phase this time slightly differs from the test protocol. Only the owner key is not used since this key has not been exchanged and is thus unknown to the intended new owner. Instead we use the tag key which is known, and can be used by the agent to achieve mutual authentication with the tag.

Note that in contrast to the previous protocol we now need to assume previous owners to be honest. Since a malicious previous owner can easily violate secure ownership.

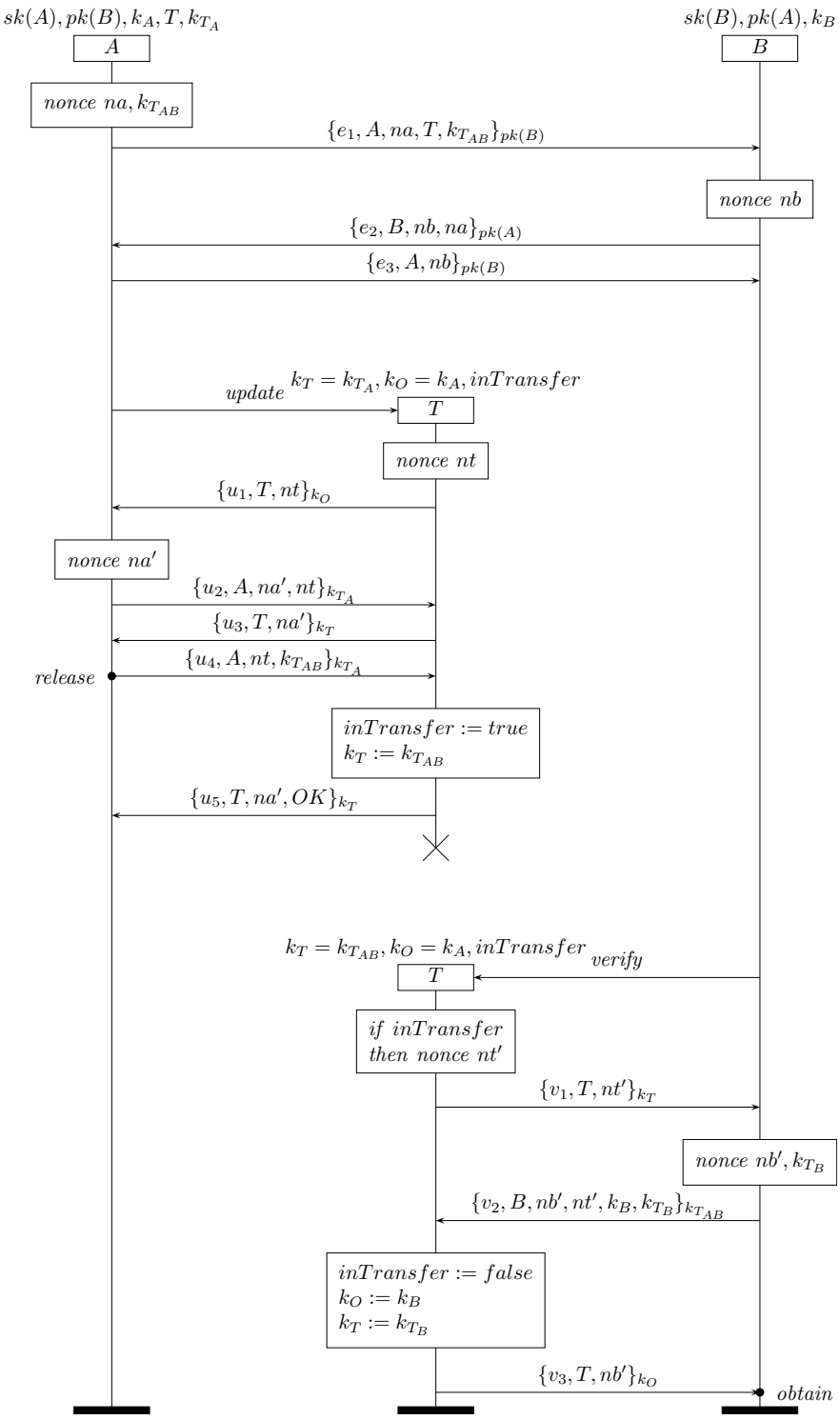

Figure 5.5: OTP_{ISO}: shared-key secure ownership transfer protocol

Lemma 5.2. Suppose that all decryption keys are only known to their corresponding agents. Then the tag key stored on the tag can only be updated by the owner executing the release phase of OTP_{ISO}.

Proof. Suppose that an intruder is able to update the key stored on the tag. Then he must have constructed a message which has been accepted by the tag. This message is authenticated by means of the ISO Symmetric Key Three-Pass Mutual Authentication protocol. Hence, if the intruder is capable to construct such message, without knowledge of the decryption key of the owner, then he is also able to break the authentication protocol, which has been proved secure [DNL99]. □

Lemma 5.3. Suppose that all decryption keys are only known to their corresponding agents and that the previous owner is honest. Then the owner key stored on the tag can only be updated by the new owner executing the obtain phase of OTP_{ISO}.

Proof. Suppose that an intruder is able to update the key stored on the tag. Then he must have constructed a message which has been accepted by the tag. Since the previous owner is considered to be honest, the intruder must be able to construct this message without knowledge from this agent. The update message is authenticated by means of the ISO Symmetric Key Three-Pass Mutual Authentication protocol. Hence, if the intruder is capable to construct such message, then he is also able to break the authentication protocol. □

Theorem 5.4. Suppose that the initial state is consistent, that is, all tag holders own their tags, that all decryption keys are only known to their corresponding owners, and that the previous owner is honest. Then $\Pi = \{TEST_{ISO}, OTP_{ISO}\}$ satisfies secure ownership with respect to $TEST_{ISO}$.

Proof (Sketch). Suppose towards a contradiction that secure ownership is violated at a certain moment. This either means that the *holds* variable is set while the agent did not own it. Or ownership was lost while this variable was still set.

The *holds* variable is only set at the end of the new owner's run. The new owner B has proved to known both the tag and owner key stored on the tag, by successfully executing the ISO Symmetric Key Three-Pass Mutual Authentication protocol. Thus B is the owner of the tag at this point. This situation cannot be violated by B already transferring the tag again, since only a holder can release the tag. Furthermore, this situation cannot be violated by any other agent, since B is the only agent knowing both the tag and owner key, which are both required to own the tag. The only other agent knowing the tag key is assumed to be honest, that is, this agent cannot derive the owner key.

Losing ownership of the tag while holding it can only be caused by updating the keys stored on the tag. By Lemma 5.3 only the new owner can update the owner key in the obtain phase. This can only be done when the *inTransfer* flag is set. This flag is only set after the tag key update. By Lemma 5.2 only the owner can update the tag key by executing the release phase. In this phase the tag is released hence secure ownership is not violated.

Finally, $TEST_{ISO}$ and OTP_{ISO} are independent of each other because of the message tagging applied to OTP_{ISO}. Thus we can discuss $TEST_{ISO}$ in isolation. This protocol does not modify the key or the *holds* variable. Hence there is no violation possible. Therefore this protocol set satisfies secure ownership. □

Exclusive Ownership

OTP_{ISO} does not satisfy exclusive ownership. The problems with satisfying exclusivity using symmetric cryptography are explained in Section 5.4.

Since exclusive ownership provides protection, among others, against the previous owner, it does not make any sense to assume the previous owner to be honest for this requirement. Thus exclusive ownership is easily violated.

Theorem 5.5. Suppose that the initial state is consistent, that is, all tag holders are the only owners of their tags. Then $\Pi = \{TEST_{ISO}, OTP_{ISO}\}$ does *not* satisfy exclusive ownership with respect to $TEST_{ISO}$.

Proof. By eavesdropping on the communication between the new owner and the tag, the previous owner obtains the message $\{v_2, B, nb', nt', k_B, k_{T_B}\}_{k_{T_{AB}}}$. From this message the new keys can be derived by decrypting the message using the tag key $k_{T_{AB}}$ which was chosen by the previous owner. With this knowledge the previous owner can successfully execute the test protocol, such that he remains owner, which violates exclusive ownership. □

Secure Transfer

The previous requirements provide security as long as an agent is holder of a tag. We extend this security further such that also the transfer is protected against malicious activities. Therefore our protocol satisfies secure transfer such that the tag ends up with the intended owner.

Theorem 5.6. Suppose that a release is blocking, that is, an agent does not continue execution when it cannot release a tag (since it does not hold the tag) and that the previous owner is honest. Then $\Pi = \{TEST_{ISO}, OTP_{ISO}\}$ satisfies secure transfer with respect to $TEST_{ISO}$.

Proof (Sketch). Suppose towards a contradiction that an agent achieves to become owner of a tag without the occurrence of a release signal. An agent can only become owner by knowing the decryption key for the public-key stored on the tag. This can be achieved by executing the ownership transfer protocol. The previous owner does not continue if it is not the holder of the tag. Hence if ownership changes, a release signal will be emitted since the previous owner was holder of the tag. This contradicts the assumption that there was no release signal.

The keys can also not be derived from the communication since the ISO Symmetric Key Three-Pass Mutual Authentication protocol does not leak any information. Furthermore, by Lemmas 5.2 and 5.3, the keys can only be updated by the owner and the new owner. Thus an intruder cannot store his own key on the tag.

We also need to verify that a release signal cannot be abused. Injectivity and recentness of the correspondence between ownership changes and release signals follow from the injective synchronisation provided by the ISO Symmetric Key Three-Pass Mutual Authentication protocol. Furthermore, a release signal is only emitted if the agent is holder of the tag, hence another release signal will be preceded by an obtain signal. Thus corresponding release signals and ownership changes cannot be interleaved. □

5.3.3 Practical Analysis

Similar to the previous design we shortly discuss scalability and computational complexity. Scalability is achieved by the use of two shared keys. Using the owner key for the first message allows identification of the tags in constant time. While the tag key is used to achieve mutual authentication. Compromise of the owner key, which is more likely since it is used for all tags of an owner, does not violate any of the ownership related security requirements. It does however allow an intruder to identify and trace tags, thus violating untraceability requirements. Not using the owner key in the obtain phase is acceptable. This is possible since the number of tags involved in a transfer for an owner at the same time will not be large. Therefore an exhaustive search for the decryption key, by checking all tag keys of tags which are involved in a transfer, can be done within a reasonable amount of time.

The computational complexity is less than in our previous design. This allows our protocol to be implemented on RFID tags which support symmetric encryption. Such tags exist, but they are still quite expensive. They are too expensive to be used for supply chain purposes where large numbers of tags are used. The current generation of tags in use hardly provides any computational power. Hence another design might be needed to reach this level of complexity. However the current design already fails to satisfy all ownership related security requirements.

5.4 Exclusivity Problems

Designing a protocol which satisfies exclusive ownership using only shared keys is not possible under the given circumstances. Using symmetric keys gives rise to two scenarios. Either the intended new owner provides the original owner with a key to put on the tag. This allows the new owner to communicate with the tag and, for example, update the key. The other option is that the original owner shares his key with the intended new owner, such that the new owner can put his own key on the tag. Both cases give rise to violations for exclusive ownership, since a malicious previous owner will be able to listen in to the communication between the tag and the new owner.

This problem concerns secure channel establishment, which has been studied by Maurer and Schmid [MS96]. OTP_{ISO} provides an authenticating channel, but there is no secret shared between the new owner and the tag. According to Maurer and Schmid public-key cryptography is required in order to construct a confidential channel from an authenticating channel. This results in the conclusion that a certain level of asymmetry is required to satisfy exclusivity.

Conjecture 5.1. In the model as given in Appendix A, without a shared secret between the new owner and the tag, exclusive transfer cannot be satisfied in a shared-key only setting (without an asymmetric aspect).

Proof (Sketch). Exclusive transfer concerns the establishment of a shared secret between the new owner and the tag. Suppose towards a contradiction that the new owner and tag can establish such secret. Then this secret must be infered from their knowledge and the communication between them. The intruder, however, has the same initial knowledge since he possesses the knowledge from

the malicious previous owner. Furthermore, the network is controlled by the intruder, such that he observes all interactions between the two agents. Therefore, the intruder is able to infer any knowledge which the agents might choose as their secret key. This contradicts the assumption that a shared secret can be established. □

To provide an actual proof this model has to be implemented in a model checker and the statement has to be verified. Our model, however, models perfect operations. Therefore this conjecture should also be stated in a computational model, which resembles the real world situations more closely. This does, however, require clarification concerning channel properties and drawing the boundary between asymmetric and symmetric cryptography within such a model.

The previous statement concerns exclusive transfer, however, based on Theorem 4.2 we can also conclude that exclusive ownership is also not possible to satisfy under these conditions. This problem can be solved by altering the situation. We discuss to possible solutions for this problem: weakening the definition or adding an assumption.

Weakening the definition to an honest previous owner solves the problem for the strict transfer definition. The previous owner loses ownership, since an honest agent does not eavesdrop on the communication between the tag and the new owner. For exclusive transfer this does not hold since the intruder might be an owner of the tag, and remain owner of the tag, violating this requirement. All previous owners have to be assumed honest, that is the intruder has never owned the tag, in order to satisfy exclusive transfer.

Another approach is to assume that at least one message can be delivered in a secure fashion, for example by transferring this message within a Faraday cage. This is sufficient since it prevents the described attacks which are based on eavesdropping on the update message. Due to this shielded transfer the tag and new owner can establish a shared secret which can be used to achieve confidentiality. In order to prove protocols, using such assumption, secure channels should be added to the formal model. This is considered as a part of future research.

Chapter 6

Conclusion

"I have quite finished, Sam," said Frodo. "The last pages are for you."
– Frodo Baggins in The Return of the King by J.R.R. Tolkien

The research for this thesis has been done in two major directions. First, the development of a verification framework for ownership and ownership transfer related requirements. Next, the design of two protocols which achieve secure ownership transfer. In this chapter we discuss our contributions to this area and provide an overview of future research.

6.1 Verification Framework

We have presented formal definitions of ownership and ownership transfer, as well as their secure variants. Together, these definitions, which have been published in [DMRV09], allow us to verify the security of ownership transfer protocols. We have demonstrated the applicability of these definitions by exhibiting attacks on secure ownership, exclusive ownership, and secure ownership transfer on a number of proposed ownership transfer protocols [FA07, JH08, Son08, YY08]. Furthermore, we sketched how correctness of the protocol proposed by Lei and Cao [LC07] can be proved.

As an application of our definitions we have formalised desynchronization resistance. We have used this formalisation to uncover a flaw in a stateful RFID authentication protocol by Song and Mitchell [SM08].

6.2 Protocol Design

We have designed two protocols using existing authentication protocols as main ingredients. The first protocol, based on the Needham-Shroeder-Lowe protocol [Low96], satisfies all of the security requirements, at the cost of a high computational complexity.

The second protocol, based on the ISO Symmetric Key Three-Pass Mutual Authentication protocol [ISO08], reduces the computational complexity, but fails to satisfy exclusive ownership. This requirement can only be achieved using public-key cryptography or by assuming that one message can be delivered without being eavesdropped on.

6.3 Future Research

Combinations While we consider a formal definition of ownership to be of independent interest, it will clearly become much more valuable when combined with existing security and privacy requirements. For instance, in a parcel delivery system, where RFID tags are attached to parcels, *non-repudiation* for obtaining ownership of RFID tags and *untraceability* of these tags by unauthorised entities become important.

Model Checking The model used in this work has been designed in such a way that the verification of our security requirements should be possible with a model checking tool. As already stated in this thesis, model checking is required in order to prove the correctness of the discussed protocols.

Protocol Design Finally, the work started in Chapter 5 can be continued to decrease the computational complexity, while trying to satisfy as many security requirements as possible. Furthermore, the research towards combinations with other requirements might lead to new challenges in protocol design.

Appendix A

Stateful Security Protocols

We give a description of security protocol syntax and semantics. The model presented has been developed by Van Deursen and included in [DMRV09]. It is based on the model for stateless protocols by Cremers and Mauw [CM05]. Their model has been extended by adding support for stateful protocols. While stateless protocols start in the same state for every execution, stateful protocols may use information from previous and parallel protocol executions. Van Deursen's description of his model has only been slightly modified to better suit this thesis.

A.1 Protocol specification

We describe a specification language for security protocols. This language is generated by the context-free grammar in Figure A.1. In this section we discuss the elements from this language.

We specify a *protocol* as a map from an n-tuple of distinct *roles* to an n-tuple of *role specifications*. A role specification consists of a declaration of the nonces and variables (defined below) used by that role and the *events* defining the messages that an honest agent sends and expects to read, when executing the role. Events can be composed in several ways. *Sequential composition*, denoted by $(\_ \cdot \_)$, specifies consecutive execution of events while *alternative composition*, denoted by $(\_ + \_)$, models the branching structure that security protocols may have. Finally *conditional branching*, denoted by $(\_ \lhd \_ \rhd \_)$, choses the left branch if the guard evaluates to true, otherwise it executes the right branch.

In order to simplify role specifications, we assume that sequential composition $(\_ \cdot \_)$ binds stronger than alternative composition $(\_ + \_)$ and alternative composition binds stronger than conditional branching $(\_ \lhd \_ \rhd \_)$. To improve readability of specifications we leave out superfluous parentheses whenever no confusion can arise.

Messages that are sent over the network are constructed by a term algebra. We define *Agent* to be the set of agent names of the agents that are allowed to execute protocols. The set of constants, *Const*, contains constants that are globally known, such as the natural numbers. The set *Nonce* contains nonces, that is, values that are freshly generated for every protocol execution. Functions that can be applied to terms are contained in the set $\mathcal{F}$.

ProtSpec	$\rightarrow$	$ProtName(RoleName)$ = RoleSpec |
		ProtSpec, ProtSpec
RoleSpec	$\rightarrow$	$(\mathcal{P}(Nonce); \mathcal{P}(Var_L); \mathcal{P}(Var_G); \mathcal{P}(\mathcal{G});$ Event$)$
Event	$\rightarrow$	(Event $\cdot$ Event) |
		(Event + Event) |
		(Event $\triangleleft$ Term = Term $\triangleright$ Event) |
		$send$(Term)[Assignment] |
		$read$(Term)[Assignment]
Assignment	$\rightarrow$	Var_G := Term |
		$\mathcal{G}(RoleName)$:= Term |
		Assignment, Assignment
Term	$\rightarrow$	$Agent$ |
		$Const$ |
		$Nonce(\theta)$ |
		$RoleName$ |
		Var_L |
		Var_G |
		$\mathcal{G}(RoleName)$ |
		$\mathcal{F}$(Term) |
		(Term,Term) |
		$\{\text{Term}\}_{Term}$

Figure A.1: Grammar for protocol specifications

We consider four sets of variables that are pairwise disjoint. The set *RoleName* contains the role names of the roles in the protocol. During protocol execution, role names are instantiated by the names of the agents executing the protocol. Local variables are variables that are instantiated during an execution of a run, but lose their value after the run finishes. They are contained in Var_L. The set Var_G contains global variables which represent the persistent knowledge of an agent. Their values are maintained across protocol runs. Global variable arrays, contained in $\mathcal{G}$, are a generalization of global variables. They group global variables in order to simplify role specifications. Encryption keys for communication with other agents can for instance be grouped in a global variable array. We use a special variable θ to denote the identifier of a run. This variable is used to disambiguate nonces of different runs. A fresh value is assigned to θ when a role is instantiated. Note that θ must not occur in any of the variable sets.

Complex terms can be constructed by pairing terms, denoted by $(\_, \_)$, encrypting a term by another term, denoted by $\{\_\}_\_$, or applying a function $f \in \mathcal{F}$ to a term, denoted by $f(\_)$.

Send and read events can be accompanied by a list of variable *assignments*. Assignments can be done to both global variables and global variable arrays. Execution of a send or read event accompanied by an assignment of variables is considered as an atomic step.

Finally a specification might contain *signals*, similar to Ryan et al. [RSG+00]. These signals are used to identify points in a protocol run which are of special interest when specifying properties. They are only used for specification and verification purposes, hence they do not appear in actual executions of a protocol.

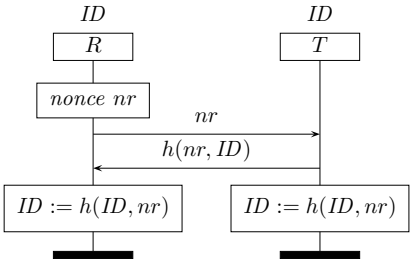

Figure A.2: Example protocol

Example A.1. Consider the tag authentication protocol in Figure A.2. The reader role R has a global variable array ID which contains an identifier ID for every tag T. The role starts by generating a nonce nr and sending it to T. Since nr is a nonce, it is indexed by the special variable θ which is instantiated immediately when the run starts. Note that at the time of sending, R does not know to whom it is sending the nonce. It then waits for a response $h(nr, ID)$ from any tag T. Upon receipt of a valid response, it updates the ID for the responding tag. The reader role specification is as follows:

$$
\begin{aligned}
example(R) = (\\
&\{nr\};\ \emptyset;\ \emptyset;\ \{ID\};\\
&send(nr(\theta)) \cdot\\
&read(h(nr(\theta), ID(T)))[ID(T) := h(ID(T), nr(\theta))]\\
)
\end{aligned}
$$

The tag role T has its identity ID as persistent knowledge. It starts its run by reading a reader challenge nr from the network. Since the reader challenge is different every run, nr is a local variable. After reading the challenge nr, it constructs a response by pairing nr and ID and applying the function h to this pair. This value $h(ID, nr)$ is sent to R as well as assigned to the global variable ID, thus performing the same update as R. The tag role specification is as follows:

$$
\begin{aligned}
example(T) = (\\
&\emptyset;\ \{nr\};\ \{ID\};\ \emptyset;\\
&read(nr) \cdot\\
&send(h(nr, ID))[ID := h(ID, nr)]\\
)
\end{aligned}
$$

A.2 Protocol execution

In this section we describe how, through instantiation of variables, an abstract role specification can be transformed into an execution by an agent. A particular execution of a protocol role by an agent is called a *run*. Furthermore, we define how the interleaved execution of a collection of runs defines the behavior of a system.

A system state $\langle A, G, SB, RB, I \rangle$ is determined by the active runs A, the global knowledge of the agents G, the send buffer SB, the read buffer RB, and the intruder's knowledge I. An active run contains a *run identifier*, the name of the *agent* executing the run, a list of remaining *events*, as well as the local variable assignment for that run. The *global knowledge* contains the global variable assignment for every agent. Since we assume communication between agents to be asynchronous, agents write messages to a *send buffer* and read messages from a *read buffer*. The *intruder knowledge* contains the set of terms that the adversary initially knows, extended with the terms learned during protocol executions.

The behavior of the system is defined as a transition relation between system states. The derivation rules, depicted in Figures A.3, A.4, and A.5, are of the form

$$\frac{C}{S \xrightarrow{l} S'},$$

expressing that a system in state S can do a transition to state S' with label l if condition C is satisfied. A state transition is the conclusion of applying one of these rules. In this way, starting from an initial state $\langle \emptyset, \emptyset, \emptyset, \emptyset, I_0 \rangle$, where I_0 denotes the initial intruder knowledge, we can derive all possible behavior of a system executing a set of protocols.

We separate the derivation rules into three categories. The agent rules, given in Section A.2.1, express under which conditions an agent may execute one of its protocol steps. Agent rules can be composed in several ways to model possible protocol flow, expressed by the composition rules given in Section A.2.2. Finally, the intruder rules, which can be found in Section A.2.3, model the capabilities of the intruder.

A.2.1 Agent rules

The rules in Figure A.3 describe the actions which an agent can perform. The *create* rule creates a run with a fresh run identifier f and adds it to the set of active runs. We use $\mathit{runids}(A)$ to denote the set of run identifiers in A. We capture the set of agents that is allowed to execute role R by $\mathit{agentsof}(R)$. This is to optimize the verification of protocols in which agents only implement a subset of the protocol roles. The *type* of an agent refers to the possibility of the agent to be active in at most one run ($\mathit{type} = 1$) or more than one run at a time ($\mathit{type} = *$). We denote the set of agents that currently have an unfinished run by $\mathit{unfinished}(A)$. The new active run is a tuple containing the run identifier f, the agent name n, the events of the role (denoted by $\mathit{eventsof}(R)$) and the initial local variable assignment. The variable assignment maps the role name to the agent name ($R \mapsto n$) and the run identifier variable to its fresh value ($\theta \mapsto f$).

$$[\mathsf{create}]\ \frac{\begin{array}{c} n \in agentsof(R) \quad f \notin runids(A) \\ type(n) = * \vee (type(n) = 1 \wedge n \notin unfinished(A)) \\ a = (f, n, eventsof(R), \{R \mapsto n, \theta \mapsto f\}) \end{array}}{\langle A, G, SB, RB, I\rangle \xrightarrow{create(f,R,n)} \langle A \cup \{a\}, G, SB, RB, I\rangle}$$

$$[\mathsf{end}]\ \frac{a = (f, n, x, \rho) \quad x \neq \epsilon \quad a' = (f, n, \bot \cdot x, \rho)}{\langle A \cup \{a\}, G, SB, RB, I\rangle \xrightarrow{end(f)} \langle A \cup \{a'\}, G, SB, RB, I\rangle}$$

$$[\mathsf{send}]\ \frac{\begin{array}{c} x \xrightarrow{send(m)[\overrightarrow{x}:=\overrightarrow{c}]/T/F/} x' \quad a = (f, n, x, \rho) \quad a' = (f, n, x', \rho) \\ G(n) = \sigma \quad \sigma' = \sigma[\overrightarrow{c}/\overrightarrow{x}] \quad SB' = SB \cup \{\sigma\rho(m)\} \\ \forall_{(v,w)\in T}\ \sigma\rho(v) = \sigma\rho(w) \quad \forall_{(v,w)\in F}\ \sigma\rho(v) \neq \sigma\rho(w) \end{array}}{\langle A \cup \{a\}, G, SB, RB, I\rangle \xrightarrow{send(f,\sigma\rho(m))} \langle A \cup \{a'\}, G[\sigma'/n], SB', RB, I\rangle}$$

$$[\mathsf{read}]\ \frac{\begin{array}{c} x \xrightarrow{read(m)[\overrightarrow{x}:=\overrightarrow{c}]/T/F/} x' \quad a = (f, n, x, \rho) \quad a' = (f, n, x', \rho) \\ G(n) = \sigma \quad \sigma' = \sigma[\overrightarrow{c}/\overrightarrow{x}] \quad Match_{\rho,\sigma}(m, m', \rho') \\ \forall_{(v,w)\in T}\ \sigma\rho(v) = \sigma\rho(w) \quad \forall_{(v,w)\in F}\ \sigma\rho(v) \neq \sigma\rho(w) \end{array}}{\langle A \cup \{a\}, G, SB, RB \cup \{m'\}, I\rangle \xrightarrow{read(f,m')} \langle A \cup \{a'\}, G[\sigma'/n], SB, RB, I\rangle}$$

Figure A.3: Agent rules

The execution state of a run can be determined by inspecting its list of events. An agent has successfully completed a run when this list is empty (denoted by ϵ). An event list which has been marked (with $\bot$), by means of the *end* rule, indicates that the run has been terminated before it was able to finish successfully. Otherwise the run is currently unfinished.

Any agent executing a *send* event, thereby changing from state x to x' (for x and x' lists of events), changes the overall system state. The sent message (obtained by applying the local variable assignment ρ and global variable assignment σ to the message) is added to the send buffer.

A send event can be accompanied by a list of global variable assignments of the form $x := c$. We denote by $\overrightarrow{x} := \overrightarrow{c}$ the simultaneous assignment of a list of variables x to a list of values c of the same length. The rule changes the current global variable assignment σ to $\sigma[\overrightarrow{c}/\overrightarrow{x}]$, where $\sigma[c/x]$ denotes the substitution σ altered such that $x \mapsto c$. When the execution of the send event is part of a (nested) conditional branching statement, a (number of) equalities (T) and/or inequalities (F) have to be fulfilled. Each of these (in)equalities must hold after substituting the local and global variables with their respective values.

An agent executing a *read* event changes the system state similar to a send event. It takes a message m' from the read buffer and matches it against the message that an agent expects to receive. It furthermore extends the local variable assignment ρ to ρ' such that any free variables in the expected message are assigned a value making $\sigma(m)$ and m' equivalent. Finally, the message m' is removed from the read buffer.

The purpose of the match predicate, used in the read rule, is to fix a minimal substitution ρ' that maps every variable in m to a ground term, such that $\sigma\rho(m) = m'$. Furthermore, the term m' is required to be readable. Formally,

$$Match_{\rho,\sigma}(m, m', \rho') \equiv \\ m' = \sigma\rho(m) \wedge dom(\rho) = vars(m) \wedge Rd(rng(\rho) \cup rng(\sigma), \rho', \sigma(m), m')$$

The readability predicate *Rd* decides whether a given term is readable. A received term m' is readable with respect to an expected term m if there is a substitution ρ that makes them syntactically equivalent. Furthermore, every subterm required to read the term must be inferable from the agent's knowledge extended with the received message. More formally, let $m, p \in Term$, $K \in \mathcal{P}(Term)$, and $\rho(m) = m'$, then

$$Rd(K, \rho', m, m') \equiv \forall_{a \sqsubseteq m}\ K \cup \{m'\} \vdash \rho(a) \vee K \cup \{m'\} \vdash \rho(a)^{-1}.$$

The subterm operator, denoted by $\sqsubseteq$, is used to decompose a term into the terms from which it was constructed. Let $t, t_1, t_2 \in Term$, then:

$$\begin{array}{lll} t \sqsubseteq t & t_1 \sqsubseteq (t_1, t_2) & t_2 \sqsubseteq (t_1, t_2) \\ t_1 \sqsubseteq \{t_1\}_{t_2} & t_2 \sqsubseteq \{t_1\}_{t_2} & t \sqsubseteq h(t) \end{array}$$

The knowledge inference operator, denoted by $\vdash$, is used to indicate which terms can be derived from a set of knowledge M. Let $t, t_1, t_2, k \in Term$ and $f \in \mathcal{F}$, then:

$$\begin{array}{rlrl} t \in M \Rightarrow & M \vdash t & M \vdash t_1 \wedge M \vdash t_2 \Rightarrow & M \vdash (t_1, t_2) \\ M \vdash t \Rightarrow & M \vdash f(t) & M \vdash t \wedge M \vdash k \Rightarrow & M \vdash \{t\}_k \\ M \vdash (t_1, t_2) \Rightarrow & M \vdash t_1 \wedge M \vdash t_2 & M \vdash \{t\}_k \wedge M \vdash k^{-1} \Rightarrow & M \vdash t \end{array}$$

A.2.2 Composition rules

The rules in Figure A.4 describe the semantics for composition of events. They are very similar to the transition rules for Basic Process Algebra [Fok00]. The main difference is the treatment of the conditional branching statement $x \triangleleft v = w \triangleright y$. Instead of requiring $v = w$ (or $v \neq w$) as a premise we add it as a proof obligation. We therefore have rules of the form

$$\frac{A}{x \xrightarrow{a/T/F} x'},$$

stating that an agent in state x can execute a and transition to x', if the premise A is satisfied. The execution of a additionally introduces the proof obligations in T (equalities) and F (inequalities).

In the following, let a be a read, send, or claim event and x and y be variables ranging over lists of events. The *exec* rule states that an event a can be successfully executed introducing no proof obligations. The *choice* rules express that in an alternative composition either of the branches can be executed. The sequential composition states that when executing $x \cdot y$, first x is executed and then y. The conditional branching statement $x \triangleleft v = w \triangleright y$ expresses that the left branch can be executed, introducing a proof obligation $v = w$, or the right branch can be executed, introducing a proof obligation $v \neq w$.

$$[\mathsf{exec}]\ \frac{}{a \xrightarrow{a/\emptyset/\emptyset} \checkmark}$$

$$[\mathsf{choice}_1]\ \frac{x \xrightarrow{a/T/F} x'}{x + y \xrightarrow{a/T/F} x'} \qquad [\mathsf{choice}_2]\ \frac{y \xrightarrow{a/T/F} y'}{x + y \xrightarrow{a/T/F} y'}$$

$$[\mathsf{seq}_1]\ \frac{x \xrightarrow{a/T/F} x'}{x \cdot y \xrightarrow{a/T/F} x' \cdot y} \qquad [\mathsf{seq}_2]\ \frac{x \to \checkmark \quad y \xrightarrow{a/T/F} y'}{x \cdot y \xrightarrow{a/T/F} y'}$$

$$[\mathsf{cond}_1]\ \frac{x \xrightarrow{a/T/F} x'}{x \lhd v = w \rhd y \xrightarrow{a/T \cup (v,w)/F} x'} \qquad [\mathsf{cond}_2]\ \frac{y \xrightarrow{a/T/F} y'}{x \lhd v = w \rhd y \xrightarrow{a/T/F \cup (v,w)} y'}$$

Figure A.4: Composition rules

A.2.3 Intruder rules

The rules in Figure A.5 describe the capabilities of the intruder. The intruder operates on the send and read buffer (SB and RB). The *deliver* rule transfers a message from the send buffer to the read buffer. If the intruder has eavesdropping capabilities he may additionally add that message to his knowledge, as stated by the *eavesdrop* rule. The *block* rule expresses that any message in the send buffer may be removed by the intruder, but the intruder still learns the message. The intruder may also be able to *inject* messages, that is, add messages he can infer from his knowledge to the read buffer.

Different adversaries can be modeled by selecting a subset of the rules in Figure A.5. An adversary with no powers is modeled by having only the deliver rule. A passive adversary can be modeled by additionally having the eavesdrop rule. The Dolev-Yao intruder [DY83], which is an adversary that essentially controls the network, is modeled by the union of the four rules.

$$[\mathsf{deliver}]\ \frac{}{\langle A, G, SB \cup \{m\}, RB, I\rangle \xrightarrow{deliver} \langle A, G, SB, RB \cup \{m\}, I\rangle}$$

$$[\mathsf{eavesdrop}]\ \frac{}{\langle A, G, SB \cup \{m\}, RB, I\rangle \xrightarrow{eavesdrop} \langle A, G, SB, RB \cup \{m\}, I \cup \{m\}\rangle}$$

$$[\mathsf{block}]\ \frac{}{\langle A, G, SB \cup \{m\}, RB, I\rangle \xrightarrow{block} \langle A, G, SB, RB, I \cup \{m\}\rangle}$$

$$[\mathsf{inject}]\ \frac{I \vdash m}{\langle A, G, SB, RB, I\rangle \xrightarrow{inject} \langle A, G, SB, RB \cup \{m\}, I\rangle}$$

Figure A.5: Intruder rules

Appendix B

Additional Attacks

We discuss additional attacks on ownership transfer protocols which were found during our research. First we have a look at the ownership transfer protocol proposed by Song [Son08] and demonstrate a desynchronisation attack which has not been mentioned before. Finally we describe another attack on the ownership transfer protocol proposed by Fouladgar and Afifi [FA07].

B.1 The Song Protocol

Song [Son08] proposes an ownership transfer protocol which consists of three protocols: a transfer protocol, an update protocol and a recovery protocol.

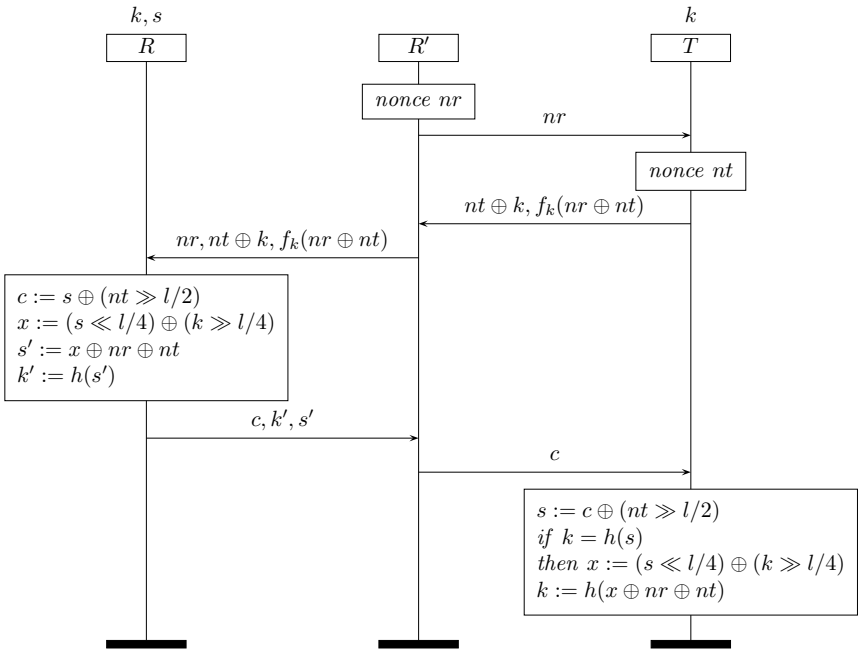

Figure B.1: Transfer protocol of RFID tag ownership transfer [Son08]

The transfer protocol, depicted in Figure B.1, is used by the new owner to authenticate the tag and retrieve the information from the previous owner. This information consists of a secret s and a key k, which is the hashed secret. The protocol is based on the authentication protocol by Song and Mitchell [SM08]. As already mentioned in Section 3.6.2, this protocol is vulnerable to tag authentication and desynchronisation attacks. These vulnerabilities have not been resolved and can thus still be exploited.

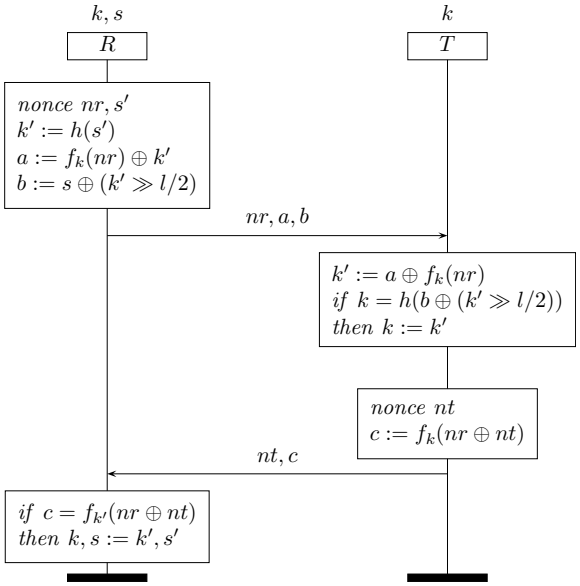

Figure B.2: Update protocol of RFID tag ownership transfer [Son08]

The update protocol, depicted in Figure B.2, is used to update the key stored on the tag to achieve exclusive ownership. First of all exclusive ownership cannot be achieved in this manner since the previous owner can listen in on the communication and derive the new keys from the exchanged messages using his knowledge of the old keys.

Furthermore, this protocol is vulnerable to a desynchronisation attack, as depicted in Figure B.3, which violates secure ownership. To achieve this the intruder intercepts the message nr, a, b sent by the reader R. This message is disturbed by adding the random value x to a. In order to get the message accepted by the tag this value, but then shifted $x \gg l/2$, also has to be added to b using exclusive-or. This message is send to the tag. After verifying the structure the tag will update its key k to $k' \oplus x$ whereas the reader expects the key to be either k, in case the update failed, or k', in case the update succeeded. Both cases do not match the actual value thus the reader has lost ownership. Furthermore, the intruder has not learned the value k', hence he does also not know the tag's key. Therefore, the tag has lost all owners and is thus desynchronised.

Finally, the recovery protocol is used to restore ownership for a previous owner. It uses the update protocol to restore an old key on the tag. Hence it suffers from the same flaws as discussed before.

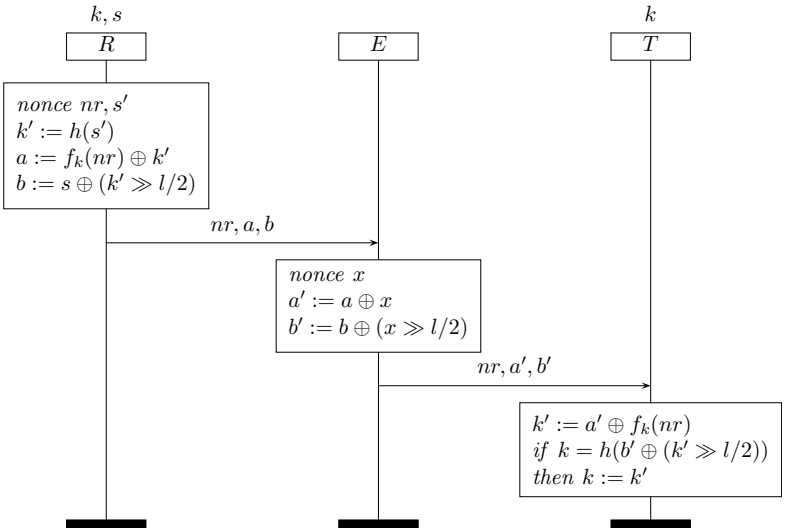

Figure B.3: Desynchronisation attack

B.2 The Fouladgar and Afifi Protocol

Fouladgar and Afifi [FA07] have proposed an ownership transfer protocol with two implementations. One based on symmetric encryption, the other one based on a cryptographic hash function. We discuss this second implementation and point out a flaw.

The protocol is based on two keys k_p and k_u. The former is used for pseudonymous identification of the tag whereas the latter is used to authen-

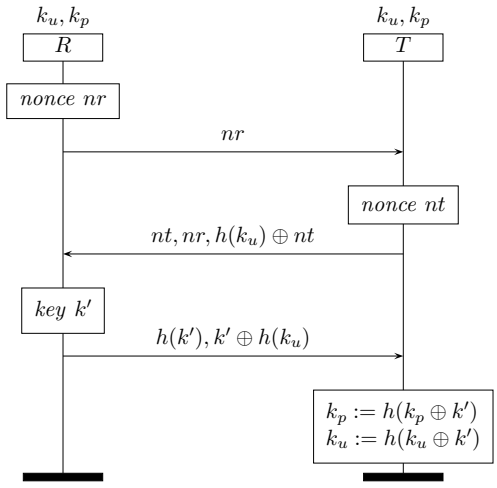

Figure B.4: Hash based implementation of A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags [FA07]

ticate the update messages. In order to transfer ownership the tag is forced into key update mode by adding an ownership transfer flag to the authentication protocol. Next the protocol from Figure B.4 is used to update the keys on the tag.

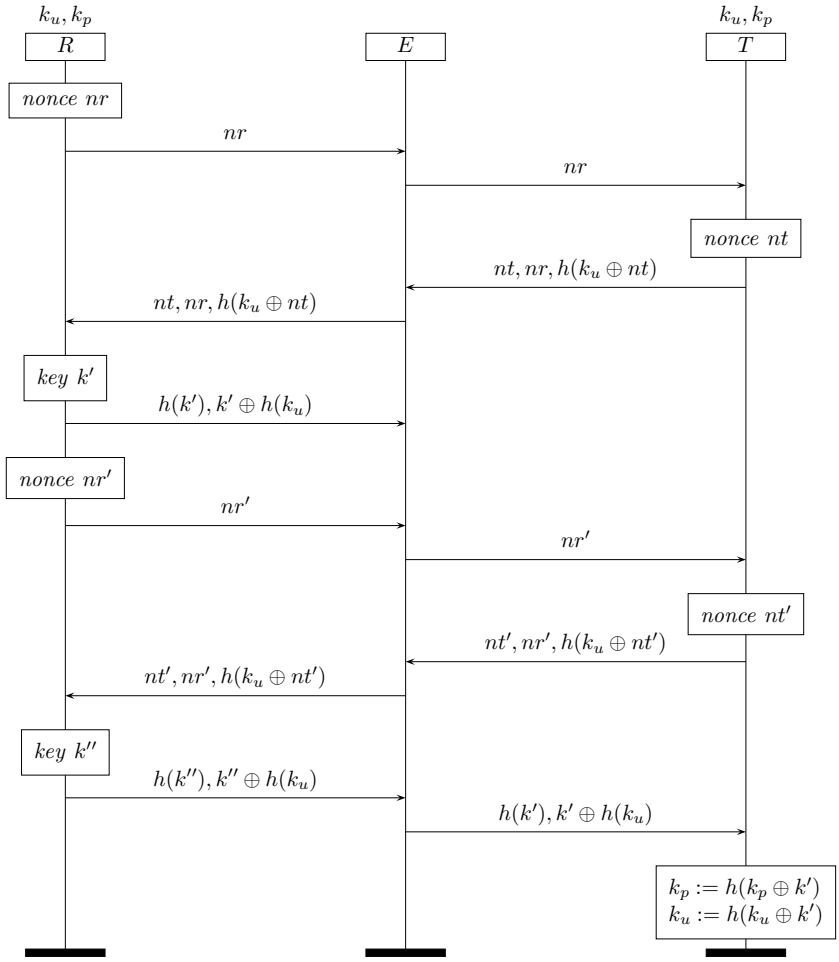

Figure B.5: Desynchronisation attack

This protocol is vulnerable to a desynchronisation attack, as depicted in Figure B.5, which violates secure ownership. To achieve this desynchronisation the intruder intercepts the final message $h(k'), k' \oplus h(k_u)$. When the reader restarts the update protocol with the tag the intruder blocks the final message and replaces it with the previously intercepted one. The tag accepts this message and updates its keys to $k'_p = h(k_p \oplus k')$ and $k'_u = h(k_u \oplus k')$. The reader, however, expects the tag to be updated using the values from the last session, hence $k'_p = h(k_p \oplus k'')$ and $k'_u = h(k_u \oplus k'')$, or that the tag still has not updated its keys. Both scenarios do not match the tag's state, such that ownership was lost. Furthermore, the intruder has not gained ownership, since he does not know the original values for k_p and k_u. Therefore, the tag has been desynchronised.

Bibliography

[BGK+07] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-key cryptography for RFID-tags. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 217–222, March 2007.

[CM05] C. Cremers and S. Mauw. Operational semantics of security protocols. In *Scenarios: Models, Algorithms and Tools (Dagstuhl 03371 post-seminar proceedings, September 7–12, 2003)*, volume 3466 of *Lecture Notes in Computer Science*, pages 66–89, August 2005.

[Dim08] T. Dimitriou. rfidDOT: RFID delegation and ownership transfer made simple. In *Proc. 4th International Conference on Security and Privacy in Communication Networks*, pages 1–8. ACM, September 2008.

[DMR08] T. van Deursen, S. Mauw, and S. Radomirović. Untraceability of RFID protocols. In *Proc. Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, volume 5019 of *Lecture Notes in Computer Science*, pages 1–15, May 2008.

[DMRV09] T. van Deursen, S. Mauw, S. Radomirović, and P. Vullers. Secure ownership and ownership transfer in RFID systems. In *Proceedings of the 14th European Symposium on Research in Computer Security, ESORICS 2009*, Lecture Notes in Computer Science. Springer, September 2009. (to appear).

[DNL99] B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using Casper and FDR. In *Workshop on Formal Methods and Security Protocols*, July 1999.

[DR08a] T. van Deursen and S. Radomirović. Attacks on RFID protocols. Cryptology ePrint Archive, Report 2008/310, July 2008. Available at `http://eprint.iacr.org/`.

[DR08b] T. van Deursen and S. Radomirovic. Security of an RFID protocol for supply chains. In *ICEBE '08: Proceedings of the 2008 IEEE International Conference on e-Business Engineering*, pages 568–573. IEEE Computer Society, October 2008.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, January 1983.

[EHJ04] S. Engberg, M. Harning, and C. Damsgaard Jensen. Zero-knowledge device authentication: Privacy & security enhanced RFID preserving business value and consumer convenience. In *Proceedings of the Second Annual Conference on Privacy, Security and Trust*, pages 89–101, October 2004.

[FA07] S. Fouladgar and H. Afifi. A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags. *Journal of Communications*, 2:6–13, November 2007.

[Fok00] W. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, February 2000.

[GJP05] S. Garfinkel, A. Juels, and R. Pappu. RFID privacy: An overview of problems and proposed solutions. *IEEE Security and Privacy*, 3(3):34–43, May 2005.

[ISO08] ISO/IEC 9798-2. Information technology – security techniques – entity authentication – part 2: Mechanisms using symmetric encipherment algorithms. Technical report, ISO, 2008.

[JH08] P. Jäppinen and H. Hämäläinen. Enhanced RFID security method with ownership transfer. In *Proc. International Conference on Computational Intelligence and Security*, pages 382–385. IEEE Computer Society, December 2008.

[KRM+07] K. Koralalage, S. Mohammed Reza, J. Miura, Y. Goto, and J. Cheng. POP method: an approach to enhance the security and privacy of RFID systems used in product lifecycle with an anonymous ownership transferring mechanism. In *Proc. ACM Symposium on Applied Computing*, pages 270–275. ACM, March 2007.

[LC07] H. Lei and T. Cao. RFID protocol enabling ownership transfer to protect against traceability and dos attacks. In *Proc. The First International Symposium on Data, Privacy, and E-Commerce*, pages 508–510. IEEE Computer Society, November 2007.

[LD07] Y. Li and X. Ding. Protecting RFID communications in supply chains. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 234–241. ACM, March 2007.

[LK06] C. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In *Proc. Conference on Information and Communications Security*, volume 4307 of *Lecture Notes in Computer Science*. Springer, December 2006.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166. Springer-Verlag, March 1996.

[LSBV08] Y. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic-curve-based security processor for RFID. *IEEE Transactions on Computers*, 57(11):1514–1527, November 2008.

[MS96] U. Maurer and P. Schmid. A calculus for security bootstrapping in distributed systems. *Journal of Computer Security*, 4(1):55–80, September 1996.

[MSW05] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *Proc. Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 276–290. Springer, August 2005.

[OTYT06] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. An efficient and secure RFID security method with ownership transfer. In *Proc. Computational Intelligence and Security*, pages 778–787. Springer-Verlag, November 2006.

[RFI03] Merloni unveils RFID appliances. RFID Journal, April 2003. Available at `http://www.rfidjournal.com/article/view/369/`.

[RGG96] E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on message sequence charts. *Computer Networks and ISDN Systems*, 28(12):1629–1641, June 1996.

[RSG+00] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols.* Addison-Wesley Professional, December 2000.

[SIS05] J. Saito, K. Imamoto, and K. Sakurai. Reassignment scheme of an RFID tag's key for owner transfer. In *Proc. Embedded and Ubiquitous Computing*, volume 3823 of *Lecture Notes in Computer Science*, pages 1303–1312. Springer, December 2005.

[SM08] B. Song and C. Mitchell. RFID authentication protocol for low-cost tags. In *Proc. First ACM Conference on Wireless Network Security*, pages 140–147. ACM, April 2008.

[Son08] B. Song. RFID tag ownership transfer. In *Proc. Workshop on RFID Security*, July 2008.

[STF05] T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC network: the potential of RFID in anti-counterfeiting. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1607–1612. ACM, March 2005.

[YY08] E. Yoon and K. Yoo. Two security problems of RFID security method with ownership transfer. In *Proc. IFIP International Conference on Network and Parallel Computing*, pages 68–73. IEEE Computer Society, October 2008.