Attacks on RFID Protocols

Ton van Deursen
ton.vandeursen@uni.lu

Saša Radomirović
sasa.radomirovic@uni.lu

August 6, 2009
Version 1.1

Abstract

This document consists of a description of attack methodologies and a collection of detailed attacks upon RFID protocols. It is meant to serve as a quick and easy reference and it will be updated as new attacks are found. Currently the only attacks on protocols shown in full detail are the authors' original attacks with references to similar attacks on other protocols.

The main security properties considered are authentication, untraceability, and desynchronization resistance.

Keywords: RFID, identification protocols, attacks.

Parts of this document have appeared in [DR08a, DR08b, DR09].

Contents

Preliminaries

Terminology

In this document, *reader* refers to the actual RFID reader as well as a potential database or server communicating with the reader, since in all protocols considered this communication takes place over a secure channel. An *agent* can be a tag or a reader, while a *role* refers to the protocol steps a tag or reader is expected to carry out. A *run* is the execution of a role by an agent. A *nonce* is a random number or a random string.

For convenience and intuition, we will refer to certain attacks on protocols as *quality-time* attacks. These are attacks in which the adversary interacts with a tag in absence of an honest or trusted RFID reader. The attacks can be carried out on tags that happen to be in the vicinity of an adversary for a short period of time or on tags the attacker is able to isolate from their environment for an extended period of time.

When we refer to the *untraceability* property of a protocol, we mean the *tag's* untraceability.

Notation

The exclusive or (*xor*) operator is a commutative, associative operator, denoted by $\oplus$. The *xor* operator has the property that equal terms cancel each other out, i.e. $(a \oplus b) \oplus a = b$ for any a and b.

We use message sequence charts, such as in Figure 1, for the description of protocols as well as attacks on protocols[1].We add textual explanations only when the message sequence chart is ambiguous or insufficient in some form.

Every message sequence chart shows the role names, framed, near the top of the chart. Above the role names, the terms known to the role 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 its role. Other conditions that need to be satisfied are shown in diamond boxes. Such conditions will include security claims made by the protocol's authors,

[1] Note that attacks can be viewed as protocols in which the intruder's role has been specified.

such as untraceability or authentication claims, which will appear typically at the bottom of the chart. There are two types of condition boxes that represent security claims. The first type is a crossed-out diamond box, representing a security claim we invalidate. Such an invalidated claim will be accompanied by an explicit attack on the security claim. The second type is a normal diamond box, representing a security claim we have not invalidated nor proven.

For example, in Figure 1, the role names are R and T, both know the secret term k, only T knows TS_{last}. The picture represents the following execution flow. R generates the timestamp TS before sending the first message. After reception of the first message, T verifies the condition $TS > TS_{last}$ before continuing its run. T generates a nonce r and sends the second message to R. The reader hashes the key k and the second part of the message (r) *and verifies that the hash is equal to the first part of the message* ($h(k, r)$). If not, the reader stops its execution, else it continues by hashing r and k and sending the third message to T. The tag *verifies that the received value matches* $h(r, k)$ and if so it sets TS_{last} to TS. The protocol has been claimed to satisfy *untraceability* of the tag role and *authentication* of the tag role towards the reader role but the latter claim can be shown to be false.

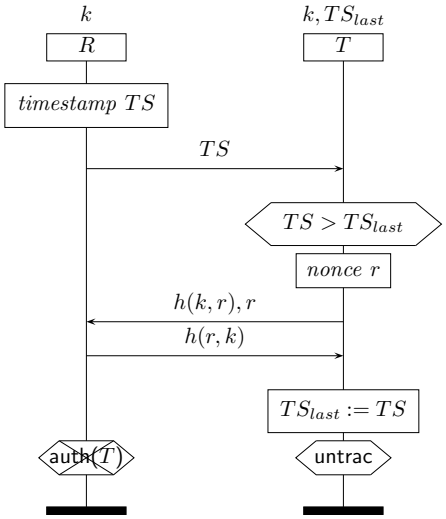

Figure 1: Example protocol

We simplify the presented protocols whenever possible by leaving out irrelevant steps, communications, and terms. The description given suffices to reconstruct the attacks on the original protocols. Furthermore, for the reader's convenience, when describing a protocol, we consistently use the notation shown in Table 1. Whenever additional functions and variables are needed we use the notation that was originally chosen by the authors of the protocol. When several runs of a protocol are shown, the terms used in the second run are primed, the terms in the third run are double primed, etc. Similarly, in protocols where

Table 1: Notation

Symbol	Meaning
A, B, R, T	agent names
h	cryptographic hash function
,	concatenation
$\oplus$	exclusive or operator
$ID,\ k, k_0, k_1, \ldots$	shared secret between reader and tag
$r, r_0, r_1, r_2 \ldots$	random numbers

reader or tag update stored variables, the variables whose values are being updated are shown with a prime after the update.

Convention

To simplify references, we name the presented protocols with the citation key which consists of the first letters of the last names of the protocol's authors and the year of publication appended. Thus for instance, the Diffie-Hellman key exchange protocol would be named [DH76].

Security Properties

In terms of Lowe's authentication hierarchy [Low97], we consider *recent aliveness* to be the most appropriate authentication requirement for RFID protocols. Recent aliveness captures the fact that the tag needs to have generated a message as a consequence of a reader's query. More formally, a protocol guarantees to an agent a in role A that any corresponding agent b in role B has been recently alive, iff whenever a completes a run, there has been an event of b during that run. What recent aliveness does not capture, is the requirement that the tag needs to be in the vicinity of the reader at the time of the communication. We do not consider this issue in the present paper.

We consider the notion of *untraceability* as defined in [DMR08] which captures the intuitive notion that a tag is untraceable if an adversary cannot tell whether he has seen the same tag twice or two different tags.

The third security property we investigate is *desynchronization resistance* as defined in [DMRV09]. It ensures that the system in which the protocol runs will never evolve into a state where there is nobody who can successfully execute the protocol with a tag. In this paper, we restrict ourselves to the three aforementioned properties.

Other properties which are relevant to the RFID setting are *distance bounding* and *scalability*. A distance-bounding protocol ensures that the tag is in close proximity to the reader when communicating. In particular, distance-bounding protocols prevent man-in-the-middle attacks on the protocols. Scalability ensures that the reader can efficiently authenticate any tag and is is therefore only tangentially related to the security of an RFID protocol.

Intruder Model

We perform our security analyses in the Dolev–Yao intruder model [DY83]. In this model, the adversary may eavesdrop on any message exchanged between tag and reader, modify or block any message sent from tag to reader or vice versa, and may inject his own messages making them look like they were sent by tag or reader. We additionally assume that the adversary can observe whether an agent successfully completed its run. This is in line with adversary models designed for RFID protocol analysis, such as Avoine [Avo05], Juels and Weis [JW07], Vaudenay [Vau07], Damgård and Pedersen [DP08], and Paise and Vaudenay [PV08].

Attacking RFID Protocols

From the general description of the adversary's capabilities we derive three types of attack strategies for the adversary which are meant to enhance the intuition for the attacks and simplify their description.

The simplest strategy is to eavesdrop on messages transmitted between tag and reader. The adversary may then deduce information and combine messages to later impersonate or trace a tag.

The second strategy is to spend some "quality time" with a tag. In these attacks, to which we refer as *quality-time* attacks, the adversary interacts with a tag in absence of an honest or trusted RFID reader. The point of such an attack is to send carefully designed challenges to the tag in order to obtain information which can later be used to impersonate a reader or the tag, trace the tag, or attack any other security requirement of a protocol.

Quality-time attacks are facilitated by the mobile and wireless nature of RFID tags. The attacks can be carried out on tags that happen to be in the vicinity of an adversary for a short period of time or on tags the attacker is able to isolate from their environment for an extended period of time. In case tags and readers share secret keys, a quality time attack might be mounted on the reader as well.

The third strategy involves modifying messages transmitted between a reader and a tag. This attack works best when the adversary has simultaneous access to a legitimate reader and a tag which is not in the reader's vicinity. The adversary may modify transmitted messages and then observe the evolution of the communication session.

For each of the three strategies, the feasibility of an attack depends on many factors. In general, it is obvious that the fewer interactions an adversary needs to engage in, eavesdrop on, or modify, the more feasible the attack becomes.

By applying these strategies to several proposed RFID protocols we have identified three types of attacks, which we discuss in subsequent sections. These types of attacks are what we call *algebraic replay attacks* targeting the challenge-response mechanism in authentication protocols, *attribute acquisition attacks* on untraceability of tags, and cryptanalytic attacks on secrecy of keys and tag

identities.

Algebraic Replay Attacks

A common way to authenticate RFID tags is by means of the following challenge-response mechanism. The RFID reader challenges the tag with a nonce r_1 to which the tag replies with a term derived from the nonce r_1, some information s identifying the tag, and potentially a nonce r_2 generated by the tag. If present, the nonce r_2 serves as the tag's challenge to the reader in mutual authentication protocols or as a "blinding term" to achieve tag untraceability. We can thus represent the tag's reply to the reader's challenge as the term $r_2, g(r_1, r_2, s)$ with the understanding that r_2 may be constant or empty. The reader verifies the authenticity by applying the inverse of the function g to the term and checking whether the response contains r_1 and a valid s. If g is a one-way function then the reader verifies the authenticity of the tag by computing the function $g(r_1, r_2, s)$ and comparing it to the received value. The reader can compute this function, since it generated the value r_1 itself, the value r_2 is supplied by the tag, and the reader has a database with values of s for every tag it may authenticate.

We now argue that the following two properties are necessary in order for the challenge-response mechanism to guarantee recent aliveness of the tag.

Freshness For fixed r_2 and s the range of the function $r_1 \to g(r_1, r_2, s)$ must be large. More precisely, given r_2, s, the adversary's advantage in guessing $g(r_1, r_2, s)$ correctly for an unknown, randomly chosen r_1 must be negligible.

ARR Let $O_s(x)$ be an oracle which upon input x randomly chooses y and returns y and $g(x, y, s)$. If s is unknown, then given access to a polynomial number of queries $O_s(x_1), \ldots, O_s(x_l)$ to the oracle, it is infeasible to compute $g(r_1, r_2, s)$ for a given $r_1 \notin \{x_1, \ldots, x_l\}$ and any r_2.

If the freshness property is satisfied, then as stated, the probability of the adversary guessing $g(r_1, r_2, s)$ is negligible. Thus with overwhelming probability, a response $r_2, g(r_1, r_2, s)$, to the reader's challenge r_1 must have been generated *after* the challenge was sent. This property is obviously necessary for recent aliveness and in particular excludes classic replay attacks.

The ARR (algebraic replay resistance) property guarantees that there is no efficient algorithm to compute a response $r_2, g(r_1, r_2, s)$ to the challenge r_1 even after having observed previous challenge-response pairs. Clearly, an attacker's ability to compute such a response violates recent aliveness and this property is thus necessary for recent aliveness. Such an attack generalizes replay attacks in that instead of merely replaying previously observed information, the attacker combines previously obtained challenge-response pairs to compute the response to a fresh challenge. Hence, we refer to attacks on challenge-response authentication protocols exploiting the lack of the ARR property as *algebraic replay attacks*.

It is obvious that for a function $g(r_1, r_2, s)$ to have the ARR property, it must preserve the secrecy of s. Indeed, cryptographic hash functions are frequently used for the type of challenge-response mechanism considered here. Since the collision resistance property of cryptographic hash functions does not seem necessary for the challenge-response mechanism, the question arises whether all one-way functions satisfy the ARR property and the answer is negative. It is certainly false for all homomorphic one-way functions. Consider, for instance, the Rabin function, defined by $x \to x^2 \mod N$ for certain composite integers N. If $(r_1, r_2, s) \to g(r_1, r_2, s) = (r_1 r_2 s)^2 \mod N$ is a Rabin function, then given only one challenge-response pair, $r_1, g(r_1, r_2, s)$ it is easy to compute responses for any challenge r_1', since $g(r_1', r_2, s) = g(r_1, r_2, s) \cdot (r_1'/r_1)^2$.

Furthermore, even non-homomorphic one-way functions will in general not have the ARR property if their *argument* has algebraic properties. As demonstrated in the examples below, there are several protocols that fail to achieve recent aliveness for this very reason. In these protocols the challenge-response construction can typically be represented as $g(r_1, r_2, s) = f(r_1 \circ r_2, s)$, where f is a (non-homomorphic) cryptographic hash function and $\circ$ denotes an operator with the following algebraic property. Given a, b, and c, it is easy to find d with $a \circ b = c \circ d$. This construction clearly does not have the ARR property, regardless of the properties of f. The algebraic replay attack on such a protocol works as follows. An adversary observing one execution of the protocol learns r_1, r_2, and $f(r_1 \circ r_2, s)$. When challenged with r_1', the adversary finds r_2' such that $r_1 \circ r_2 = r_1' \circ r_2'$ and replies with $r_2', f(r_1 \circ r_2, s)$. The attack succeeds because $f(r_1 \circ r_2, s) = f(r_1' \circ r_2', s)$.

Examples of operators $\circ$ for which this type of attack succeeds are *xor*, modular addition, and any associative operator for which it is easy to compute left inverses.

Examples

The protocols by Chien and Huang [CH07], Kim et al. [KCLL06], Lee et al. [LAK06], and Song and Mitchell [SM08], shown in sections 1, 5, 7, 14, respectively, are vulnerable to this type of attack. This is due to the fact that they employ a hash-like function or a cryptographic hash function composed with *xor* which fits into the challenge-response construction with the function $f(r_1 \circ r_2, s)$ shown above.

Attacks which we classify as algebraic replay attacks have also been described by Peris-Lopez et al. [PLHCETR07, §4.2] and Bringer et al. [BCI08]:

1. Chien and Chen [CC07] implement the challenge-response mechanism by composing the cyclic redundancy check (CRC) function with *xor*. To a challenge r_1, the tag responds with $r_2, CRC(EPC, r_1, r_2) \oplus k$, where EPC is a constant representing the identity of the tag. The attack on this protocol has been first reported by Peris-Lopez et al. [PLHCETR07, §4.2]. It uses the fact that CRC is a homomorphism, i.e. $CRC(a) \oplus CRC(b) = CRC(a \oplus b)$.

To attack the protocol, the adversary observes one protocol execution. When challenged with r_1' the adversary computes the *xor* of the observed response $CRC(EPC, r_1, r_2) \oplus k$ with

$$CRC(\mathbf{0}_{EPC}, r_1, \mathbf{0}_{r_2}) \oplus CRC(\mathbf{0}_{EPC}, r_1', \mathbf{0}_{r_2}).$$

The terms $\mathbf{0}_{EPC}$ and $\mathbf{0}_{r_2}$ are 0-bit strings of length equal to the length of EPC and r_2, respectively. Because CRC is a homomorphism, the computation will result in a correct response $CRC(EPC, r_1', r_2)$ to the challenge r_1'.

2. The protocol proposed by Lee et al. [LBV08], described in detail in Section 9, is vulnerable to an algebraic replay attack in which the adversary needs to observe three protocol executions or perform a quality-time attack consisting of three queries. The algebraic replay attack can then be executed by solving a small system of equations yielding a constant particular to the tag. While this constant does not reveal the tag's secret information, it can still be used to compute the correct response to a reader's challenge. This attack has been first described by Bringer et al. [BCI08].

Attribute Acquisition Attacks

A simple, necessary condition for tag untraceability is that an adversary, which has observed a particular tag once, must not be able to recognize the tag as being the same tag in the future. To make this more precise, we call a term, which the adversary can derive from one or more runs of a tag and which identifies the tag to the adversary, a *unique attribute* of the tag. The necessary condition for a tag to be untraceable then is that the adversary must not be able to derive a unique attribute for the tag. Should the adversary be able to compute a unique attribute, then we refer to the adversary's steps to arrive at such a term as the *attribute acquisition attack*.

A simple unique attribute can be found in protocols where the tag's answer to a challenge c is merely a function $f(c, k)$ of the challenge and a secret (or collection of secrets) k and does not involve any nonce created by the tag. In this case, c is under the adversary's control, k is unique to the tag, and the adversary learns $f(c, k)$ after one round of communication with the tag. Thus for constant c chosen by the adversary, $f(c, k)$ is a unique attribute of the tag whose secret is k.

To prevent long-term traceability in protocols that employ the challenge-response mechanism described, the tag typically updates its secret k at the end of a run. The secret k must therefore also be updated by the reader and in order to avoid desynchronization attacks, the tag needs to authenticate the communicating reader before updating k. Yet, a tag following such a protocol can still be traced by an adversary between two updates by querying the tag and then aborting the protocol. Furthermore, if the update of the secret k at the end of the protocol involves operators with algebraic properties, it is frequently possible for the adversary to compute a unique attribute for the tag which will

be valid after the update. References to such protocols are given in the examples section below.

To find unique attributes in general, consider a given RFID protocol in a formal trace model such as the one proposed by Cremers and Mauw [CM05], extended by Van Deursen et al. [DMRV09], or the strand spaces model of Thayer Fàbrega et al. [THG98]. Then the unique attribute for the tag role can be obtained, if it exists, by computing the intersection of the adversary's knowledge with the set of terms which can be constructed from constants that are unique to the tag and terms that are under the adversary's control. Such a term can be found effectively, provided that the intersection is non-empty.

To find a term in the intersection for the special class of challenge-response protocols in which the tag includes a fresh nonce r in its reply $f(c, k, r)$ to a challenge c, the adversary needs to find challenges $c_1, \ldots, c_l$ and an efficiently computable function $g(x_1, \ldots, x_l)$, such that

$$g(f(c_1, k, r_1), \ldots, f(c_l, k, r_l)) = \tilde{g}(c_1, \ldots, c_l, k)$$

does not depend on the tag's nonces $r_1, \ldots, r_l$. In this case $\tilde{g}(c_1, \ldots, c_l, k)$ is the unique attribute. The attribute acquisition problem displayed in this form is more amenable to solutions by algebraic methods, as the following examples show.

Examples

1. A simple attribute acquisition attack exists on the protocol proposed by Kim et al. [KCL07], shown in Section 4. In this protocol, the tag's response can be represented by $f(c, k, r) = k_1 \oplus r, h(c, k_2) \oplus r$, where $k = k_1, k_2$ is the tag's secret, c the reader's challenge and r the tag's nonce. To find a unique attribute, the attacker challenges the tag with a constant c_1 and computes the unique attribute by taking the *xor* of the two terms in the response: $k_1 \oplus r \oplus h(c_1, k_2) \oplus r = k_1 \oplus h(c_1, k_2) = \tilde{g}(c_1, k)$.

2. The protocols by Li and Ding [LD07], Osaka et al. [OTYT06], and Yang et al. [YPL+05], shown in Sections 10, 11, 15, respectively, are stateful protocols that update the shared secrets between reader and tag at the end of a successful protocol execution. The updates take the old secret and a fresh value exchanged in the protocol execution, and apply an operator with algebraic properties to obtain the new secret. By observing the messages exchanged in a protocol execution, the attacker can fabricate a challenge to which the tag will respond with the same term: the unique attribute. In other words, the attacker uses his knowledge to "undo" the update of the tag. In the simplest of these, the protocol by Osaka et al. [OTYT06], the reader's challenge is c, the tag's response is $f(c, k) = h(k \oplus c)$, where k is the tag's secret. The tag updates its secret by computing the *xor* of it with a third message r it receives from the reader. Disregarding other flaws this protocol suffers from, the attribute acquisition attack consists in challenging the tag the first time with a constant c_1. After an update

with message r the tag is challenged with $c_1 \oplus r$. After the next update with message r', the tag is challenged with $c_1 \oplus r \oplus r'$ and so forth. The tag's response to these challenges is each time $h(k \oplus c_1)$.

3. A more challenging example is the authentication protocol proposed by Lee et al. [LBV08] and shown in Section 9. The protocol is based on a fixed, system-wide elliptic curve over a finite field. The points P, $Y = yP$, x_1P, x_2P on the elliptic curve are publicly known, the scalar y is only known to the reader, and the scalars x_1, x_2 are unique to each tag and only known to the tag. The elliptic curve is assumed to have been chosen such that the computational Diffie-Hellman problem is hard, that is, given only the points xP, yP, and P on the elliptic curve, it is hard to compute xyP.

   In the protocol, the reader challenges the tag with a random number $r_2 \neq 0$ to which the tag responds with two points $T_1 = r_1P$, $T_2 = (r_1 + x_1)Y$ on the elliptic curve and a scalar $v = r_1x_1 + r_2x_2$. Using this information, the reader can infer the tag's identity.

   Thus, this protocol, too, is a challenge-response protocol with challenge r_2 and a response that can be written as $f(r_2, k, r_1) = r_1P, (r_1+x_1)yP, r_1x_1+r_2x_2$, where $k = x_1, x_2$. The points P and yP are constant. To find a unique attribute, the adversary needs to find challenge terms $c_1, \ldots, c_l$ and functions $g, \tilde{g}$ such that $g(f(c_1, k, r_1), \ldots, f(c_l, k, r_1^{(l)})) = \tilde{g}(c_1, \ldots, c_l, k)$, where $\tilde{g}$ does not depend on the tag's random numbers $r_1, \ldots, r_1^{(l)}$.

   If we write $f(c, k, r_1) = T_1, T_2, v$ as in the protocol specification, and recall that primes indicate terms transmitted in the second run, then

$$g(f(c, k, r_1), f(c, k, r_1')) = \frac{T_1 - T_1'}{v - v'} = x_1^{-1}P$$

   depends only on the first part of the secret $k = x_1, x_2$. Thus $\tilde{g}(k) = x_1^{-1}P$ is a unique attribute.

   From the definition of the function g, it is now easy to obtain the attribute acquisition attack. By carrying out a quality-time attack, the adversary challenges the tag twice with the same value c. The information received from the tag in the two runs can be used to compute the term $x_1^{-1}P$ as follows. Observe that $v - v' = (r_1 - r_1')x_1$ and $T_1 - T_1' = (r_1 - r_1')P$, thus, multiplying $T_1 - T_1'$ with the inverse of $v - v'$ modulo the order of the elliptic curve, the attacker obtains $x_1^{-1}P$.

   Note that after executing this quality-time attack, it suffices for the adversary to challenge any given tag only once with the previously used value c to determine whether the presented tag is equal to the tag identified by $x_1^{-1}P$.

   A similar attack on untraceability of the protocol was independently found by Bringer et al. [BCI08]. The authors observe that for any two protocol executions, the following equations hold:

$$r_2 v' - r_2' v = (r_2 r_1' - r_2' r_1) x_1$$
$$r_2 T_1' - r_2' T_1 = (r_2 r_1' - r_2' r_1) P$$

The attacker may then combine these two equations to obtain $x_1^{-1} P$ and proceed as described above.

Cryptanalytic Attacks

The authentication and untraceability properties of RFID protocols often rely on the secrecy of shared keys. In some cases, revealing parts of a secret key may already be enough to trace the tag. If sufficiently many bits of a key can be revealed, brute-forcing the remaining bits may become feasible. Formal methods approaches typically do not consider attacks in which an adversary may learn just a few bits of a key, since keys are modeled as atomic terms.

If we assume that operators with algebraic properties are applied to terms sent back and forth between a reader and a tag, then a natural point of attack is to set up equations involving the terms on whose secrecy a protocol depends. Such equations may be obtained by observing several protocol runs, but also by selectively modifying parts of messages. In other words, one may attempt to apply any cryptanalytic method known to mankind. While this is hardly an original strategy, it turns out to be quite successful in the domain of RFID protocols. One reason for this is the popularity of simple operators with algebraic properties. The other reason is due to the simple structure of typical RFID protocols. The reader challenges the tag with a nonce r to which the tag responds with a message involving that nonce and a secret k. This leads to a function $r \mapsto f(k, r)$ which can be compared to a cipher $m \mapsto C(k, m)$ or keyed hash function $x \mapsto h(k, x)$. The tag's response can further be analyzed by forwarding a modified version of it to the reader and checking the reader's response. For RFID protocols with three or more messages, a tag-generated nonce, may frequently be considered as a known plaintext. Finally, stateful RFID protocols, i.e. RFID protocols in which the tag upon successful completion of the protocol updates its secret ID or cryptographic key, can be analyzed by taking advantage of algebraic relations between previous and future ID's or keys.

Examples

The protocol [KN05], shown in Section 6, is vulnerable to cryptanalytic attacks. There are also several examples of cryptanalytic attacks in the literature:

- In the HB$^+$ protocol of Juels and Weis [JW05], tags use the binary inner product and *xor*operator to hide their secret keys while proving knowledge of it. The attack by Gilbert et al. [GRS05] breaks secrecy of a tag's key by first modifying the messages exchanged between reader and tag, then observing the reader's behavior, and finally using the observed information to set up and solve a system of linear equations.

- Van Deursen et al. [DMR08] use information obtained through eavesdropping on executions of the Di Pietro and Molva protocol [DM07] to expose two thirds of the bits of a tag's secret key. In the protocol execution, bits of the tag's secret key are combined with random nonces using *xor* and logical *and* and *or* operators and then sent from the tag to the reader. The attack is carried out by solving a system of linear equations derived from the observed messages which yields two thirds of the secret key's bits. This is enough to break untraceability. It furthermore permits a brute force attack on the remaining bits in order to break authentication. A simpler, but less efficient attack is shown in Section 2.

- In the protocols of Peris-Lopez et al. [PLCETR06c, PLCETR06a, PLCETR06b], logical *and* and *or* operators are used in addition to *xor* and modular arithmetic leading to information leaks exploited by Alomair et al. [ALP07] and Li and Wang [LW07].

- Vajda and Buttyán have proposed several lightweight authentication protocols in [VB03]. Their first protocol uses *xor* and bit permutations to update keys shared between reader and tag. The attack of Alomair et al. [ALP07] correlates keys across updates thereby breaking authentication. Vajda and Buttyán's second protocol is vulnerable to an active attack in which the adversary recovers the shared secret by querying the tag with a challenge of his choice and analyzing the response.

1 [CH07]

1.1 Description

The reader R and tag T share secrets k and ID. The reader starts by sending a random bit string r_1. The tag generates a random string r_2 and hashes the *xor* of r_1, r_2, and the secret k. This hash and ID are used as input for a function in which the ID is rotated by a value depending on the hash. The tag computes the *xor* of the rotated ID and the hash, before sending the left half of the resulting bits and r_2 to the reader. The reader performs the same operations on every pair of ID and k until it finds the corresponding tag. It then sends the right half of the corresponding bits to the tag.

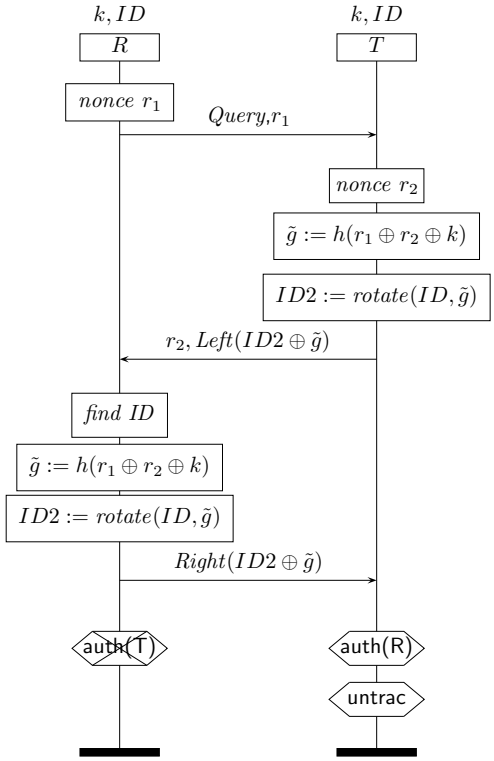

Figure 2: The protocol

1.2 Claimed Attacks

1.2.1 Tag authentication

To impersonate a tag, it suffices to notice that the tag's response to the reader's challenge only depends on $r_1 \oplus r_2$ and a shared secret. The adversary can

challenge a tag with any r_1 to obtain a valid combination of $r_1, r_2, Left(ID2 \oplus \tilde{g})$. This information suffices for the adversary to be able to respond to any future challenge r_1' received from a reader. When challenged, the adversary sets $r_2' = r_1' \oplus r_1 \oplus r_2$ and sends $r_2', Left(ID2 \oplus \tilde{g})$.

1.3 Related Protocols

We have found the same attack on the protocols [LAK06, KCLL06, SM08].

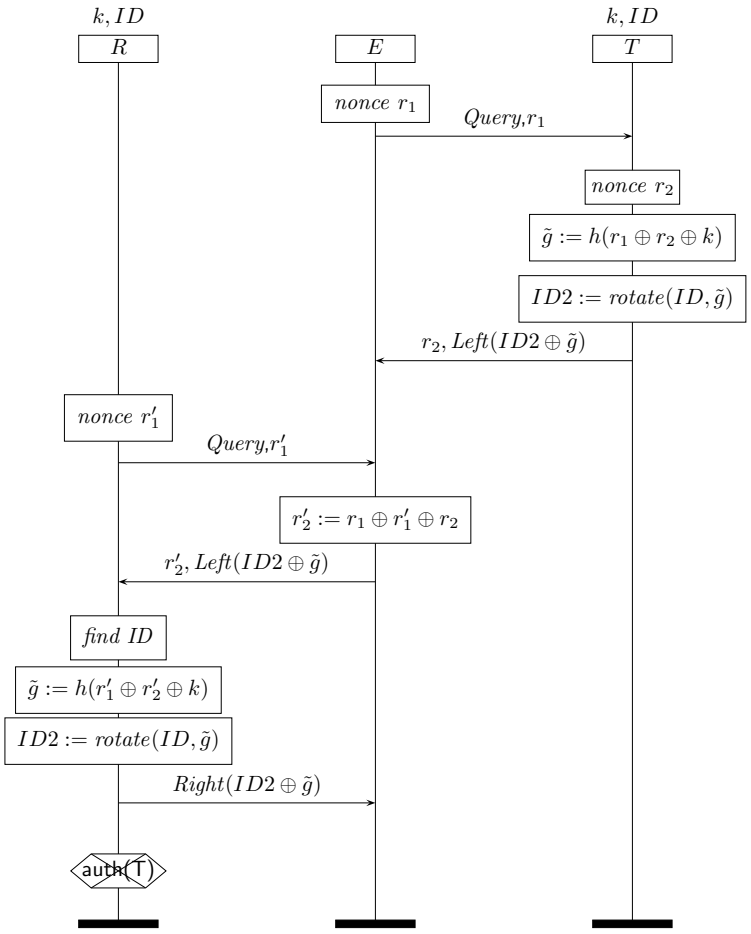

Figure 3: Attack on tag authentication

2 [DM07]

2.1 Description

This is an authentication protocol which not only aims to keep tags untraceable, but also to limit the damage a compromised reader can cause.

In the protocol, depicted in Figure 4, the function $DPM(x)$ is defined as the parity of majority functions of consecutive bit-triplets of x. The size of its output is therefore one bit. The protocol begins with the reader sending its name and a nonce r_0 to the tag. The tag replies with the message $\alpha_1, \ldots, \alpha_q, V, \omega$, where $\alpha_i = k \oplus r_i$ for randomly chosen r_i (a bit-string of length ℓ, $\ell = 117$ suggested by authors), the i-th bit of V (a bit string of length q) is $DPM(r_i)$, and $\omega = h(k, r_0, r_1, k)$. The reader has a database of all tags' keys it is authorized to identify. The reader can find a particular tag's key k with the help of the vectors α_i and values $DPM(r_i)$ by going through all the keys in its database and iteratively excluding the impossible ones, namely those for which $DPM(k \oplus \alpha_i) \neq DPM(r_i)$. It is expected that each α_i reduces the number of possible keys by approximately one half. At last, the reader uses ω to uniquely identify the correct key and authenticate the tag. The last message of the protocol allows the tag to authenticate the reader.

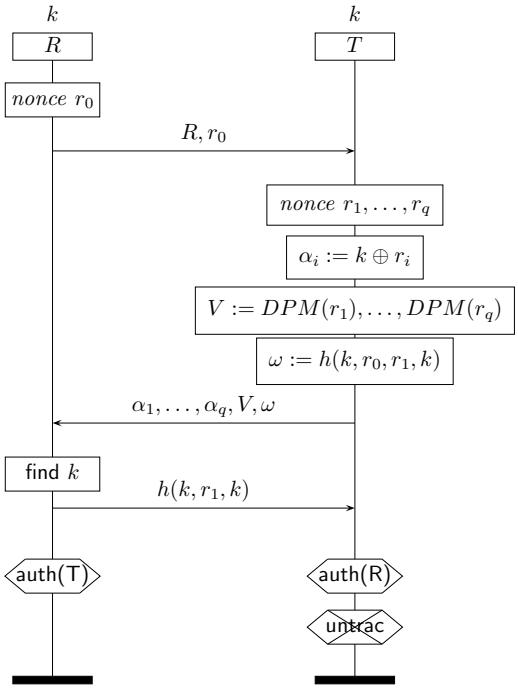

Figure 4: The protocol

2.2 Claimed Attacks

2.2.1 Authentication and Untraceability

In the following we show that over several rund, the protocol leaks $\frac{2\ell}{3}$ bits of k. This allows an attacker to brute-force the remaining bits of k for the suggested parameter $\ell = 117$.

Let $x = x_1 x_2 \cdots x_\ell$ be a bit string of length ℓ, for some positive integer ℓ divisible by three. Then $DPM(x) = M(x_1, x_2, x_3) \oplus \cdots \oplus M(x_{\ell-2}, x_{\ell-1}, x_\ell)$, where $M(a, b, c)$ is the majority function on three bits. Let $\bar{x_i}$ denote the complement of the bit x_i. It is easy to see that $M(\bar{x}_1, x_2, x_3) = M(x_1, x_2, x_3)$ if and only if $x_2 = x_3$. Analogous equations hold for the complements of x_2 and x_3. It follows that

$$DPM(\bar{x}_1, x_2, x_3, \ldots) = DPM(x_1, x_2, x_3, \ldots) \Leftrightarrow x_2 = x_3, \tag{1}$$

again with analogous equations for any other bit of x.

The adversary can take advantage of the property (1) as follows. Suppose the adversary intercepts the tag's message, flips the first bit of $\alpha_2 = r_2 \oplus k$ to obtain $\tilde{\alpha}_2$ and forwards the modified message to the reader. If the second and third bit of r_2 are equal, then $DPM(k \oplus \tilde{\alpha}_2) = DPM(k \oplus \alpha_2) = DPM(r_2)$. In this case, the reader will still be able to find the correct key k and answer the tag with the third message of the protocol. However, if the second and third bit of r_2 are not equal, then $DPM(k \oplus \tilde{\alpha}_2) \neq DPM(r_2)$ and the reader will remove the key k from the list of possible keys. No other key will pass the verification with ω, thus the reader will not answer with the third message. The adversary can therefore distinguish the two cases.

It follows that by selectively flipping bits of α_2 an adversary may, after several protocol executions, determine for each consecutive bit triplet of k which bits are equal to each other. In other words, the adversary may determine the bits of k up to complements of consecutive bit-triplets.

This information can be used to reduce the complexity of computing all bits of k to a brute force search of a space whose size is the cubic root of the full key space. For the parameters of the system suggested by Di Pietro and Molva, this brute force search becomes feasible (2^{39} keys). The knowledge of the secret key k then allows the attacker to also impersonate the tag to the reader, thus breaking the authentication claim of the protocol. By sufficiently increasing the key length, however, this attack becomes infeasible.

To break untraceability, the brute force search is not necessary. The probability that two keys are equal up to complements of consecutive bit-triplets is vanishingly small [DMR08]. Increasing the key length does not prevent this attack.

The attack outlined above is not efficient. In [DMR08] we describe an efficient quality-time attack on this protocol which reveals the same information about k as the attack described above.

2.3 Related Protocols

The presented attack is similar to the active attack on the HB^+ protocol [JW05] discovered by [GRS05] in that it exploits an algebraic property by modifying messages and observing the reader's behavior.

3 [HMNB07a]

3.1 Description

The protocol starts with the reader querying the tag with a nonce r_1. The response of the tag depends on the value of a state variable S. In case the previous run ended successfully the value of S is 0 and the tag will respond with $h(ID)$. In case it did not end successfully the value of S is 1 and the tag will respond with $h(ID, r_2, r_1)$. In either case, the tag will set its S to 1. The reader will authenticate the tag if the response is equal to HID, $h(ID, r_2, r_1)$ or $h(PID, r_2, r_1)$ for any stored value of HID, ID or PID. The reader will then update the information for the particular tag according to Table 2. The reader then sends $h(PID, r_2)$ to the tag, after which the tag replaces its ID by $h(PID, r_1)$ and sets S to 0. The protocol is depicted in Figure 5.

Table 2: Reader's verification and update procedure

Tag response	Reader action
$h(ID), r_2$	$ID' := h(ID, r_1)$; $HID' := h(ID)$; $PID' := ID$;
$h(ID, r_2, r_1), r_2$	$ID' := h(ID, r_1)$; $HID' := h(ID)$; $PID' := ID$;
$h(PID, r_2, r_1), r_2$	$ID' := h(PID, r_1)$; $HID' := h(ID)$; $PID' := PID$;
other	reject tag

3.2 Claimed Attacks

3.2.1 Tag authentication

Note that if no messages are blocked or lost, the tag always responds with $h(ID)$ allowing for an efficient lookup by the reader. An attacker can thus impersonate any tag which is in state 0 by sending a query to it and replaying the tag's response before the tag has been queried by an authorized reader. The attack is depicted in Figure 6.

3.2.2 Untraceability

The tag's response depends on the value of S, i.e. the state the tag is in. If $S = 0$ the tag responds with $h(ID), r_2$ and otherwise the tag responds with $h(ID, r_1, r_2)$. Because the attacker does not know ID, he can not conclude from the response in which state the tag is. However, the attacker may use the fact that if the tag is in state 0, changing r_2 does not result in a rejection of the response by the reader. If the tag is in state 1, changing r_2 would lead to a rejection of the response and a termination of the execution of the reader.

3.2.3 Desynchronization resistance

Any tag that is in state $S = 0$ can be desynchronized from a reader by a man-in-the-middle attack. In a communication between the reader and a tag, the adversary intercepts and modifies the reader's challenge r_1 to any value $r_1' \neq r_1$. The adversary then sends the modified value to the tag and forwards all other messages between reader and tag without modification. Since in the case $S = 0$ the reader does not verify that the tag received the correct value r_1, the adversary's modification goes by unnoticed. Thus, at the end of the protocol execution, reader and tag update ID to different values. The reader stores $h(ID, r_1)$, while the tag stores $h(ID, r_1')$. Therefore, the reader and tag will be in a *desynchronized* state and future authentication of the tag becomes impossible. The attack is depicted in Figure 7.

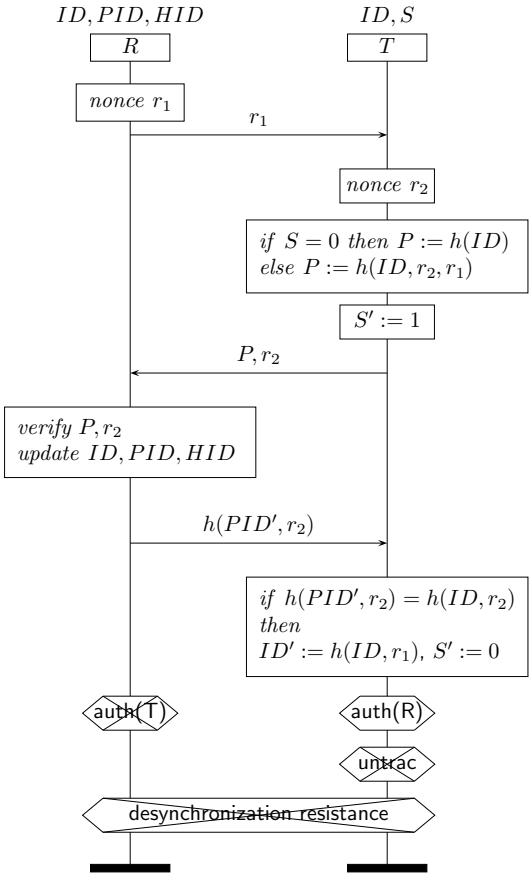

Figure 5: The protocol

3.3 Related Protocols

The protocols in [LY07c, LY07a, LY07b, HM04] are challenge-response-based protocols with a similar authentication flaw.

A similar untraceability flaw in [HM04] was found by [Avo05]. There a *quality time* attack is used to increase a tag's internal counter to an abnormal level in order to recognize the tag later.

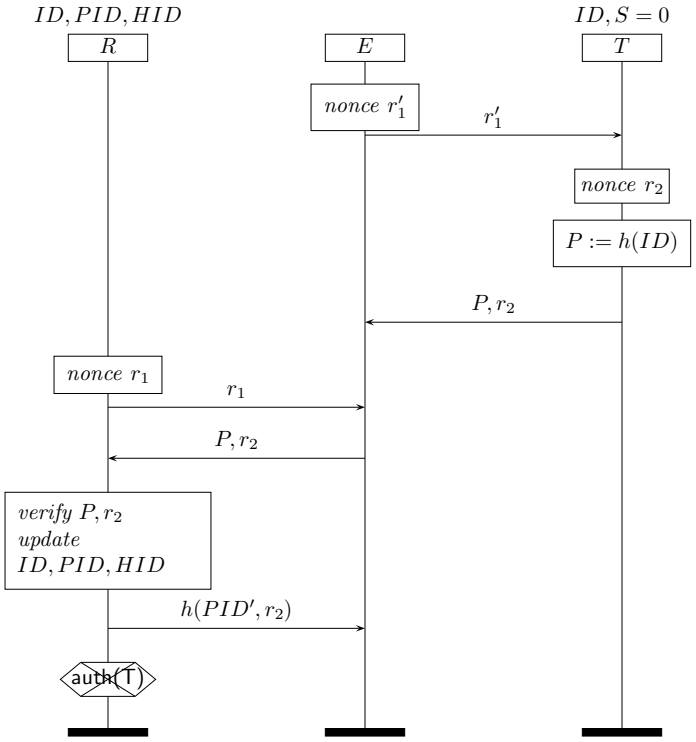

Figure 6: Attack on tag authentication

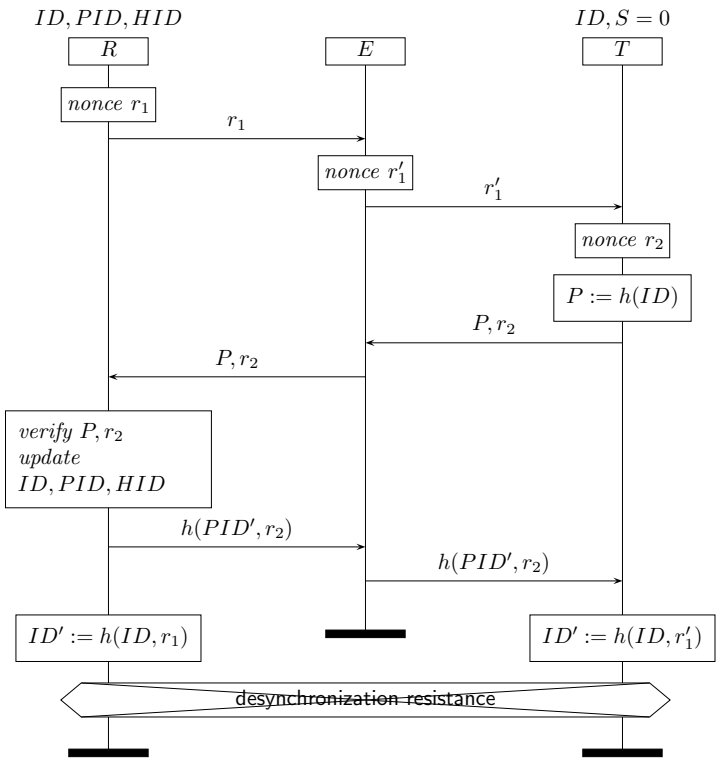

Figure 7: Attack on desynchronization resistance

4 [KCL07]

4.1 Description

The protocol is depicted in Figure 8.

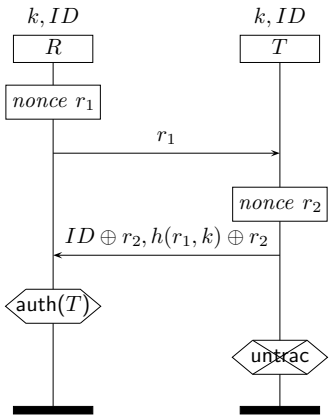

Figure 8: The KCL07 protocol

4.2 Claimed Attacks

4.2.1 Untraceability

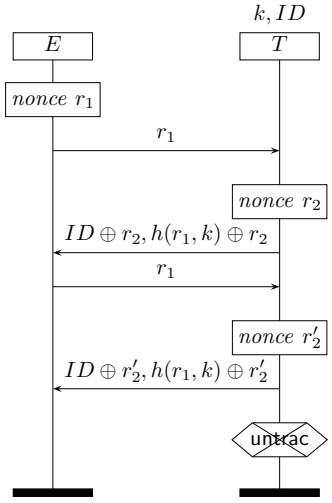

Figure 9: The attack on untraceability

To attack untraceability, the adversary challenges the tag twice with the

same nonce. He can then calculate the *xor* of the two parts $ID \oplus r_2$ and $h(r_1, k) \oplus r_2$ of the responses, the adversary then twice obtains $ID \oplus h(r_1, k)$, if and only if it was twice the same tag that he challenged. The attack is depicted in Figure 9.

5 [KCLL06]

5.1 Description

The protocol is depicted in Figure 10. In the original specification, the protocol control bits (PC) and a CRC are transmitted in the fourth message. These are irrelevant to any of the considered security properties and are therefore left out.

After being powered up, the tag generates a nonce r_1 and sends the xor of the nonce and its $PIN1$ to the reader. The reader acknowledges the message and generates a nonce r_2. Both the acknowledgement and the nonce are sent to the tag. The tag calculates the next message as is displayed in Figure 10. The reader applies a one-way function f to the xor of the two nonces and the $PIN2$. The result is xored with the PIN and sent to the tag.

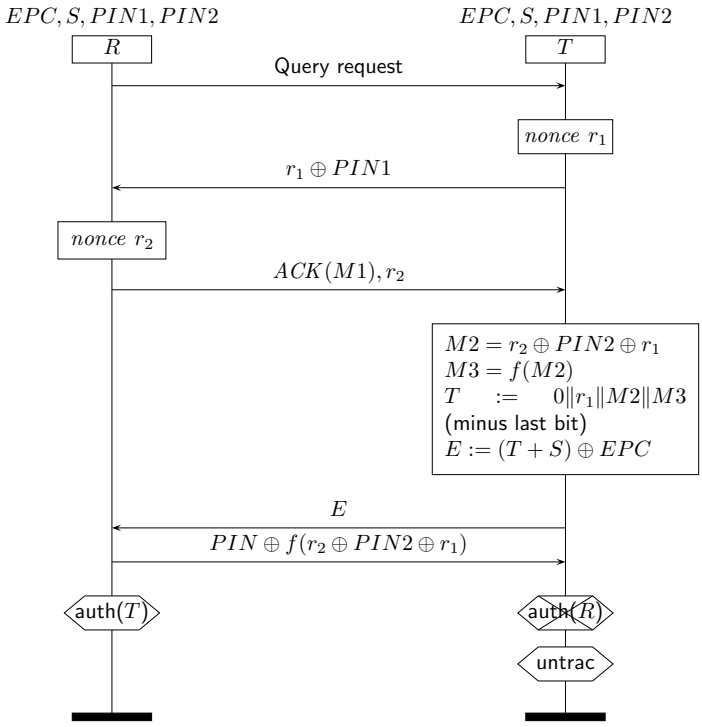

Figure 10: The protocol

5.2 Claimed Attacks

5.2.1 Reader authentication

The adversary can impersonate a legitimate reader by sending a nonce r_2' that allows him to replay a message he previously observed as a last message. In

order to be able to replay $PIN \oplus f(r_2 \oplus PIN2 \oplus r_1)$ in another session, the following condition must be satisfied: $r_1 \oplus r_2 = r'_1 \oplus r'_2$. This can be done by setting r'_2 to $r_1 \oplus r_2 \oplus r'_1$. The attack is depicted in Figure 11.

5.3 Related Protocols

We have found a similar attack on the protocols [CH07, LAK06, SM08].

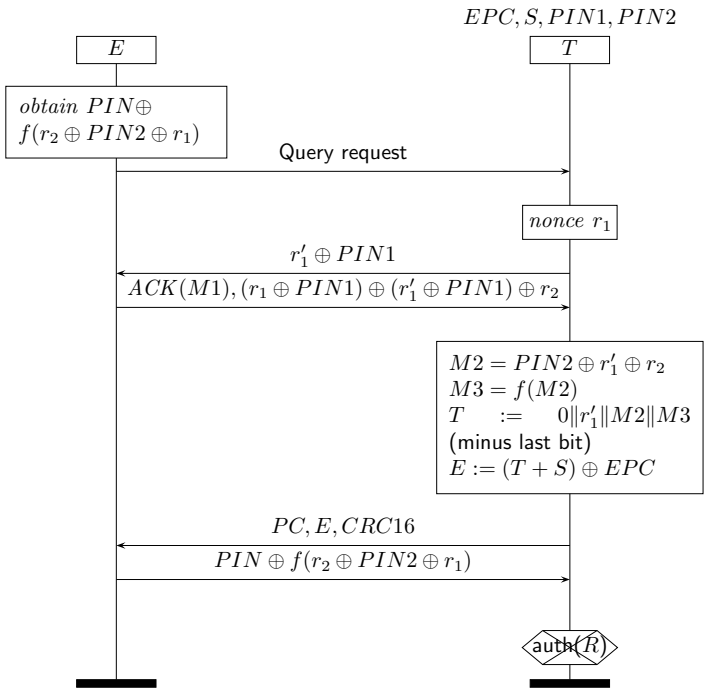

Figure 11: Attack on reader authentication

6 [KN05]

6.1 Description

In this protocol, the tag generates a random value r_0 from a small domain and a random value r_1 of length n. The tag sends the two hashes $h(ID, r_0)$, $h(r_1, k)$ and $ID \oplus r_1$ to the reader. Using $h(ID, r_0)$, the reader finds ID by trying out all combinations of values for ID stored in its database and of all possible values for r_0. This is possible for the reader because r_0 is chosen from a small domain and the number of IDs stored in its database is very small compared to the number of possible IDs. Using ID the reader retrieves k from its database, and using $ID \oplus r_1$ and ID, the reader finds r_1 and may then verify the correctness of the value of $h(r_1, k)$. The reader then generates a random value r_2 of length n and sends $ID \oplus r_2$ and $h(r_1, r_2)$ to the tag. The tag verifies these and sends $r_1 + r_2 \mod 2^n$ back to the reader. Both tag and reader update the ID by *xor*-ing it with $r_1 \oplus r_2$.

The protocol is depicted in Figure 12. Note that r_0 is chosen from a small domain, and can therefore be brute-forced from $h(ID, r_0)$ if ID is known.

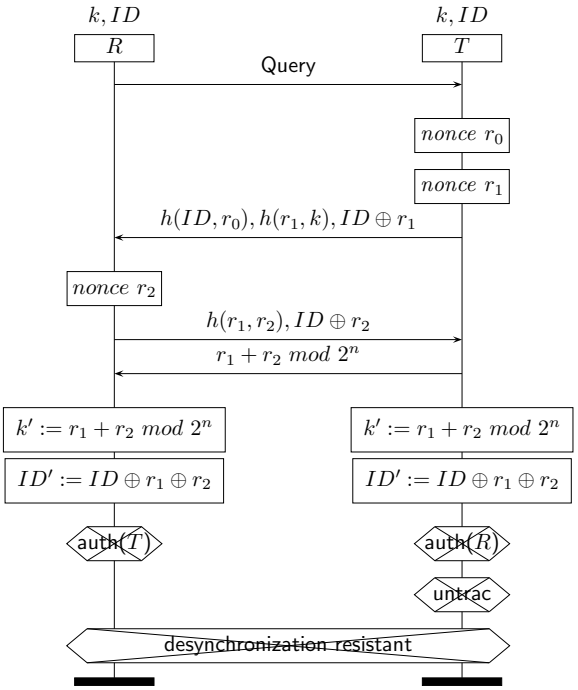

Figure 12: The protocol

6.2 Claimed Attacks

6.2.1 Tag authentication

An eavesdropping adversary is able to find bits of the ID by combining $h(ID, r_0)$, $ID \oplus r_1$, $ID \oplus r_2$, and $r_1 + r_2 \bmod 2^n$ observed in the last three messages of the protocol.

Since hash functions are assumed to be perfect, we consider the terms $ID \oplus r_1$, $ID \oplus r_2$, and $r_1 + r_2 \bmod 2^n$, setting up a system of equations involving the variables ID, r_1, r_2, and the values observed during runs of the protocol. A moment's thought shows that we may combine the first two equations to obtain $r_1 \oplus r_2$.

For convenience, we set $V = r_1 + r_2 \bmod 2^n$ and $W = r_1 \oplus r_2$. Let $V[i]$ be the i-th bit of V, and similarly for W, r_1, and r_2. Furthermore, let $V[1]$ be the least significant bit of V. By comparing addition modulo 2^n with *xor* it is easy to see that $V[i+1] \neq W[i+1]$ only if there is a carry bit in the computation of $V[i]$. If this is the case, then $r_1[i] \neq r_2[i] \Leftrightarrow W[i] = 1$ and $r_1[i] = r_2[i] = 1 \Leftrightarrow W[i] = 0$.

Since the latter case determines $r_1[i]$ and $r_2[i]$ uniquely, it follows that it can be used to find the i-th bit of ID. More bits from ID can be obtained by noticing that a carry bit in $V[i]$ followed by no carry bit in $V[i+1]$ implies $r_1[i+1] = r_2[i+1] = 0$.

Since r_1 and r_2 are chosen at random, on average, every communication session leaks roughly $\frac{n-1}{4}$ bits of ID. Revealing all bits of ID, once sufficiently many bits are known, can be achieved with a brute-force search over possible values for ID and r_0 and comparing their hash to $h(ID, r_0)$. Revealing all bits of ID is made a little more complicated by the fact that reader and tag update ID at the end of every protocol execution by setting it to $ID \oplus r_1 \oplus r_2$. The adversary may therefore need to keep track of two or three consecutive protocol executions between the tag and reader before performing the exhaustive search in order to completely reveal the tag's ID. Knowing the ID, the adversary can impersonate both tag and reader and furthermore trace the tag.

6.2.2 Reader authentication

Revealing the tag's ID as in Section 6.2.1 breaks reader authentication as well.

6.2.3 Untraceability

Revealing the tag's ID as in Section 6.2.1 breaks untraceability as well.

6.2.4 Desynchronization resistance

Revealing the tag's ID as in Section 6.2.1 breaks desynchronization resistance as well since the adversary can falsely authenticate to either the reader or the tag. The result is that reader and tag are desynchronized.

6.3 Related protocols

Many similar flaws have been documented in the literature. [CLL05] uses a counter in conjunction with *xor*. In [HMNB07b] the predictability of the counter and its interaction with *xor* are used to break the protocol. In [PLCETR06c, PLCETR06a, PLCETR06b] logical *and* and *or* operators are used in addition to *xor* and modular arithmetic leading to flaws described in [ALP07, LW07]. The cyclic redundancy check function is used with *xor* in [CC07] making the proposed protocol vulnerable to impersonation of tags and readers, and traceability of tags discovered in [PLHCETR07]. Finally, [DFJ07] breaks authentication in [VB03] where *xor* is used with bit-permutations.

7 [LAK06]

7.1 Description

The reader and tag share a secret k which is used for mutual authentication. To prevent desynchronization due to message loss, the old values of k is stored in k'. The reader initiates the protocol by challenging the tag with a nonce r_0. The tag generates a nonce r_1 and computes the response as in Figure 13. The reader uses the response to find the corresponding k in its database. The reader *xor*s the response with the reader nonce and the key and sends the cryptographic hash of the result to the tag.

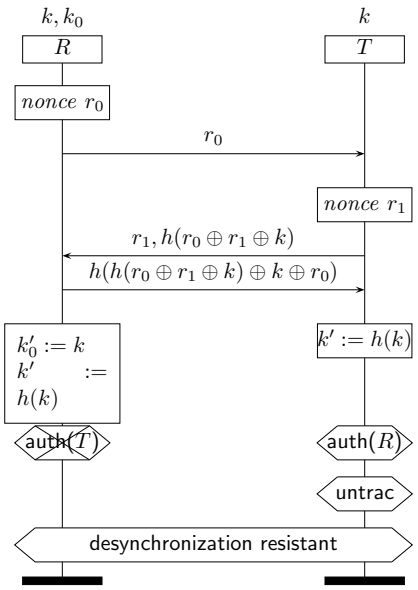

Figure 13: The protocol

7.2 Claimed Attacks

7.2.1 Tag Authentication

The adversary challenges the tag with a random value r_0, obtaining a response $r_1, h(r_0 \oplus r_1 \oplus k)$. When queried by a trusted reader with challenge r'_0, the adversary uses the response generated by the tag.

The attack is depicted in Figure 14. The adversary may replay $h(r_0 \oplus r_1 \oplus k)$ if he ensures that $r_0 \oplus r_1 = r'_0 \oplus r'_1$. To satisfy this condition the adversary sets r'_1 to $r_0 \oplus r_1 \oplus r'_1$.

7.3 Related Protocols

We have found a similar attack on the protocols [CH07, KCLL06, SM08].

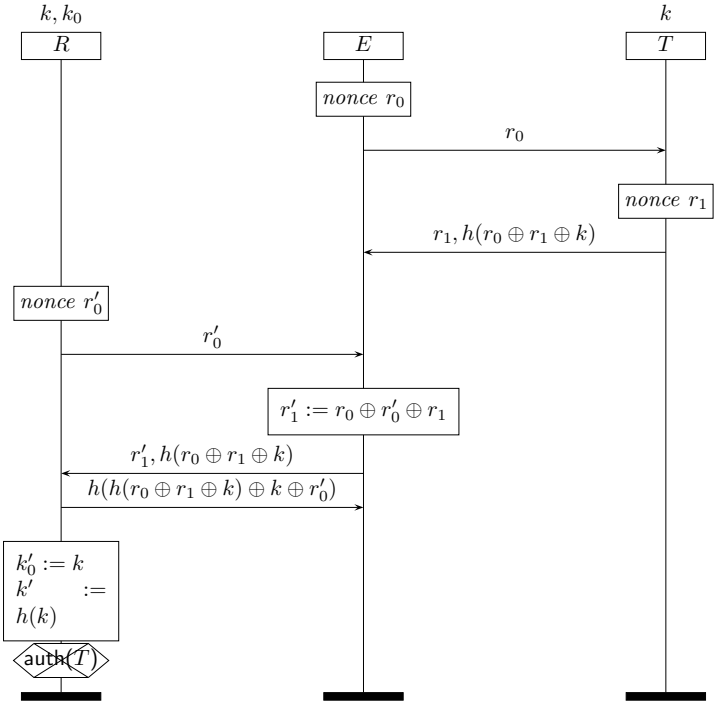

Figure 14: Attack on tag authentication

8 [LBV07]

8.1 Description

The protocol, shown in Figure 15 aims to efficiently authenticate a tag to a reader while keeping the tag untraceable. The protocol is based on a fixed, system-wide elliptic curve over a finite field. P, yP, x_1P, x_2P are publicly known points on the elliptic curve, the scalar y is only known to the reader, and the scalars x_1, x_2 are unique to each tag and only known to the tag. The elliptic curve is assumed to have been chosen such that it is difficult to compute, x_1, x_2, y from x_1P, x_2P, yP. The reader challenges the tag with a random number r_1, the tag responds with two points $T_1 = r_2P$, $T_2 = (r_2 + x_1)Y$ on the elliptic curve and a scalar $v = r_1(x_2 + r_2) + x_1$. The reader infers the tag's identity and authenticates it from the points and the scalar as follows. Since the reader knows y it can compute $y^{-1}T_2 - T_1 = x_1P$ to obtain the identity of the tag and then compute $(vP - x_1P)r_1^{-1} - T_1 = x_2P$ to authenticate the tag.

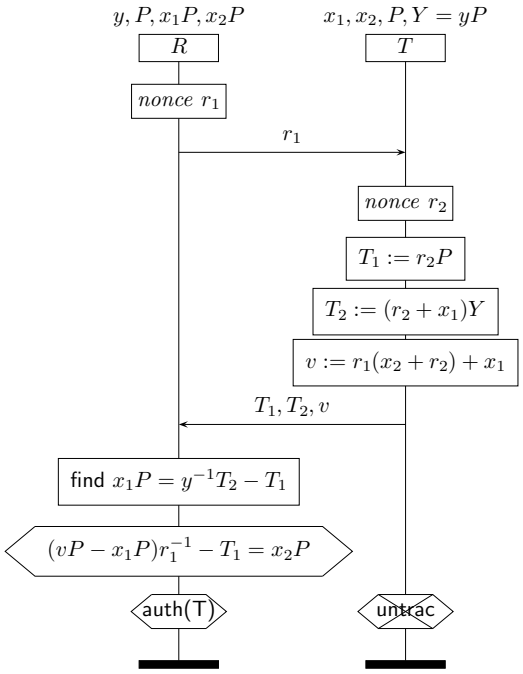

Figure 15: The protocol.

8.2 Claimed Attacks

8.2.1 Untraceability

- If the tag is challenged with $r_1 = 0$ the tag always responds with $v = x_1$.

- If the tag is challenged with $r_1 = 1$, the information obtained from the tag's response, $T_1 = r_2 P$, $T_2 = (x_1 + 1)yP$, $v = (x_2 + r_2) + x_1$, can be used to compute a constant, unique value for the tag $vP - T_1 = (x_1 + x_2)P$.
- If a tag is challenged twice, once with a random value r_1 and once with $r_1' = r_1 + 1$, then the information received from the tag in the two runs can be used to compute the constant term $-x_2 P$ as follows. Recall that primes indicate terms transmitted in the second run. Observe that

  $$v - v' = r_1(x_2 + r_2) - (r_1 + 1)(x_2 + r_2') = -x_2 - r_2' + r_1(r_2 - r_2'),$$

  thus we can compute

  $$-x_2 P = (v - v')P + T_1' - r_1(T_1 - T_1')$$

  since the terms on the right-hand side are known.

8.3 Related Protocols

[LBV08] is an improvement over [LBV07] but only addresses the first two flaws listed in section 8.2.1 but not the third one.

9 [LBV08]

9.1 Description

The protocol, shown in Figure 16 aims to efficiently authenticate a tag to a reader while keeping the tag untraceable. The protocol is based on a fixed, system-wide elliptic curve over a finite field. P, $Y = yP$, x_1P, x_2P are publicly known points on the elliptic curve, the scalar y is only known to the reader, the scalars x_1, x_2 are unique to each tag and only known to the tag. The elliptic curve is assumed to have been chosen such that it is difficult to compute, x_1, x_2, y from x_1P, x_2P, yP.

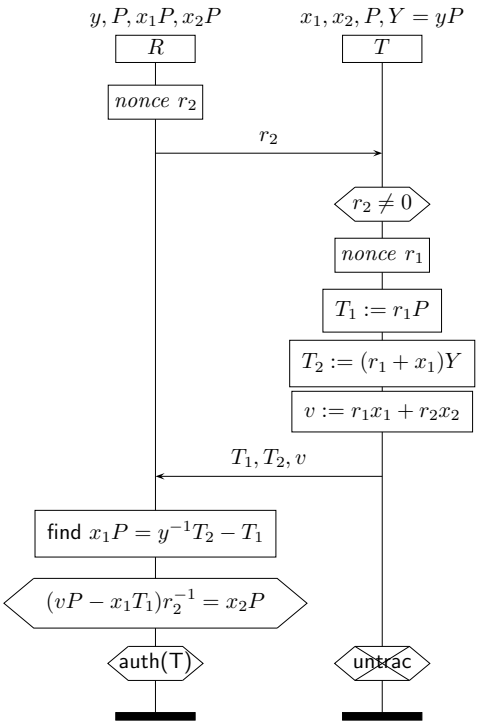

Figure 16: The protocol

9.2 Claimed Attacks

9.2.1 Untraceability

An attacker carries out two sessions with the tag sending the same nonce r_2 in both sessions. The attacker then computes (using primes for the second session) $v - v' = (r_1 - r_1')x_1$ and $T_1 - T_1' = (r_1 - r_1')P$. Thus computing the inverse of $v - v'$ modulo the order of the elliptic curve, the attacker obtains $x_1^{-1}P$ which identifies the tag uniquely.

9.3 Related Protocols

This is an improved version of [LBV07].

10 [LD07]

10.1 Description

The [LD07] protocol was designed for use in supply chains. Each supply chain consists of a chain of partners, each of which is represented by a reader. Reader R_i and tag T share a secret k_0. Additionally, reader R_i knows secrets k_i and k_{i+1}. At the end of a successful protocol execution, the tag updates the shared secret.

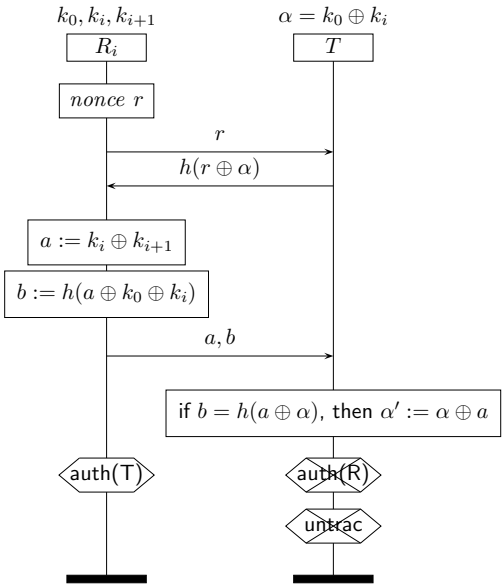

Figure 17: The protocol

10.2 Claimed Attacks

10.2.1 Untraceability

The protocol does not satisfy untraceability for the tag role, which is acknowledge by the protocol's authors and hence not claimed. This is because between any two updates of α, an adversary that twice sends the same challenge r to the same tag, will twice receive the same response. The authors do claim a weak form of untraceability, namely untraceability after updates. This claim is not satisfied either. The attack is shown in Figure 18 and runs as follows. By observing the authentication session the adversary learns $r, h(r \oplus \alpha), a$, and b. The adversary can now query the tag with $r' = r \oplus a$, to which the tag will respond with $h(r' \oplus \alpha')$. This response is equal to the previously observed one:

$$h(r' \oplus \alpha') = h(r \oplus a \oplus \alpha \oplus a) = h(r \oplus \alpha). \tag{2}$$

10.3 Reader Authentication

Reader authentication can be broken by setting $a = r$ and $b = h(r \oplus \alpha)$. The tag accepts a and b, because $b = h(a \oplus \alpha) = h(r \oplus \alpha)$. The attack is shown in Figure 19. This attack also results in desynchronization of the database and the tag.

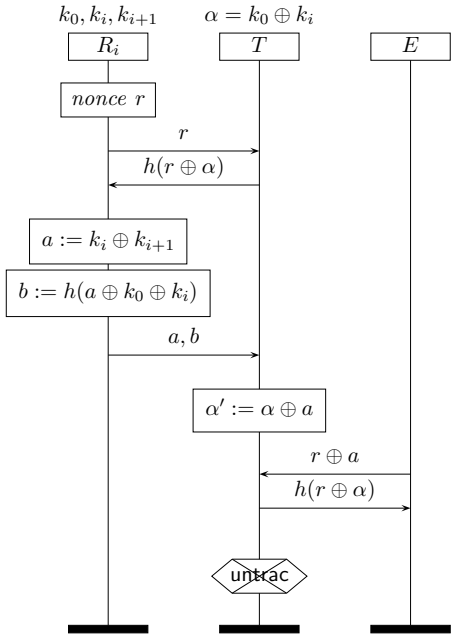

Figure 18: Attack on untraceability

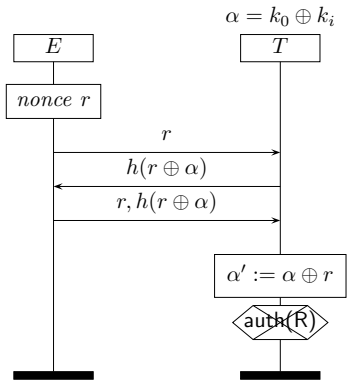

Figure 19: Attack on reader authentication

10.4 Related Protocols

We have found similar attacks on untraceability in [YPL+05, OTYT06, KCL07]. The protocol [LCUL06] is vulnerable to a simpler form of this attack which has been shown in [CH07].

11 [OTYT06]

11.1 Description

The protocol is depicted in Figure 20.

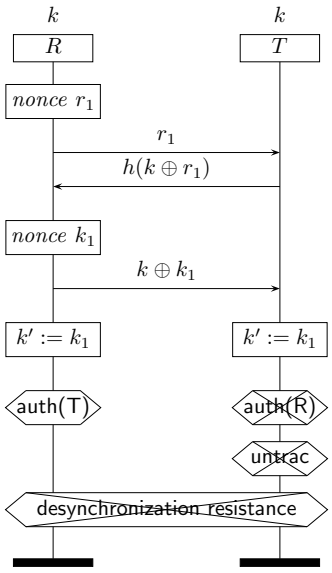

Figure 20: The protocol

11.2 Claimed Attacks

11.2.1 Reader authentication

Since the tag does not know the new key k_1, the tag is not able to verify whether the third message is indeed $k \oplus k_1$. Since no check can be performed by the tag, the adversary may send a random message r to the tag which will cause the tag to replace k by $k \oplus r$.

11.2.2 Desynchronization resistance

- The attack on reader authentication desynchronizes the secret key k, shared between the tag and the reader, rendering future authentication impossible. Note that the attacker is the only one who can re-synchronize the secret information between reader and tag since he is the only one who knows $k \oplus r$.
- Modifying the third message leads tag and reader to carry out different key updates, leaving them in a desynchronized state.

- Blocking the last message from reader to tag leads the reader to update k while the tag does not carry out the update, leaving tag and reader in a desynchronized state.

11.2.3 Untraceability

An attacker observing a protocol run obtains a triple $(r, h(k \oplus r), k \oplus k_1)$. He may now challenge a tag with $r \oplus k \oplus k_1$ giving him the same response he already observed, provided that the tag is the same as the one which was eavesdropped on before. The attack is depicted in Figure 21.

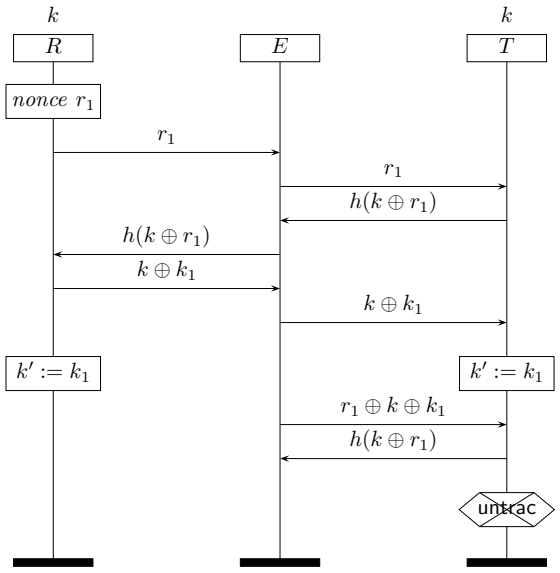

Figure 21: Attack on untraceability

11.3 Related Protocols

We have found similar flaws in the protocols [YPL+05, KCL07].

12 [LY07a, LY07c, LY07b, HM04]

12.1 Description

The protocols have a challenge-response structure as depicted in Figure 22. The reader challenges the tag, the tag computes a function over one or more terms in its knowledge and sends the result to the reader. However, the challenge is not used by the tag as an input to the function.

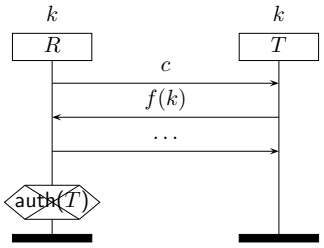

Figure 22: General protocol structure.

12.2 Claimed Attacks

12.2.1 Tag authentication

Because the tag's response does not depend on the reader's challenge, an adversary may query a tag and later replay the response to a reader when challenged. Therefore, none of these protocols satisfy the recent aliveness claim with respect to the tag role. The general structure of the attack is depicted in Figure 23.

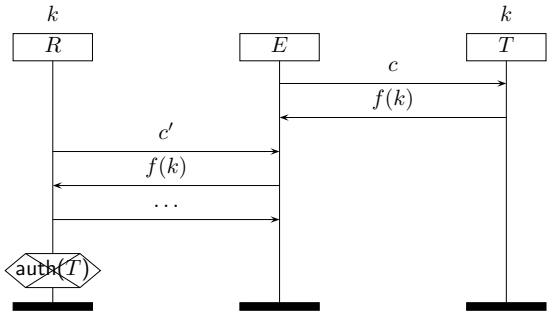

Figure 23: Attack on tag authentication

12.3 Related Protocols

The protocols [SLK06, HMNB07a] suffer from the same problem.

13 [SLK06]

13.1 Description

The protocol assumes that the reader and tag share the secrets k, ID, and PIN. While ID and PIN are unique to each tag, k is equal for all tags the reader is allowed to authenticate. The tag further stores the timestamp TS_{last} of the last successful mutual authentication initialized to 0 at the factory. The protocol is depicted in Figure 24.

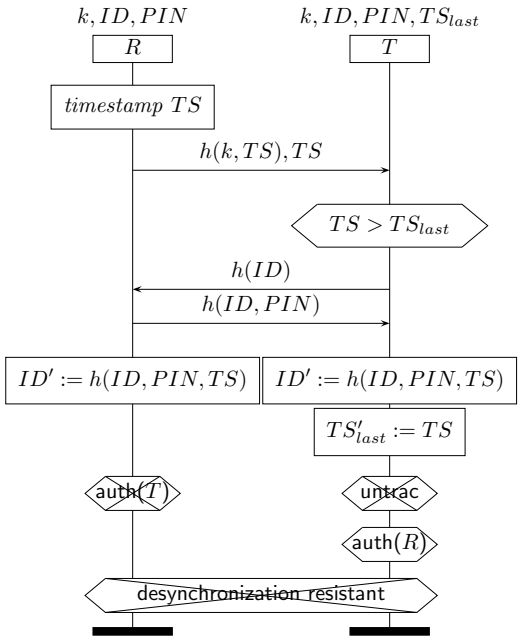

Figure 24: The protocol

13.2 Claimed Attacks

13.2.1 Tag authentication

To attack the protocol, it suffices to note that the challenge of the reader and the response of the tag are not related. See Section 12 for the attack.

13.2.2 Desynchronization resistance

The attack described in section 13.2.1 leads to a situation in which the reader updates ID, but the tag does not. The same result can be achieved by blocking the last message from a reader to a tag. This essentially kills the tag since the reader will not accept the tag's $h(ID)$ message in a future protocol run.

13.2.3 Untraceability

The fact that a reader and tag do not agree on the value ID, i.e. are desynchronized, is observable, since in such a case the reader terminates the protocol early. Thus the adversary can trace such tags. Furthermore, when a tag becomes desynchronized, it will not be able to update ID and TS_{last} anymore, thus its response to any valid challenge $h(k, TS), TS$ with $TS > TS_{last}$ will remain constant allowing an adversary to distinguish between recently desynchronized tags and earlier desynchronized tags.

13.3 Related Protocols

The same authentication problem exists in the protocols [LY07c, LY07a, LY07b, HMNB07a].

In [Avo05] a quality-time attack on the untraceability claim of the stateful protocol [HM04] is presented. The attack involves increasing a tag's internal counter to an abnormal level in order to recognize the tag later.

14 [SM08]

14.1 Description

The protocol is depicted in Figure 25. Bit rotations are denoted by $\gg$ and $\ll$ where $a \gg b$ means a shifted cyclically to the right by b bits. The function f_t that is used to compute M_2 is a keyed hash function, where t is the key.

The reader and tag share a secret t which is used for mutual authentication. The reader also stores the hash of t in u. To prevent desynchronization due to message loss, the old values of t and u are stored in t' and u'. The reader initiates the protocol by generating a random value r_1 of length ℓ and challenging the tag with r_1. The tag generates a nonce r_2 and computes the response M_1, M_2 as in Figure 25. The reader uses M_1 and M_2 to find the tag in its database. The reader computes M_3 and sends it to the tag, after which both reader and tag update their secrets.

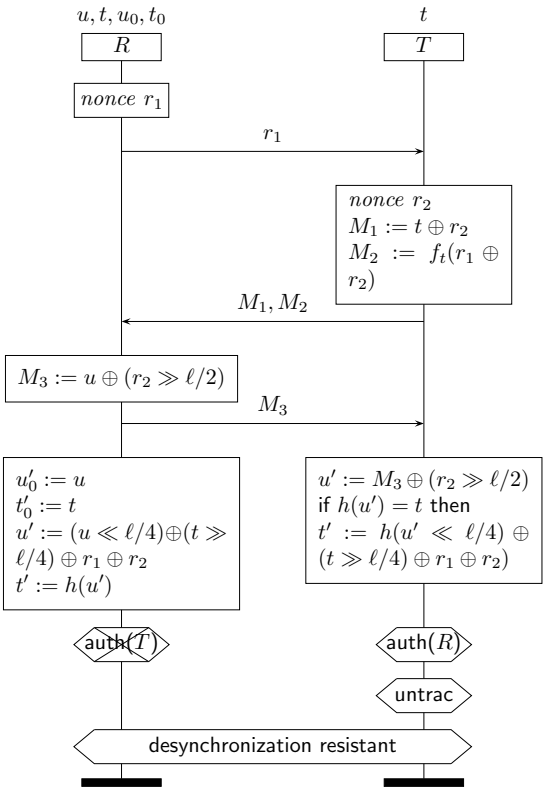

Figure 25: The protocol

14.2 Claimed Attacks

14.2.1 Tag authentication

The attack on tag authentication is depicted in Figure 26. The attacker uses the fact that he may replay M_2 for M_2' if he ensures that $r_1 \oplus r_2 = r_1' \oplus r_2'$. To satisfy this condition he sets M_1' to $M_1 \oplus r_1 \oplus r_1'$.

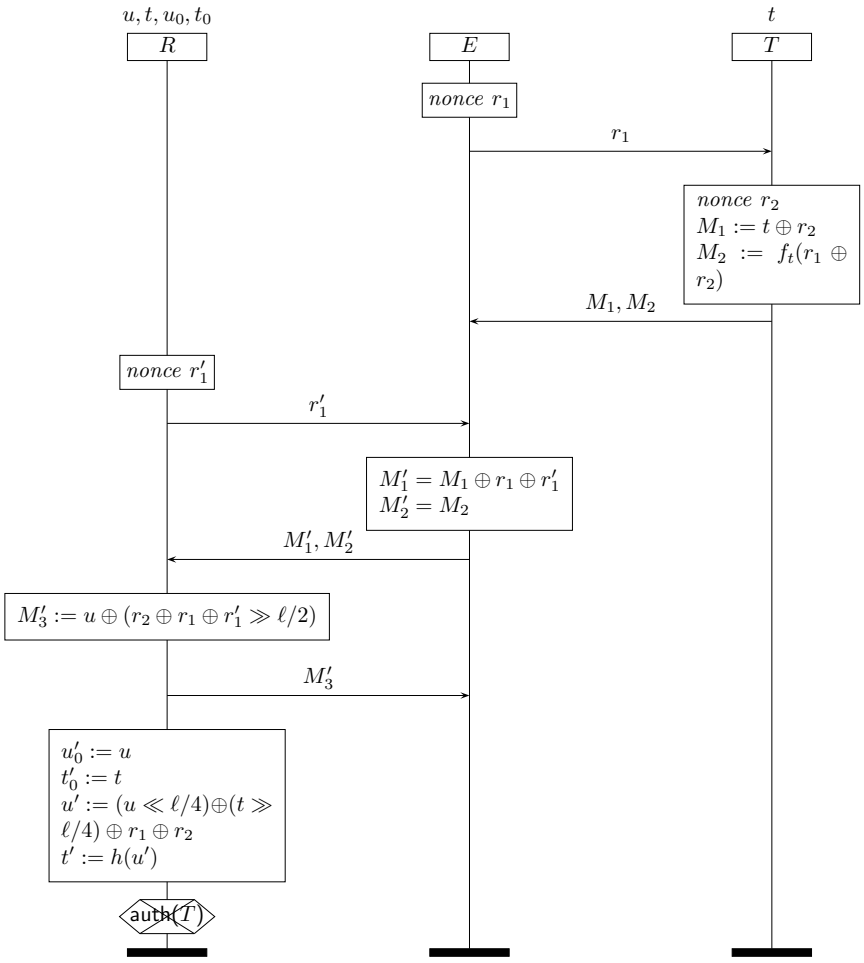

Figure 26: Attack on tag authentication

14.2.2 Reader authentication

The attack on reader authentication is depicted in Figure 27.

14.2.3 Desynchronization resistance

The attack depicted in Figure 27 desynchronizes the reader from the tag.

14.2.4 Untraceability

After the reader and tag are desynchronized using the attack depicted in Figure 27, the adversary can recognize it since it is no longer accepted by a valid reader.

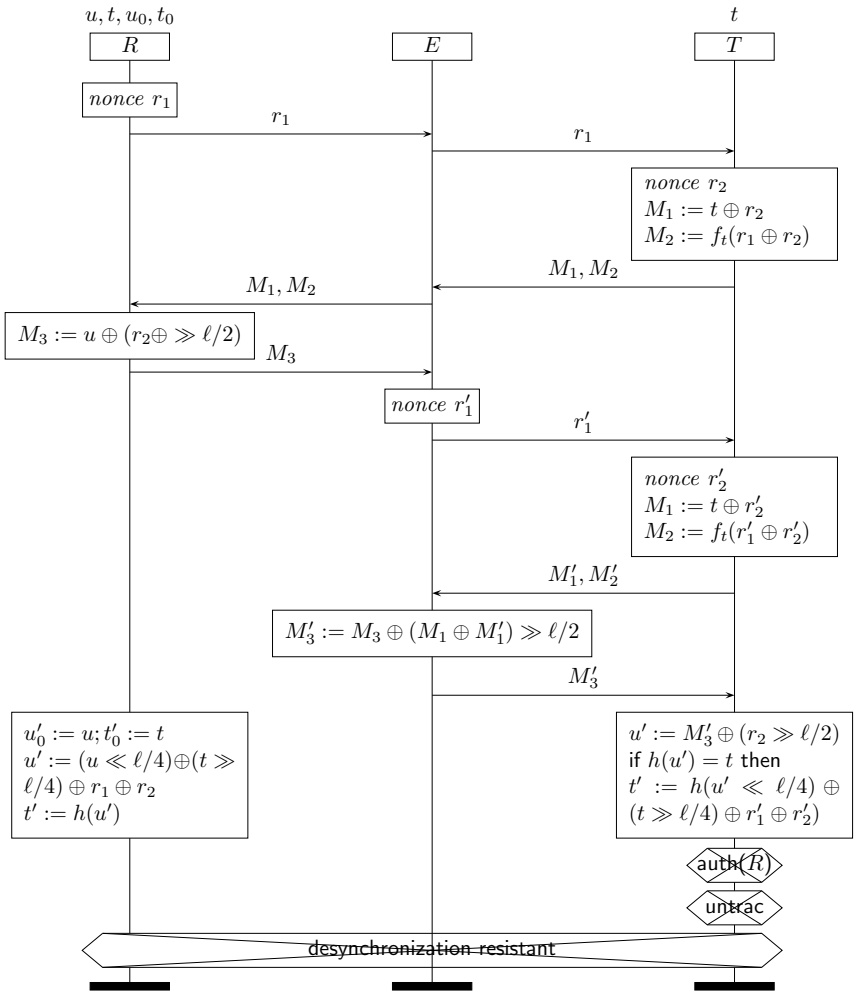

Figure 27: Attack on reader authentication

14.3 Related Protocols

We have found a similar attack on the protocols [CH07, LAK06, KCLL06].

15 [YPL+05]

15.1 Description

The reader and tag share secrets k, k_1, k_2. The reader initiates the protocol by challenging the tag with a nonce r_1. The tag responds with $h(k_1 \oplus r_1 \oplus k)$. The reader then replies with $h(k_2)$ and both tag and reader update secrets k_1 and k_2. Figure 28 depicts the protocol.

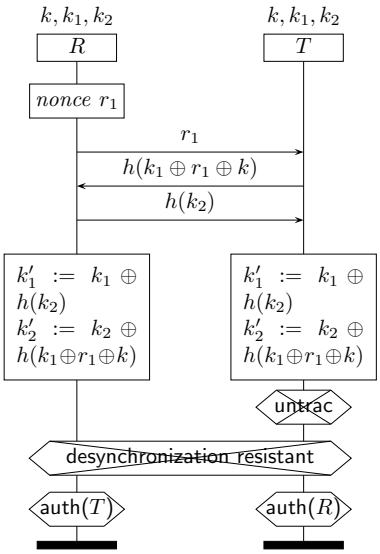

Figure 28: The protocol

15.2 Claimed Attacks

15.2.1 Untraceability

An attacker observing a communication session of the protocol obtains the messages $r_1, h(k_1 \oplus r_1 \oplus k), h(k_2)$. Reader and tag then update their secrets. The attacker can recognize the tag by challenging it with $r_1 \oplus h(k_2)$ to which the previously observed tag will respond with $h(k_1 \oplus r_1 \oplus k)$. Figure 29 depicts the attack.

15.2.2 Desynchronization resistance

Blocking the third message in the protocol from the reader to the tag, leads to the reader updating its secrets while the tag does not update them. Therefore, the secret information between the reader and tag will be desynchronized, rendering future authentication impossible.

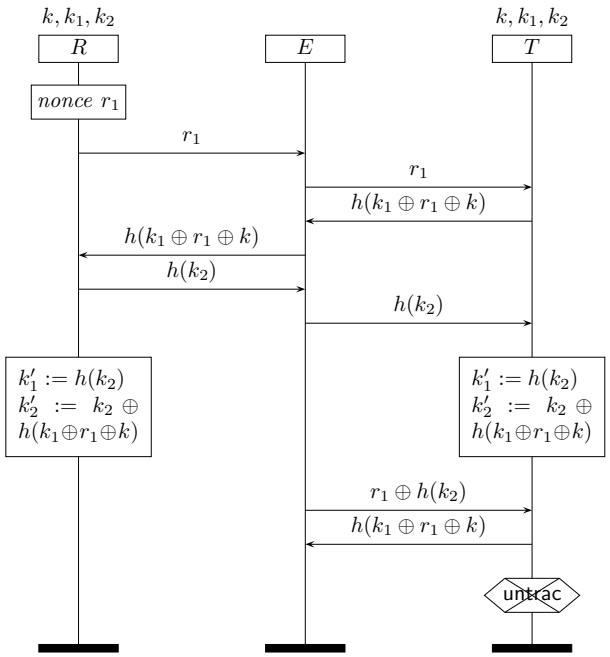

Figure 29: Attack on the untraceability

15.3 Related Protocols

We found similar flaws in the protocols in [OTYT06, KCL07].

References

[ALP07] Basel Alomair, Loukas Lazos, and Radha Poovendran. Passive attacks on a class of authentication protocols for RFID. In *ICISC*, pages 102–115, 2007. 6.3

[Avo05] Gildas Avoine. Adversary model for radio frequency identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, September 2005. 3.3, 13.3

[BCI08] Julien Bringer, Hervé Chabanne, and Thomas Icart. Cryptanalysis of EC-RAC, a RFID identification protocol. In *CANS*, pages 149–161, 2008.

[CC07] Hung-Yu Chien and Che-Hao Chen. Mutual authentication protocol for RFID conforming to EPC class 1 generation 2 standards. *Computer Standards & Interfaces, Elsevier Science Publishers*, 29(2):254–259, February 2007. 6.3

[CH07] Hung-Yu Chien and Chen-Wei Huang. A lightweight RFID protocol using substring. In *Embedded and Ubiquitous Computing (EUC)*, pages 422–431, 2007. 1, 5.3, 7.3, 10.4, 14.3

[CLL05] Eun Young Choi, Su Mi Lee, and Dong Hoon Lee. Efficient RFID authentication protocol for ubiquitous computing environment. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuanshun Dai, and Laurence Yang, editors, *International Workshop on Security in Ubiquitous Computing Systems – secubiq 2005*, volume 3823 of *Lecture Notes in Computer Science*, pages 945–954, Nagasaki, Japan, December 2005. Springer-Verlag. 6.3

[CM05] C.J.F. Cremers and S. Mauw. Operational semantics of security protocols. In S. Leue and T.J. Systä, editors, *Scenarios: Models, Algorithms and Tools (Dagstuhl 03371 post-seminar proceedings, September 7–12, 2003)*, volume 3466 of *LNCS*, pages 66–89, 2005.

[DFJ07] Benessa Defend, Kevin Fu, and Ari Juels. Cryptanalysis of two lightweight RFID authentication schemes. In *PerCom Workshops*, pages 211–216, 2007. 6.3

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DM07] Roberto Di Pietro and Refik Molva. Information confinement, privacy, and security in RFID systems. In *ESORICS*, pages 187–202, 2007. 2

[DMR08] Ton van Deursen, Sjouke Mauw, and Saša Radomirović. Untraceability of RFID protocols. In *Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, volume 5019 of *Lecture Notes in Computer Science*, pages 1–15, Seville, Spain, 2008. Springer.

[DMRV09] Ton van Deursen, Sjouke Mauw, Saša Radomirović, and Pim Vullers. Secure ownership and ownership transfer in RFID systems. In *Proc. 14th European Symposium On Research In Computer Security (ESORICS'09)*, Lecture Notes in Computer Science. Springer, 2009. To appear.

[DP08] Ivan Damgård and Michael Østergaard Pedersen. RFID security: Tradeoffs between security and efficiency. In *CT-RSA*, pages 318–332, 2008.

[DR08a] Ton van Deursen and Saša Radomirović. Security of an RFID protocol for supply chains. In *Proceedings of the 1st Workshop on Advances in RFID, AIR'08 (to appear)*. IEEE Computer Society, October 2008.

[DR08b] Ton van Deursen and Saša Radomirović. Security of RFID protocols – A case study. In *Proceedings of the 4th International Workshop on Security and Trust Management, STM 2008 (to appear)*, ENTCS. Elsevier, June 2008.

[DR09] Ton van Deursen and Saša Radomirović. Algebraic attacks on RFID protocols. In *Information Security Theory and Practices. Smart Devices, Pervasive Systems, and Ubiquitous Networks (to appear)*, Lecture Notes in Computer Science, Brussels, Belgium, 2009. Springer.

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

[GRS05] Henri Gilbert, Matthew Robshaw, and Hervé Sibert. An active attack against HB^+ – a provably secure lightweight authentication protocol. Manuscript, July 2005. 2.3

[HM04] Dirk Henrici and Paul Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *PerCom Workshops*, pages 149–153, 2004. 3.3, 12, 13.3

[HMNB07a] JaeCheol Ha, Sang-Jae Moon, Juan Manuel González Nieto, and Colin Boyd. Low-cost and strong-security RFID authentication protocol. In *Embedded and Ubiquitous Computing (EUC) Workshops*, pages 795–807, 2007. 3, 12.3, 13.3

[HMNB07b] JaeCheol Ha, Sang-Jae Moon, Juan Manuel González Nieto, and Colin Boyd. Security analysis and enhancement of one-way hash based low-cost authentication protocol (OHLCAP). In *PAKDD Workshops*, pages 574–583, 2007. 6.3

[JW05] Ari Juels and Stephen Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO'05*, volume 3126 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. IACR, Springer-Verlag. 2.3

[JW07] Ari Juels and Stephen Weis. Defining strong privacy for RFID. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 342–347, New York, USA, March 2007. IEEE, IEEE Computer Society Press.

[KCL07] Il Jung Kim, Eun Young Choi, and Dong Hoon Lee. Secure mobile RFID system against privacy and security problems. In *SecPerU 2007*, 2007. 4, 10.4, 11.3, 15.3

[KCLL06] Kyoung Hyun Kim, Eun Young Choi, Su-Mi Lee, and Dong Hoon Lee. Secure EPCglobal class-1 gen-2 RFID system against security and privacy problems. In *On The Move (OTM) Workshops (1)*, pages 362–371, 2006. 1.3, 5, 7.3, 14.3, 15.3

[KN05] Jeonil Kang and Daehun Nyang. RFID authentication protocol with strong resistance against traceability and denial of service attacks. In Refik Molva, Gene Tsudik, and Dirk Westhoff, editors, *European Workshop on Security and Privacy in Ad hoc and Sensor Networks – ESAS'05*, volume 3813 of *Lecture Notes in Computer Science*, pages 164–175, Visegrad, Hungary, July 2005. Springer-Verlag. 6, 15.3

[LAK06] RFID mutual authentication scheme based on synchronized secret information. In *Symposium on Cryptography and Information Security*, Hiroshima, Japan, January 2006. 1.3, 5.3, 7, 14.3, 15.3

[LBV07] Yong Ki Lee, Lejla Batina, and Ingrid Verbauwhede. Provably secure RFID authentication protocol EC-RAC (ECDLP based randomized access control). 2007. 8, 8.3, 9.3

[LBV08] Yong Ki Lee, Lejla Batina, and Ingrid Verbauwhede. EC-RAC (ECDLP based randomized access control): Provably secure RFID authentication protocol. In *Proceedings of the 2008 IEEE International Conference on RFID*, pages 97–104, 2008. 8.3, 9

[LCUL06] Yong-Zhen Li, Young-Bok Cho, Nam-Kyoung Um, and Sang Ho Lee. Security and privacy on authentication protocol for low-cost RFID. In *Computational Intellegence and Security (CIS)*, pages 788–794, 2006. 10.4

[LD07] Yingjiu Li and Xuhua Ding. Protecting RFID communications in supply chains. In *ASIACCS*, pages 234–241, 2007. 10, 10.1

[Low97] Gavin Lowe. A hierarchy of authentication specifications. In *CSFW*, pages 31–44, 1997.

[LW07] Tieyan Li and Guilin Wang. Security analysis of two ultra-lightweight RFID authentication protocols. In *IFIP SEC 2007*, Sandton, Gauteng, South Africa, May 2007. IFIP. 6.3

[LY07a] N. W. Lo and Kuo-Hui Yeh. An efficient mutual authentication scheme for EPCglobal class-1 generation-2 RFID system. In *Embedded and Ubiquitous Computing (EUC) Workshops*, pages 43–56, 2007. 3.3, 12, 13.3

[LY07b] N. W. Lo and Kuo-Hui Yeh. Hash-based mutual authentication protocol for mobile RFID systems with robust reader-side privacy protection, to appear. 2007. 3.3, 12, 13.3

[LY07c] N. W. Lo and Kuo-Hui Yeh. Novel RFID authentication schemes for security enhancement and system efficiency. In *Secure Data Management*, pages 203–212, 2007. 3.3, 12, 13.3

[OTYT06] Kyosuke Osaka, Tsuyoshi Takagi, Kenichi Yamazaki, and Osamu Takahashi. An efficient and secure RFID security method with ownership transfer. In *Computational Intellegence and Security (CIS)*, pages 778–787, 2006. 10.4, 11, 15.3

[PLCETR06a] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, and Arturo Ribagorda. EMAP: An efficient mutual-authentication protocol for low-cost RFID tags. In *On The Move (OTM) Workshops (1)*, pages 352–361, 2006. 6.3

[PLCETR06b] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, and Arturo Ribagorda. LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags. Printed handout of Workshop on RFID Security – RFIDSec 06, July 2006. 6.3

[PLCETR06c] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, and Arturo Ribagorda. M^{2}AP: A minimalist mutual-authentication protocol for low-cost RFID tags. In *Ubiquitous Intellegence and Computing (UIC)*, pages 912–923, 2006. 6.3

[PLHCETR07] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan Estevez-Tapiador, and Arturo Ribagorda. Cryptanalysis of a novel authentication protocol conforming to EPC-C1G2 standard., 2007. 6.3

[PV08] Radu-Ioan Paise and Serge Vaudenay. Mutual authentication in RFID: Security and privacy. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pages 292–299. ACM Press, 2008.

[SLK06] Youngjoon Seo, Hyunrok Lee, and Kwangjo Kim. A scalable and untraceable authentication protocol for RFID. In *Embedded and Ubiquitous Computing (EUC) Workshops*, pages 252–261, 2006. 12.3, 13

[SM08] Boyeon Song and Chris J. Mitchell. RFID authentication protocol for low-cost tags. In *Wireless Network Security (WISEC)*, pages 140–147, 2008. 1.3, 5.3, 7.3, 14, 15.3

[THG98] F.J. Thayer Fàbrega, J.C. Herzog, and J.D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 1998 IEEE Symposium on Security and Privacy*, pages 66–77, Oakland, California, 1998.

[Vau07] Serge Vaudenay. On privacy models for RFID. In *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, December 2007. Springer-Verlag.

[VB03] István Vajda and Levente Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, Seattle, WA, USA, October 2003. 6.3

[YPL+05] Jeongkyu Yang, Jaemin Park, Hyunrok Lee, Kui Ren, and Kwangjo Kim. Mutual authentication protocol for low-cost RFID. Handout of the Ecrypt Workshop on RFID and Lightweight Crypto, July 2005. 10.4, 11.3, 15, 15.3

Change Log

V1.0 to V1.1

- Added Change Log.
- Added attacks to [SM08].
- Added sections describing algebraic replay attacks, attribute acquisition attacks, and cryptanalytic attacks.
- Expanded descriptions of [KCLL06], [KN05], [LAK06], and [YPL+05].
- Edited section on preliminaries, added reference for desynchronization resistance.