The Swiss-Knife RFID Distance Bounding Protocol

Chong Hee Kim*, Gildas Avoine, François Koeune*,
François-Xavier Standaert**, and Olivier Pereira**

Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Abstract. Relay attacks are one of the most challenging threats RFID will have to face in the close future. They consist in making the verifier believe that the prover is in its close vicinity by surreptitiously forwarding the signal between the verifier and an out-of-field prover. Distance bounding protocols represent a promising way to thwart relay attacks, by measuring the round trip time of short authenticated messages. Several such protocols have been designed during the last years but none of them combine all the features one may expect in a RFID system.

We introduce in this paper the first solution that compounds in a single protocol all these desirable features. We prove, with respect to the previous protocols, that our proposal is the best one in terms of security, privacy, tag computational overhead, and fault tolerance. We also point out a weakness in Tu and Piramuthu's protocol, which was considered up to now as one of the most efficient distance bounding protocol.

1 Introduction

Radio Frequency Identification (RFID) is a ubiquitous technology that enables identification of non-line-of-sight objects or subjects. Based on cheap RF - microcircuits – called tags – apposed on or incorporated into the items to identify, the RFID technology is widely deployed in our everyday lives. Several billion RFID tags are spread every year, in applications as diverse as pet identification, supply chain managment, Alzheimer's patient tracking, cattle counting, etc. RFID tags suited to such applications do not cost more than 0.20 USD.

The impressive potential of the RFID is not only exploited in identification solutions, but also in more evolved applications like access control, public transportation, payment, ePassport, etc. that require the tag to be cryptographically authenticated by the reader. To do so, a cipher and a pseudo-random number generator can be implemented on the tag while keeping its cost low – e.g. no more than 1 USD for a public transportation pass – but the number of calls to these cryptographic functions must be small enough to keep the authentication delay reasonable. Preserving privacy is also an expected feature of these protocols.

In practice, sensitive applications like those mentioned above rely on 2-pass or 3-pass challenge-response authentication protocols based on symmetric-key

* Research supported by the Walloon Region project E-USER (WIST program).
** Research Associates of the Fonds de la Recherche Scientifique - FNRS.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 98–115, 2009.

building blocks, typically block ciphers, although solutions based on asymmetric primitives have also been proposed. Such a design is secure in theory, but the real life is a bit different when dealing with RFID. Indeed, a tag is quite a simple device that automatically answers to any authentication query from a reader without alerting its holder. Hence the reader has no means to decide whether the tag's holder agreed to authenticate. Because the maximum reader-tag communication distance can not exceed a few decimeters with cryptography-compliant tags, the presence of the tag in the close environment of the reader is considered as an implicit authentication agreement from its holder. Providing the reader with a means to decide whether the distance to the tag is less than a given threshold is thus of the utmost importance to achieve practical security in RFID systems.

We introduce in this paper an RFID authentication protocol that allows such a verification. It is the first protocol that combines all the expected properties at the same time: it resists against both mafia fraud and terrorist attacks, reaches the best known false acceptance rate, preserves privacy, resists to channel errors, uses symmetric-key cryptography only, requires no more than 2 cryptographic operations to be performed by the tag, can take advantage of precomputation on the tag, and offers an optional mutual authentication. As an additional result, we also point out a weakness in the recent Tu and Piramuthu distance bounding protocol.

In Section 2, we introduce the relay attacks and the existing distance bounding protocols. We show that they all offer interesting features, but no one was yet able to combine all these features. We show in Section 3 a new attack against one of these protocols. We then describe our proposal in Section 4 and analyze it in Section 5. Finally, we provide a security and efficiency analysis.

2 Relay Attacks and Distance Bounding Protocols

2.1 Relay Attacks

There are two types of relay attacks: mafia fraud attack and terrorist fraud attack. **Mafia fraud attack** was first described by Desmedt [5]. In this attack scenario, both the reader R and the tag T are honest, but a malicious adversary is performing man-in-the-middle attack between the reader and the tag by putting fraudulent tag $\overline{T}$ and receiver $\overline{R}$. The fraudulent tag $\overline{T}$ interacts with the honest reader R and the fraudulent reader $\overline{R}$ interacts with the honest tag T. $\overline{T}$ and $\overline{R}$ cooperate together. It enables $\overline{T}$ to convince R as if R communicates with T, without actually needing to know anything about the secret information. **Terrorist fraud attack** is an extension of the mafia fraud attack. The tag T is not honest and collaborates with fraudulent tag $\overline{T}$. The dishonest tag T uses $\overline{T}$ to convince the reader that he is close, while in fact he is not. $\overline{T}$ does not know the long-term private or secret key of T. The problem with Mafia fraud attack is that this attack can be mounted without the notice of both the reader and the tag. It is difficult to prevent since the adversary does not change any data between the reader and the tag. Therefore mafia fraud attack cannot be

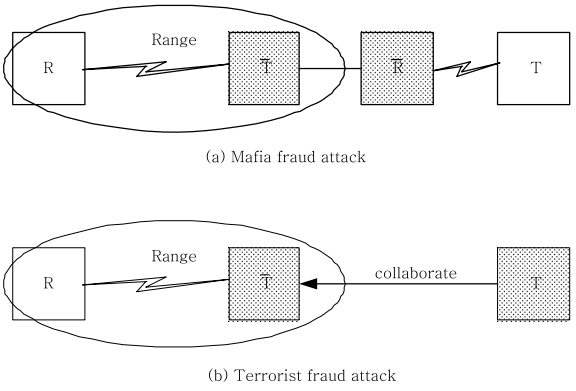


Fig. 1. Mafia and terrorist fraud attacks

prevented by cryptographic protocols that operate at the application layer. Although one could verify location through use of GPS coordinates, small resource limited devices such as RFID tags do not lend themselves to such applications. *Distance bounding protocols* are good solutions to prevent such relay attacks. These protocols measure the signal strength or the round-trip time between the reader and the tag. However the proof based on measuring signal strength is not secure as an adversary can easily amplify signal strength as desired or use stronger signals to read from afar. Therefore many works are devoted to devise efficient distance bounding protocols by measuring round-trip time [2,3,7,10,11,4,14,15,16].

2.2 Distance Bounding Protocols

In 1993, Brands and Chaum presented their distance bounding protocol [2]. It consists of a *fast bit exchange* phase where the reader sends out one bit and starts a timer. Then the tag responds to the reader with one bit that stops the timer. The reader uses the round trip time to extract the propagation time. After series of n rounds (n is a security parameter), the reader decides whether the tag is within the limitation of the distance. In order to extract the propagation time, the processing time of the tag must be as short and invariant as possible. The communication method used for these exchanges is different from the one used for the ordinary communication. It does not contain any error detection or correction mechanism in order to avoid the introduction of variable processing cycles.

Although the idea has been introduced fifteen years ago, it is only quite recently that distance-bounding protocols attracted the attention of the research community. In 2005, Hancke and Kuhn proposed a distance bounding protocol (HKP) [7] that has been chosen as a reference-point because it is the most popular distance bounding protocol in the RFID framework. As depicted in Fig. 2, the protocol is carried out as follows. After exchanges of random nonces (N_a and N_b), the reader and the tag compute two n-bit sequences, v^0 and v^1, using a pseudorandom function (typically a MAC algorithm, a hash function, etc.).

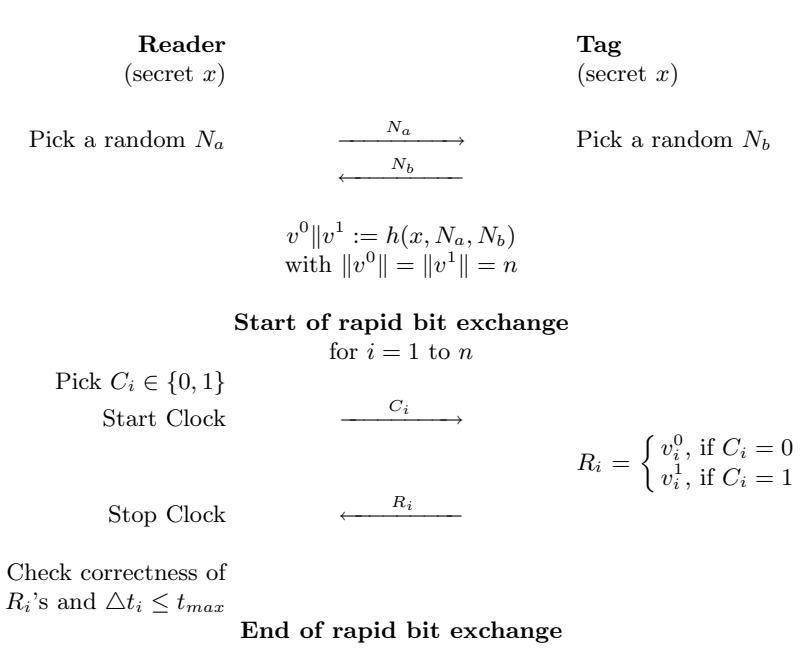


Fig. 2. Hancke and Kuhn's protocol

Then (and repeating this step n times) the reader sends a random bit. Upon receiving a bit, the tag sends back a bit from v^0 if the received bit C_i equals 0. If C_i equals 1, then it sends back a bit from v^1. After n iterations, the reader checks the correctness of R_i's and computes the propagation time. In each round, the probability that adversary sends a correct response is not $\frac{1}{2}$ but $\frac{3}{4}$. This is because the adversary could slightly accelerate the clock signal provided to the tag and transmit an anticipated challenge C_i' before the reader sends its challenge C_i. In half of all cases, the adversary will have the correct guesses, that is $C_i' = C_i$, and therefore will have obtained in advance the correct value R_i that is needed to satisfy the reader. In the other half of all cases, the adversary can reply with a guessed bit, which will be correct in half of all cases. Therefore, the adversary has $\frac{3}{4}$ probability of replying correctly.

Since this protocol's publication, several solutions have been proposed to improve its effectiveness and/or enhance its functionalities.

A solution to reduce the aforementioned probability below $\frac{3}{4}$ is to include a signing message (or message authentication code) as used in other protocols [2,14,15]. However a signing message could not be sent with the channel for fast bit exchanges as it is very sensitive to the background noise. It should be sent by normal communication method with error detection or correction technique. Therefore this approach would put an overhead on computation of a tag as well as communication, which causes the protocol to be slower.

In 2006, Munilla et al. modified the Hancke and Kuhn protocol by applying "void challenges" in order to reduce the success probability of the adversary

[10]. Their protocol is the first and only approach not using any additional signing message to reduce the success probability of the adversary. However the disadvantage of their solution is that it requires three (physical) states: 0, 1, and *void*, which is practically very difficult to implement.

HKP is vulnerable to the terrorist fraud attack but this can be solved, as proposed by Reid et al. in [13], making the bit-strings, v^0 and v^1, and the long-term key x intermingled ($v^0 = Enc_{v^1}(x)$). Thus, if a legitimate tag wants to reveal the secret, then it will allow the adversary to impersonate it in more than a single run of the protocol. However, Reid et al.'s protocol does not provide privacy as it sends identities without any protection. Furthermore, as described by Piramuthu [12], the probability of the success for an attack is higher than for HKP.

In 2007, Tu and Piramuthu proposed a protocol to reduce the success probability of an adversary [15]. They used four for-loop iterations for fast bit exchanges, which made the success probability of an adversary equal to $(9/16)^n$. That is, the reader sends different hash values after $n/4$-bit exchanges. They also used the combination of "$v^0 = v^1 \oplus x$" to prevent terrorist fraud attack. However, we will show in Section 3 that their protocol is in fact not secure against an active adversary.

Capkun et al. extended Brands and Chaum's protocol to mutual authentication, so called MAD (mutual authentication with distance-bounding) in 2003 [4]. However their protocol is not resilient to bit errors during the fast bit exchanges.

Singelée and Preneel proposed a noise resilient protocol in 2007 [14]. They used error correcting code (ECC) and MAC for the sake of channel error resistance, but this made the protocol slower.

Recently Nikov and Vauclair proposed a protocol [11]. They used more than a bit for fast exchanges. However it is susceptible to channel noise. Furthermore the tag needs to compute $2k$ secret key functions (HMAC or AES) and store the result.

Finally, Waters and Felten [16] and Bussard and Bagga [3] proposed distance bounding protocols using public key cryptography respectively. However the adoption of public key cryptography in a small device such as low-cost RFID is not applicable yet.

3 New Attack on Tu and Piramuthu Protocol

We show in this section an attack against the protocol described by Tu and Piramuthu [15].

3.1 The Protocol

The protocol is depicted in Figure 3 (for convenience, we use the same notations as in the original paper). The Reader first generates a nonce r_B and sends $x \oplus r_B$ to the tag, where x is a shared long-term secret. Similarly, the tag generates a random nonce r_A and sends $x \oplus r_A$. The reader and the tag then derive a common session key $k = h(r_A, x||r_B)$ and use this key to split the secret x in two shares,

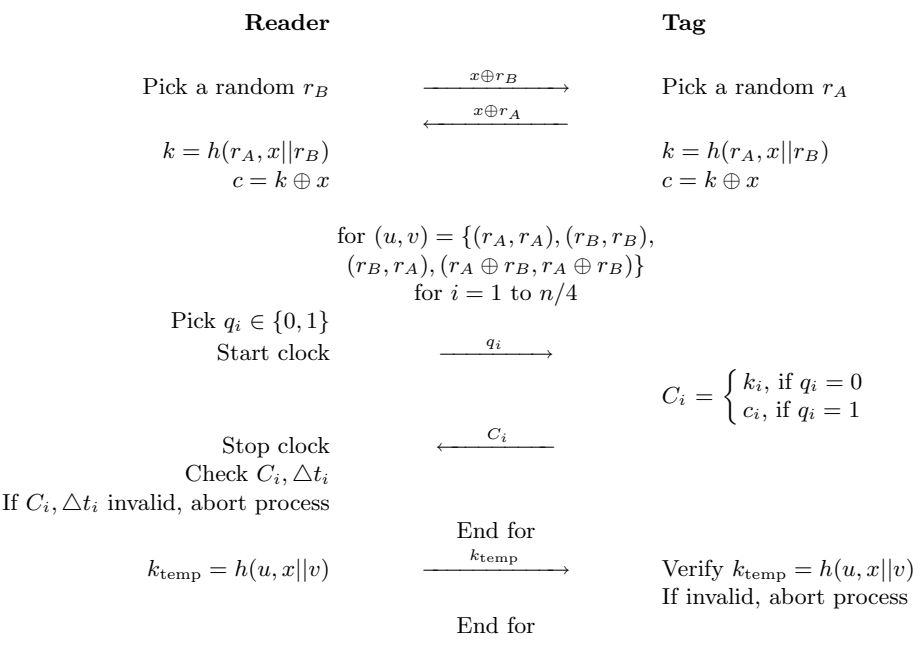


Fig. 3. Tu and Piramathu's protocol

k and $c = k \oplus x$. Then an outer loop is iterated four times, for four different combination values of (u, v), namely $(r_A, r_A), (r_B, r_B), (r_B, r_A), (r_A \oplus r_B, r_A \oplus r_B)$. During each iteration of this outer loop, an inner loop is iterated $n/4$ times. The inner loop is a rapid bit exchange consisting in a challenge bit q_i being sent by the reader and the corresponding answer C_i being sent by the tag, where $C_i = k_i$ (resp. $C_i = k_i \oplus x_i$) if the challenge was equal to 0 (resp. 1). After this inner loop, a reader verification is performed by the tag by having the reader compute and transmit a value $k_{\text{temp}} = h(u, x||v)$, which is then verified by the tag (this verification step is thus performed 4 times, one at each iteration of the outer loop). The idea of this verification step is to have the reader validated by the tag at intermediary steps of the rapid bit exchange, in order to prevent an adversary from sending queries (random) q_i's and retrieving corresponding C_i's in advance. According to [15], this step can also use bit streaming and clocking to measure (on tag's side) the distance between tag and reader.

3.2 Our Attack

We show an attack allowing an adversary to recover the long-term key x. To learn bit x_i of that key, the attacker can, during the fast bit exchange, toggle the value of challenge bit q_i when it is transmitted from reader to tag and leave all other messages untouched. The attacker then observes the reader's reaction. As a matter of fact, if the reader accepts the tag, it means that the tag's answer C_i was nevertheless correct, and thus that $c_i = k_i$. As $c_i = k_i \oplus x_i$, the adversary

can conclude that $x_i = 0$. Similarly, if the reader refuses the tag, the adversary can conclude that $x_i = 1$.

4 Proposed Scheme

4.1 Adversary

We consider an active adversary who entirely controls the channel. That is, she can eavesdrop, intercept, modify or inject messages. She can also increase the transmission speed on the channel up to a given bound. We define this bound as the speed of light. On the other hand, we consider that our adversary cannot correctly encrypt, decrypt, or sign messages without knowledge of the appropriate key. We assume that she has no way to obtain such keys except those of colluding tags.

We assume that the communication protocols are public, enabling an adversary to potentially communicate with a reader or a tag. While communicating with a tag, the adversary is able to increase or decrease its clock frequency and thus the computation speed.

We define a *neighborhood* as a geographical zone around a reader whose limits are clearly defined and publicly known. We consider that a tag present in a neighborhood agrees to authenticate. We say that a tag T has been *impersonated* if an execution of the protocol convinced a reader that it has authenticated T while the latter was not present inside the neighborhood during the said execution. In the same vein, a reader can be impersonated.

4.2 Goals

Authentication. The primary goal of the protocol is to ensure tag authentication, that is, at the end of the execution of the protocol, the reader gets the conviction that it communicates with the claimed entity. Mutual authentication is achieved if the tag also gets the conviction that it communicates with the claimed reader.

Mafia fraud attack resistance. A tag cannot be impersonated, except if it colludes with the adversary.

Terrorist fraud attack resistance. A tag cannot be impersonated, except if it reveals its secret key to the adversary.

Low computation complexity. In order to get a practical authentication delay, the number of cryptographic operations performed by the tag during the authentication process must be as small as possible. Due to their efficiency compared to asymmetric cryptography, the use of symmetric primitives is certainly desirable in this respect.

When several tags are present in the field of the reader, each tag must be singulated through a collision-avoidance protocol before starting the authentication protocol. During this process when tags are powered but mostly idled, or whenever tags can be powered without having to authenticate, they are able to perform some precomputation “for free”.

Low false acceptance rate. In order to get a practical authentication delay, the number of rounds of the fast phase and the total number of bits exchanged between the tag and the reader must be kept as small as possible for a given false acceptance rate. For accuracy reason on the round trip time, we assume that only one bit can be included per message in the fast phase.

Privacy. The protocol should not reveal the tag identifier except to the legitimate reader.[1] Moreover, given any set of recorded protocol executions, only the legitimate reader should be able to determine whether a tag is involved in two or more executions.

Channel error resistance. We assume that the channel used during the slow phase is error-free. This assumption is quite realistic in the sense that there is no specific time-constraint on that channel. An error-correcting mechanism can therefore be used.

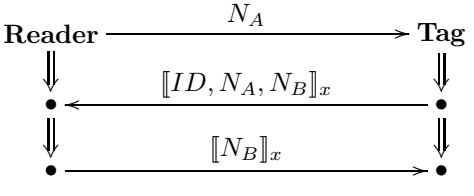
However, the channel used during the fast phase may suffer from Byzantine errors. In such a case, the authentication property must be ensured up to a given error rate threshold. Above this threshold, the reader must abort the protocol.

4.3 Description

Our authentication protocol is based on the MAP1 protocol of Bellare and Rogaway [1], in the MAP1.1 variant proposed by Guttman et al. [6]. To this protocol, which provides mutual authentication, a rapid bit exchange step has been added in order to achieve distance-bounding, and some cleartext information has been removed to ensure the privacy of the tag.

The MAP1.1 protocol works as follows. Tag and reader are assumed to share a secret key x.

1. The reader chooses a random nonce N_A and transmits it to the tag.
2. The tag chooses a random nonce N_B and transmits $[\![ID, N_A, N_B]\!]_x$ to the reader, where $[\![m]\!]_x$ means $m \| f_x(m)$, f is a pseudorandom function (PRF), x is the key of the tag, and ID is the concatenation of the reader's and tag's identifier.
3. The reader computes $[\![N_B]\!]_x$ and transmits it to the tag. Note that this extra step is only required if mutual authentication must be achieved. If it is not necessary for the tag to authenticate the reader, this last step can be discarded.



[1] Note that this is in fact not the most stringent notion of privacy that can be considered. A stronger notion, in which even the reader does not learn the identity of the tag, can also be useful in some contexts.

Taking this protocol as our starting point, four adaptations are proposed.

1. Since tags do not make any difference between the readers, we simply represent the identity of teh readers as the empty string.
2. Since we want to preserve the privacy of the tags, we do not transmit their identity in clear. The reader will then need to access a database storing the identity and key (x, ID) of each tag and to perform an exhaustive search over this DB, trying all possible keys until a match is found.[2]
3. Since honest tags and readers are not involved in concurrent sessions, we can also avoid repeating the transmission of nonces in clear after their initial transmission, so N_A (resp. N_B) does not need to be transmitted in clear during the second (resp. third) round.
4. Since we want our protocol to be distance bounding, a rapid bit exchange phase is added. The role of this phase is to prove to the reader that it is directly interacting with the tag, preventing relay attacks.

It can be observed that the first three adaptations do not have any impact on the authentication properties of the protocol, keeping the analyzes of [1,6] valid. The last adaptation will be designed in such a way that it does not interfere with the other parts of the protocol.

Basic version. We first describe a basic version of our protocol and discuss its security. A more efficient variant is discussed in Section 5.2.

First a preparation phase is performed, involving the generation of nonces, one application of the PRF and a few XORs. We will discuss below how precomputing can be used for low-resource devices. No delays are measured during this phase.

- Following the MAP1.1 protocol, the reader chooses a random nonce N_A and transmits it to the tag.
- The tag chooses a random N_B and computes a temporary key $a \coloneqq f_x(C_B, N_B)$ using its permanent secret key x and N_B (here C_B is just a system-wide constant).
- The tag splits his permanent secret key x in two shares by computing $Z^0 := a, Z^1 := a \oplus x$.
- The tag transmits N_B to the reader (which constitutes the first part of the second message of the MAP1.1 protocol).

After this preparation, the rapid bit exchange phase starts. This phase is repeated n times, with i varying from 1 to n, and the challenge-response delay is measured for each step. As explained in Section 2.2, this communication goes over a channel that does not contain any error detection or correction mechanism, so we must take into account the fact that channel errors might occur (either randomly or by action of the attacker) in this phase. Moreover, the protocol

[2] Note that the protocol is always initiated on reader's side, so that a reflection attack, in which a genuine answer from one execution of the protocol is used by an attacker in another one, is not possible here.

must involve as few tag operations as possible in this phase, and we make this number of operations fairly minimal: in each round, the tag only needs to select one out of two pre-computed bits.

- The reader chooses a random bit c_i, starts a clock and transmits c_i to the tag. We will denote by c'_i the (possibly incorrectly transmitted) value received by the tag.
- The tag answers by $r'_i := Z_i^{c'_i}$. We will denote by r_i the value received by the reader.
- Upon receiving r_i, the reader stops the clock, stores the time delay $\triangle t_i$ and answer received (note that answer's correctness is not checked at this time), and moves to the next step.

After the rapid bit exchange phase, the final phase begins. This phase also involves significant computing overhead, and no delays are measured during it.

- The tag computes $t_B := f_x(c'_1, \ldots, c'_n, ID, N_A, N_B)$ and transmits t_B and the challenges $c'_1, \ldots, c'_n$ it received during the rapid bit exchange phase. Together with the sending of N_B that took place earlier, this is the second round of the MAP1.1 protocol, with the addition that t_B also authenticates the challenges $c'_1, \ldots, c'_n$ received during the rapid bit exchange phase.
- The reader performs an exhaustive search over its tag database until it finds a pair (ID, x) such that $t_B := f_x(c'_1, \ldots, c'_n, ID, N_A, N_B))$.
- The reader computes the values Z^0 and Z^1.
- The reader checks the validity of the responses made during rapid bit exchange phase, i.e.:
  - it counts the number err_c of positions for which $c_i \neq c'_i$;
  - it counts the number of positions err_r for which $c_i = c'_i$, but $r_i \neq Z_i^{c_i}$;
  - it counts the number of positions err_t for which $c_i = c'_i$ and $r_i = Z_i^{c_i}$, but the response delay $\triangle t_i$ is above the time threshold t_{max};
  - if $err_c + err_r + err_t$ is above the fault tolerance threshold T, authentication fails and the protocol aborts.
- The reader computes $t_A := f_x(N_B)$ and transmits it to the tag (this is the last step of the MAP1.1 protocol).
- The tag checks the correctness of t_A. As stated before, the last two steps are only required if mutual authentication must be achieved. If it is not necessary for the tag to authenticate the reader, they can be omitted. Still, in both cases, privacy is guaranteed.

5 Analysis

5.1 Security

To make the security discussion easier, let us first ignore the fault tolerance parameter. That is, we will consider that the threshold T is 0. The effect on security of a larger threshold will be discussed in Section 5.3.

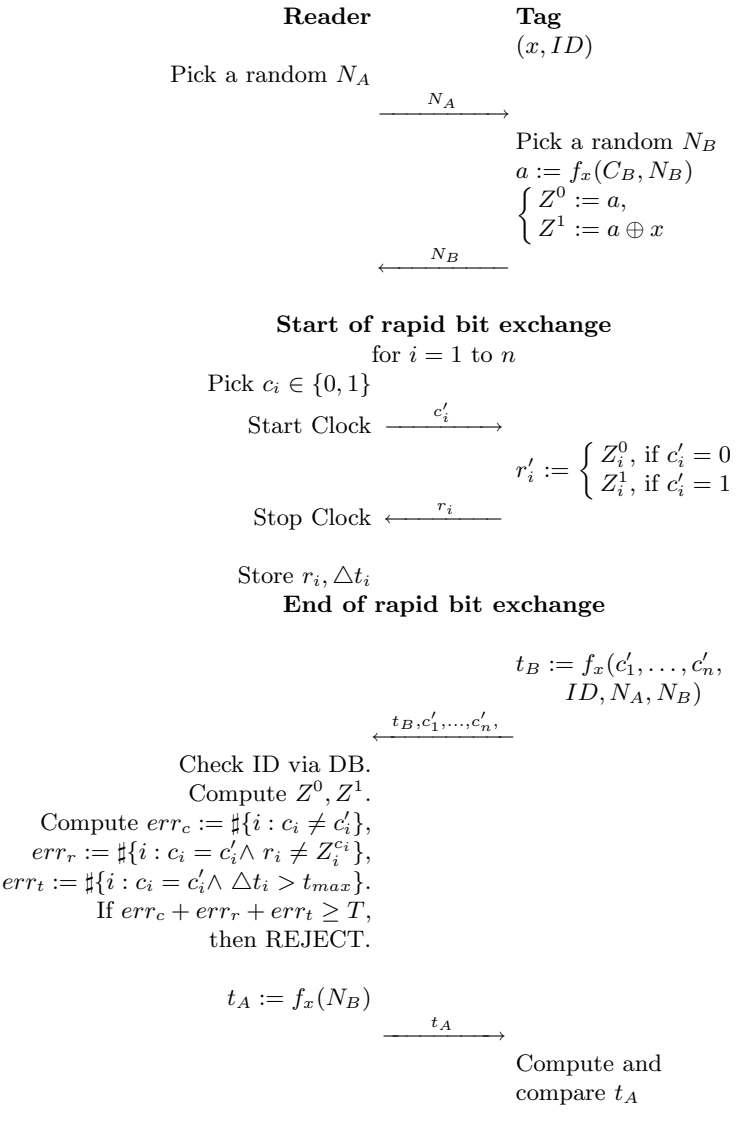


Fig. 4. Our basic authentication protocol secure against relay attacks

Authentication. The security of the basic authentication protocol has been studied in [1] and [6]. Basically, the presence of N_A – a fresh nonce generated by the reader – in the input of the PRF f authenticates the tag to the reader (as only the tag and the reader know the value x used to key f). Similarly, the presence of N_B as an argument of f_x in $f_x(N_B)$ guarantees the tag that it was successfully authenticated by the reader. We refer to the aforementioned papers for more details.

Let us show that our modifications to MAP1.1 do not modify the security of the protocol. We did the following modifications to MAP1.1:

- some values that were transmitted in clear in MAP1.1 (e.g., the tag and reader ID) are not transmitted anymore;
- the values $(c_1, \ldots, c_n)$ are additionally transmitted in clear during the rapid bit exchange phase, and included as additional argument of the function f.

It is obvious that removing the cleartext transmissions from the protocol cannot harm security.

The $(c_1, \ldots, c_n)$ bits do not depend on any secret parameter involved in the protocol, and they are transmitted in the authenticated text transmission mode proposed for the MAP1 protocol [1].

Considering the rapid bit exchange, we will first argue that a passive observation of the rapid bit exchange does not reveal any information on x. If we consider the PRF as a random oracle, a can be seen as a pure random string, and the construction of Z^0, Z^1 is a classical secret sharing of x. As only one share of each bit is revealed during the rapid bit exchange, no information on x is disclosed by the responses r_i. Besides, no information on x can be revealed by the challenges c_i, as these do not depend on x.

Things get a bit different when we consider an *active* adversary, allowed to manipulate the messages exchanged during the rapid bit exchange. As in the attack described in Section 3, if the reader checked for the correctness of r_i without verifying the value of the challenge bits used by the tag (using t_B), one easy attack would be for the adversary to flip one bit c_i during transmission from reader to tag, and then to simply forward the answer of the tag to the reader. If the authentication is successful, the attacker can conclude that $Z_i^0 = Z_i^1$ and hence that $x_i = 0$; if the authentication is unsuccessful, she can similarly conclude that $x_i = 1$. This is the reason why we authenticate the c_i' and place the verification steps after the reception and verification of t_B. More precisely, we protect against active attackers by ensuring that verification is only performed on bits for which the challenge was not manipulated by the tag. In this way,

- tampering with c_i does not reveal any additional information, as the corresponding answer will simply be ignored,[3] and
- of course, tampering with answer r_i does not reveal any information either.

Tampering with the messages can of course make the protocol fail by DoS, but, as the attacker anyway knows he is always turning a correct answer into an incorrect one, he cannot gain any information in this way. We have thus showed that authentication is well achieved by our protocol.

[3] Of course, the adversary will be able to choose which share, Z_i^0 or Z_i^1 will be revealed there, but we already showed that obtaining only one share does not compromise the secret. Also note that, although the adversary is able to modify the challenges c_i during the rapid bit exchange phase, the presence of t_B prevents her from reflecting this modification in the message sent by the tag, so that the reader will learn which values were incorrectly received.

Terrorist fraud resistance. We will show that only a tag knowing at least $n - v$ bits of the long-term key x is able to answer the requests issued during the rapid bit exchange phase with success probability at least $(\frac{3}{4})^v$. As discussed in Section 4.1, we will consider that only a tag present in the neighborhood of the reader is able to answer the requests c_i in due time (without the possibility to obtain assistance from any device not present in the neighborhood), and show that this tag must know x. Considering that, at step i of the protocol, the tag will have to respond Z_i^0 with probability $\frac{1}{2}$ and Z_i^1 with probability 1/2, a tag ignoring the value of v bits of Z^0 or Z^1 can only succeed with probability $(\frac{3}{4})^v$. From the equalities $Z^0 = a, Z^1 = a \oplus x$, the knowledge of $2n - v$ bits of Z^0, Z^1 immediately yields the knowledge of at least $n - v$ bits of x. This security bound is for example reached if we consider a terrorist attack in which a genuine, but distant, tag transmits the value of Z^0 to his accomplice: although no bit of x has been revealed to the accomplice, she has now $\frac{3}{4}$ chance to answer correctly on each step of the rapid bit exchange.

Mafia fraud resistance. It is worth noting that, as opposed to the Hancke and Kuhn protocol, the attack described in Section 2.2 (anticipated challenge transmission) does not work against our protocol. As a matter of fact, the presence in t_B of the list of challenges received by the genuine tag prevents this attack in our context. Therefore, we argue that the security bound against mafia attacks is $(\frac{1}{2})^n$.

Privacy. We first show that an adversary who does not know the value of x does not learn any information on ID. The only message depending on ID is $t_B = f_x(c_1, \ldots, c_n, ID, N_A, N_B)$. It is easy to show that any adversary who could retrieve any information on the input ID could also distinguish f_x from a random function. As f is supposed to be a PRF, we can conclude that no information on ID is revealed by the protocol. As far as tracking between various protocol executions is concerned, the values $N_B, r_0, \ldots, r_n, t_B, c_0, \ldots, c_n$ observable by an attacker are either random or the output of a PRF with fresh input and unknown key. It is thus clear that all of them appear as random values to the attacker, and tracking is not possible.

5.2 A More Efficient Variant

Our protocol, in the version we just presented, involves one rapid bit exchange step per bit of the key. In a constrained environment, this communication overhead might be problematic. On the other hand, we cannot for example restrict the rapid bit exchange to the first m bits of Z^0, Z^1, because we would then loose resistance against terrorist attack: a device knowing only the first m bits of the key x would be able to succeed in the rapid bit exchange phase.

This problem could be solved by having the reader challenge the tag on m random bit positions of Z^0, Z^1. However, although very simple, fetching a specific bit from an integer might be a too complex operation for the minimal overhead we expect during the rapid bit exchange phase.

Yet, there is a possible compromise. Basically, the reader could send a list of m bit positions (but not the corresponding challenges) to the tag before the rapid bit exchange phase, enabling the extraction of these positions from Z^0 and Z^1 and preserving a fast treatment during the phase itself. Of course, in a terrorist attack, the genuine device could take advantage of that delay to transmit only the relevant parts of Z^0, Z^1 to his accomplice, revealing her only m bits of x. Yet, as the list changes for each authentication, m random bits of x would have to be revealed for each authentication, so that the full key would be quickly revealed.

The full protocol is depicted in Figure 5.2. Below we only describe the part that changed compared to the initial protocol.

- The reader chooses a random N_A; it also chooses a random d with hamming weight m. Intuitively, d corresponds to a mask pointing the positions on which the tag will be questioned during the rapid bit exchange. The reader transmits N_A and d to the tag.
- The tag chooses a random N_B and computes $a := f_x(C_B, N_B)$, Z^0 and Z^1 as before. Then it prepares the possible answers by extracting the relevant parts of Z^0, Z^1 according to the mask d, building the m-bit vectors R^0 and R^1.
- The remainder of the protocol is unchanged, except that R^0, R^1 is used instead of Z^0, Z^1.

It can be observed that the N_A and d protocol parameters could actually be merged to save bandwidth, by requiring the tag to use N_A as a mask. However, as the hamming weight of m must be fixed to some value (or range of values) in order to guarantee the appropriate security level for the rapid bit exchange, the set of admissible nonces of n bits is reduced, and the length of N_A must be increased accordingly.

5.3 Fault Tolerance

Our protocol is tolerant to faults occurring during the rapid bit exchange transmissions:[4] if some of the bits c_i, r_i get corrupted during transmission, or get inappropriately delayed, authentication can succeed anyway, provided the percentage of such errors is sufficiently small. Basically the threshold T must be chosen so that the probability for an adversary to be successful on $m - T$ challenges is acceptably small. Taking our most pessimistic context, i.e. terrorist fraud attacks, the chances of success follow a binomial distribution $A := \mathrm{Bi}(m, \frac{3}{4})$ and we want $\Pr[A > m - T] < \epsilon$ for an appropriate security parameter ϵ. For example, taking $m = 30$ rapid bit exchange steps and tolerating up to two errors (i.e. $T = 3$) yields a success probability for the adversary of about 1%. If we consider only resistance against mafia fraud attacks (so $A := \mathrm{Bi}(m, \frac{1}{2})$) and take $m = 20$ rapid bit exchange steps, then we can tolerate up to 4 errors and still have less than 1% fraud probability (or tolerate only one error and have the probability shrink to 0.01%).

[4] As stated before, we assume that other transmissions occur over a channel capable of error detection.

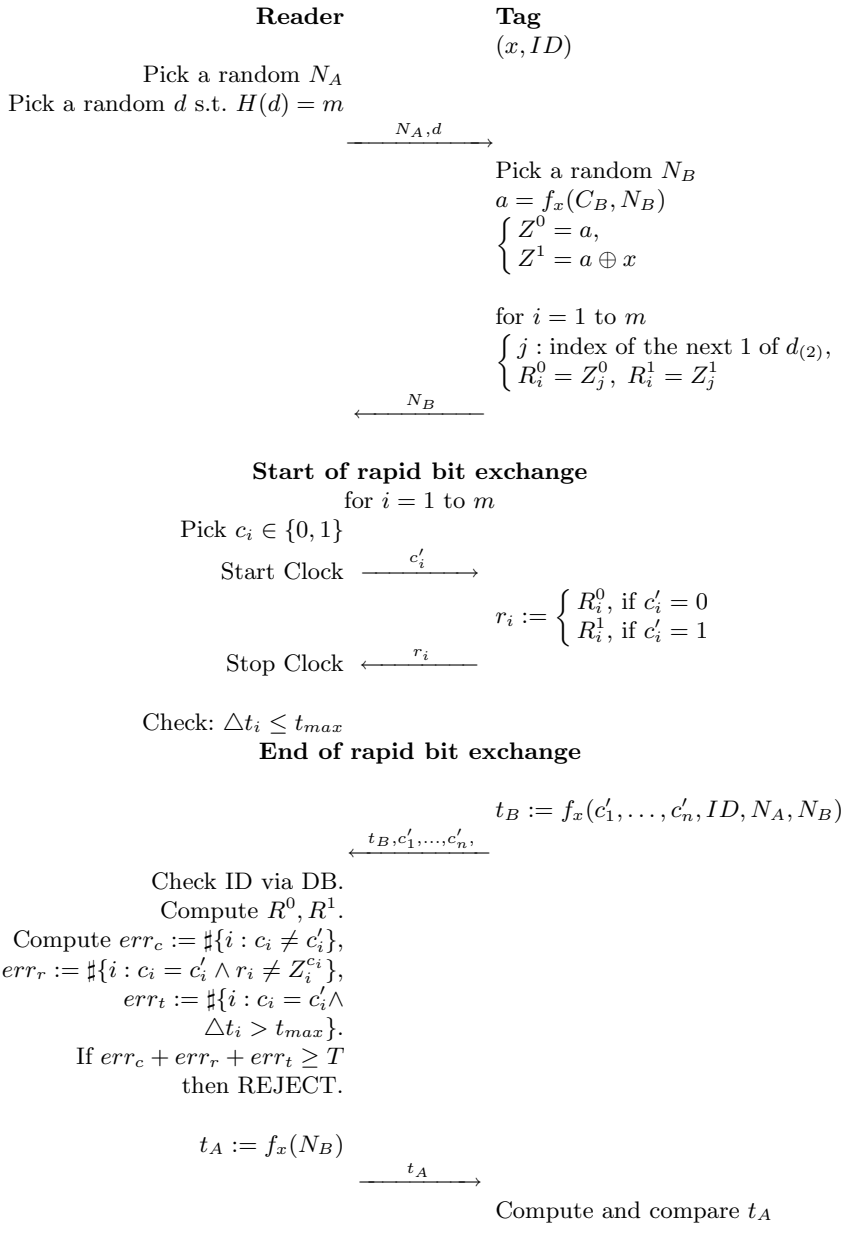


Fig. 5. A more efficient variant of the protocol

5.4 Efficiency

Let us consider the amount of computation to be performed by the most constrained device involved in the protocol, i.e. the tag. The most time consuming

part of the protocol is the computation of pseudo-random functions f. As shown in Fig. 4, the function f is used three times on the tag side: in computing a, t_B and t_A. If we need not mutual but unilateral authentication from tag to reader, we need just two computations, a and t_B.

As in [8], we can construct an RFID system which allows a precomputation in a tag. The contents of the input for the computation of a of our proposed protocol do not have any information from the reader. Therefore a can be computed before starting the protocol. Then we need two computations of pseudo-random functions to achieve mutual authentication (one if we need unilateral authentication only).

As it involves an exhaustive search in a key database, the workload on reader's side is significantly higher, and grows linearly with the number of keys deployed in the system. To the best of our knowledge, there is no existing method providing better performance without sacrificing some security, and this is certainly an interesting subject for further research. In particular, we note that most of the protocols discussed in Section 2.2 simply consider that the reader knows the identity of the tag it is questioning, or transmits this identity in clear during the protocol. Clearly, a variant of our scheme in which a single key is shared throughout the system would not have this computing overhead while still preserving privacy regarding outsiders (i.e. without knowledge of the key).

6 Protocols Comparison

Table 1 compares the proposed protocol with previous ones on several points of view: mafia fraud and terrorist attack resistance, error resistance, privacy preservation, mutual authentication and computational overhead inside the most restricted resource, i.e. the tag.

Regarding the security against mafia fraud attack, we compare the success probabilities for an adversary, in other words, false acceptance ratio against mafia fraud attack (M-FAR). This is the probability that the reader accepts the adversary as a legitimate tag. Although Reid et al. claim the M-FAR of their protocol to be $(3/4)^n$, Piramuthu later showed it to be equal to $(7/8)^n$.

The security against terrorist fraud attack and its success probability for an adversary (T-FAR) are compared in a similar way.

Then we compare the resilience against channel errors. This resilience is pretty important for protocol's robustness, as fast bit exchanges are typically sensitive to channel errors.

As far as privacy is concerned, Reid et al.'s protocol discloses identities in cleartext during protocol execution, and is thus not privacy-preserving. Most of the other protocols assume that the reader knows the identity and secret key of the tag before starting distance bounding protocol, hence ignoring the privacy issue or assuming a single secret is shared by all tags. Our protocol allows the reader to learn the tag's identity during execution, although we admit the corresponding overhead is pretty high, since an exhaustive search among all possible keys is necessary for this identification.

Table 1. Comparison of distance bounding protocols

	Mafia	M-FAR	Terrorist	T-FAR	Err. resis.	Privacy	MA	Comp.
BC [2]	Yes	$(1/2)^n$	No	-	No	-	No	2
HK [7]	Yes	$(3/4)^n$	No	-	Yes	-	No	1
Reid et al. [13]	Yes	$(7/8)^n$	Yes	$(3/4)^v$	Yes	No	No	2
SP [14]	Yes	$(1/2)^n$	No	-	Yes	-	No	1 + ECC
Capkun et al. [4]	Yes	$(1/2)^n$	No	-	No	-	Yes	4
NV [11]	Yes	$(1/2)^n$	No	-	No	-	No	$2k$
Proposed (MA)	Yes	$(1/2)^n$	Yes	$(3/4)^v$	Yes	Yes	Yes	3 (2)
Proposed (no MA)	Yes	$(1/2)^n$	Yes	$(3/4)^v$	Yes	Yes	No	2 (1)

We measure the amount of computation needed in the tag as the required number of computation of pseudo-random functions such as hash functions, symmetric key encryptions, etc.[5] We propose our protocol in two flavors, with and without mutual authentication. The number of computations of our protocol is three with mutual authentication and two without it. Additionally, one of these values can be pre-computed in each case (the values between parentheses in Table 1 refer to the number of computations that must be computed on-line).

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
2. Brands, S., Chaum, D.: Distance-Bounding Protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
3. Bussard, L., Bagga, W.: Distance-bounding proof of knowledge to avoid real-time attacks. In: IFIP/SEC (2005)
4. Capkun, S., Buttyan, L., Hubaux, J.-P.: SECTOR: secure tracking of node encounters in multi-hop wireless networks. In: 1st ACM Workshop on Security of Ad Hoc and Sensor Networks – SASN 2003, pp. 21–32 (2003)
5. Desmedt, Y.: Major security problems with the "Unforgeable" (Feige)-Fiat-Shamir proofs of identiy and how to overcome them. In: SecuriCom 1988, pp. 15–17 (1988)
6. Guttman, J.D., Thayer, F.J., Zuck, L.D.: The faithfulness of abstract protocol analysis: Message authentication. Journal of Computer Security 12(6), 865–891 (2004)
7. Hancke, G., Kuhn, M.: An RFID distance bounding protocol. In: The 1st International Conference on Security and Privacy for Emergin Areas in Communications Networks (SECURECOMM 2005), pp. 67–73. IEEE Computer Society, Los Alamitos (2005)
8. Hofferek, G., Wolkerstorfer, J.: Coupon recalculation for the GPS authentication scheme. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 162–175. Springer, Heidelberg (2008)

[5] We note that Singelée and Preneel's protocol (SP) requires additional error correcting codes (ECC), which normally requires significant computation overhead.

9. Munilla, J., Peinado, A.: Distance bounding protocols with void-challenges for RFID. In: Workshop on RFID Security - RFIDSec 2006 (2006)
10. Munilla, J., Peinado, A.: Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels. Wireless communications and mobile computing (2008); published online: January 17, 2008, an earlier version appears in [9]
11. Nikov, V., Vauclair, M.: Yet another secure distance-bounding protocol, http://eprint.iacr.org/2008/319; an earlier version appears in SECRYPT 2008
12. Piramuthu, S.: Protocols for RFID tag/reader authentication. Decision Support Systems 43, 897–914 (2007)
13. Reid, J., Nieto, J.G., Tang, T., Senadji, B.: Detecting relay attacks with timing-based protocols. In: Bao, F., Miller, S. (eds.) Proceedings of the 2nd ACM symposium on Information, computer and communications security, pp. 204–213. ACM, New York (2007), http://eprint.qut.edu.au/view/year/2006.html
14. Singelée, D., Preneel, B.: Distance bounding in noisy environments. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 101–115. Springer, Heidelberg (2007)
15. Tu, Y.-J., Piramuthu, S.: RFID distance bounding protocols. In: The 1st International EURASIP Workshop in RFID Technology, Vienna, Austria (2007)
16. Waters, B., Felten, E.: Secure, private proofs of location. Princeton Computer Science, TR-667-03 (2003)