On the Security of Tan *et al.* Serverless RFID Authentication and Search Protocols

Masoumeh Safkhani[1], Pedro Peris-Lopez[2], Nasour Bagheri[3], Majid Naderi[1], and Julio Cesar Hernandez-Castro[4]

[1] Department of Electrical Engineering, Iran University of Science and Technology (IUST), Tehran, Iran
[2] Computer Security Lab (COSEC), Carlos III University of Madrid, Spain
[3] Department of Electrical Engineering, Shahid Rajaee Teachers Training University, Tehran, Iran
[4] School of Computing, University of Portsmouth, UK

Abstract. In this paper, we analyze the security of the mutual authentication and search protocols recently proposed by Tan *et al.* [20]. Our security analysis clearly highlights important security pitfalls in these. More precisely, privacy location of the tags' holder is compromised by the authentication protocol. Moreover, the static identifier which represents the most valuable information that a tag supposedly transmits in a secure way, can be exposed by an adversary when the authentication protocol is used in combination with one of the search protocols. Finally, we point out how the improved search protocols are vulnerable to traceability attacks, and show the way an attacker can impersonate a legitimate tag.

Keywords: RFID, Mutual Authentication, Search Protocol, Cryptanalysis.

1 Introduction

In RFID systems, readers and tags can employ authentication protocols with the purpose of authenticating each other. These protocols commonly exchange a number of messages between the involved entities. Specifically, one of the protocol entities sends a challenge(s) to the other entity and then it verifies the correctness of the received response(s). To achieve the intended security objectives, readers and tags are often mutually authenticated. Besides authentication, when a reader needs to find a certain tag among a large population of tags, it requires a search protocol. An efficient scheme is difficult to design, and more so if it should be robust enough not to compromise the security and privacy of the system.

Tan *et al.* recently proposed a serverless mutual authentication and a search protocol for RFID systems [20], both heavily based on the use of hash functions. Moreover, three improved search protocols were introduced in their paper. The authors claimed optimal security for both the proposed authentication protocol

J.-H. Hoepman and I. Verbauwhede (Eds.): RFIDSec 2012, LNCS 7739, pp. 1–19, 2013.

and the basic/improved search protocols. Nevertheless, in this article, we show important security pitfalls on these schemes.

Paper Organization: We describe Tan *et al.* protocols in § 2. A traceability attack against the mutual authentication protocol is presented in § 3. In § 4, we analyze the privacy protection of confidential information regarding the mutual authentication protocol when it is used in combination with one of the proposed search protocols. Then, traceability and impersonation attacks against the improved search protocols are introduced in § 5 and § 6, respectively. Finally, in § 7 we extract some conclusions.

2 Protocols Description

The notation used through the paper can be consulted in Appendix A. A complete description of Tan *et al.* mutual authentication protocol, search protocol and improved search protocols is provided from Fig. 1 to Fig. 5. in Appendix B. We now give a brief description of these schemes, but we urge the reader to consult the original paper for further details [20].

Tan *et al.* mutual authentication protocol is divided into two phases, the setup and the mutual authentication phase. In the setup phase, the reader R_i authenticates itself to CA and obtains access to tags $T_1, \cdots, T_n$. R_i will receive L_i where:

$$L_i = \{(f(r_i, t_1), id_1), \cdots, (f(r_i, t_n), id_n)\}$$

In the mutual authentication phase, the reader sends a *request* to the tag. The tag replies with a random value n_j. Then, the reader sends its identifier r_i and a random value n_i to the tag. The tag answers a tuple $\{h(f(r_i, t_j))_m, h(f(r_i, t_j \| n_i \| n_j) \oplus id_j\}$, where n_j represents a random value generated by the tag and $h(f(r_i, t_j))_m$ symbolizes the first m bits of $h(f(r_i, t_j))$. Upon receiving these values, the reader hashes every entry in L_i and checks whether the first m bits match with the received value $h(f(r_i, t_j))_m$. After that, it computes $h(f(r_i, t_j) \| n_i \| n_j)$ and obtains id_j by a simple XOR operation. If the obtained id_j matches with the id_j stored in L_i, the reader authenticates the tag.

Tan *et al.* also proposed several search protocols. In the basic protocol, sketched in Fig.2, the reader broadcasts a triplet containing $\{h(f(r_i, t_j) \| n_r) \oplus id_j, n_r, r_i\}$. Each tag T_j computes $h(f(r_i, t_j) \| n_r)$ and XORs it with the received value $h(f(r_i, t_j) \| n_r) \oplus id_j$ to extract id. If $id = id_j$, the matched tag authenticates the reader and replies a tuple consisting of $\{h(f(r_i, t_j) \| n_t \| n_r) \oplus id_j, n_t\}$. As the authors state in the article, this protocol is vulnerable to traceability attacks (see Section 5 for more details). Motivated by this weakness, the authors proposed three improved search protocols.

In the first of these, depicted in Fig. 3, each RFID tag keeps a record of the recent random values used by the reader. Hence, a RFID tag rejects any query for which the random value n_r exists in its list. Specifically, the authors propose that each tag keeps only the last seen random value – denoted as *oldn*.

The second improved solution consists on a challenge and response scheme (see Fig. 4). The reader broadcasts m bits of id_j ($\{id_j\}_m$), its identifier r_i and a challenge r_i. Any tag on reader range that matches the first m bits of the id will reply to the query. So, depending on m, there could be multiple tags that share the same m bits on id and answer the query. This approach is used to avoid the condition where replying to a query can be used to identify a tag. Nevertheless, this protocol does not work well when the tags' id is structured, e.g. the first several bits of the tags' id represent the product code, the next several bits provide the tag origin, manufacturer and so on.

Finally, Tan *et al.* proposed a third protocol, sketched in Fig. 5, in which the use of noise attempts to mask the origin of the replies. Based on this approach, each tag that receives a search query and does not match the request will reply with some probability – λ. The authors claimed that an adversary cannot track a tag since any tag could potentially reply.

3 Traceability Attack on the Mutual Authentication Protocol

We can find in the literature a significant number of RFID authentication protocols based on hash-functions. Nevertheless, two main drawbacks dissuade authors from its practical use. Firstly, the support on-chip of hash functions may be questioned due to their demand for circuit size, memory and power consumption [7]. Secondly, the search process in the readers – matching between the values received from the tag and the records stored in the reader – often implies heavy computation load.

Tan *et al.* [20] proposed that the tags transmit $h(f(r_i, t_j))_m$ as a mechanism to improve the search time for the reader. In this Section, we evaluate how privacy location can be compromised by using this confidential information. More precisely, we use traceability model proposed by Phan [3], which is a reformulation of the model initially proposed by Juels and Weis [12] (the reader is urged to consult Appendix C for details).

3.1 Proposed Attack

We show how Tan *et al.* protocol puts at stake the privacy location of tags' holders because tags can be tracked with a high probability. Specifically, an adversary A has to follow the steps described below:

- Phase 1 (Learning): A sends an Execute(R_i, T_0, k) query and acquires the public messages passed over the insecure radio channel $\{n_0, n_i, r_i, h(f(r_i, t_0))_m, h(f(r_i, t_0)\|n_i\|n_0) \oplus id_0\}$. Then, $\mathcal{A}$ stores $X = h(f(r_i, t_0))_m$ as an static search index linked to T_0.
- Phase 2 (Challenge): $\mathcal{A}$ chooses two fresh tags $\{T_0, T_1\}$ whose associated identifiers and keys are $\{id_0, t_0\}$ and $\{id_1, t_1\}$, respectively. He then sends a Test(k', T_0, T_1) query. As a result, and depending on a chosen random bit

$b \in \{0,1\}$, A is given an static search index $Y = h(f(r_i, t_b))_m$ from the set $\{h(f(r_i, t_0))_m, h(f(r_i, t_1))_m\}$.

– Phase 3 (Guessing): $\mathcal{A}$ finishes $\mathcal{G}$ and outputs a bit $\widetilde{b}$ as its conjecture of the value b. In particular, $\mathcal{A}$ utilizes the following simple decision rule:

$$\begin{cases} \text{if } X == Y & \widetilde{b} = 0 \\ \text{if } \quad X \neq Y & \widetilde{b} = 1 \end{cases} \tag{1}$$

We emphasize here that only static values are used for the computation of the search indexes X and Y. That is, these ones are independent of the random numbers associated to each session and are unequivocally linked to a particular tag.

In [20], the authors define β as the probability that, given a tag, the probability that when a reader reads in another tag having the same first m bits, the two tags are the same. So, the advantage of an adversary after eavesdropping one authentication session is described below:

$$Adv_A^{\mathsf{UNT}}(q,1) = |(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \beta) - \frac{1}{2}| = \frac{\beta}{2}$$

If $\beta = 1$ the advantage of the adversary is maximal and the smaller the β, the more privacy protection is offered. So, the benefits in the efficiency of the protocol compromise the privacy location. The authors suggested that the way of combating this problem is that tags only answer $h(f(r_i, t_j)\|n_j\|n_j) \oplus id_j$. Nevertheless, their proposal then just turns into another proposal mixed in the huge clutter of authentication protocols based on hash functions and the benefits claimed of being a novel and efficient are ruined. In other words, efficiency in the protocol is only obtained when privacy location is at risk.

3.2 A Note on the Second Improved Search Protocol

In this protocol (see Fig. 4), the authors follow a challenge and response scheme. The reader broadcasts the first m bits of id_j – denoted as $[id_j]_m$ – and the matched tag replies $\{h(f(r_i, t_j)\|n_t) \oplus id_j,\ n_t\}$. The adversary ($A$) can follow the same strategy described in the previous section and track tags. We roughly describe the process to avoid repetition. In the Leaning Phase, A acquires the first m bits of the static identifier ($X = [id_j]_m$) linked to T_0. Then, the challenge phase is executed; depending on a chosen random bit $b \in \{0,1\}$, A captures the first m bits of a static identifier $Y = [id_b]_m \in \{[id_0]_m, [id_1]_m\}$. Finally, in the Guessing Phase, A hypothesizes on b value by using this following decision rule:

$$\begin{cases} \text{if } X == Y & \widetilde{b} = 0 \\ \text{if } \quad X \neq Y & \widetilde{b} = 1 \end{cases} \tag{2}$$

In the original protocol, Tan *et al.* assume that several tags can share the same $[id]_m$. For that reason, they emphasize that the proposed solution does not work

well for tags with an structured id. Under this condition, we consider the case where each tag is assigned a random id, which complies with the authors' requirements. If we asumme that we have a population of N tags and the reader sends $[id]_m$, $\frac{N}{2^m}$ tags are expected to share the same $[id]_m$. Nevertheless, given T_0 with $[id_0]_m$, any randomly selected tag T_1 satisfies $[id_0]_m = [id_1]_m$ with the probability of 2^{-m}. Hence, we can define $\gamma = \frac{1}{2^m}$ as the probability that, given a tag, the probability when the reader reads another tag having the same first m bits, the two tags are different (i.e. $\gamma = \beta - 1$ according definition of the above section). The adversary fails in her traceability attempt when T_1 is randomly chosen at the Challenge-Phase and it shares the same m-bits of the static identifier with T_0 (i.e. $[id_0]_m = [id_1]_m$). So, after conducting the above attack, the adversary advantage is:

$$Adv_A^{\mathsf{UNT}}(q,1) = |(\frac{1}{2} + \frac{1}{2} \cdot (1-\gamma)) - \frac{1}{2}| = \frac{1}{2} \cdot (1-\gamma) \leq \frac{1}{2}$$

To determine the lower bound of the adversary's advantage in tracking a tag, we consider the other side of the problem where the $[id]_m$ values are not random. On the other hand, according to the authors' assumption, all tags do not share the same $[id]_m$. Hence, the adversary can select its target tag T_0 such that given $[id_0]_m$ and N as the number of tags in its range, at least $\frac{N}{2}$ of the tags have different $[id]_m$ compared to $[id_0]_m$. Therefore, dividing the tags into two separated groups, based on their $[id]_m$, the adversary can select its target tag from the group in minority. Hence, we can define $\gamma \leq \frac{1}{2}$ as the probability that, given a selected tag, the probability when the reader reads another tag having the same first m bits, the two tags are different (i.e. $\gamma = \beta - 1$). The adversary fails in her traceability attempt when T_1 is randomly chosen at the Challenge-Phase and it shares the same m-bits of the static identifier with T_0 (i.e. $[id_0]_m = [id_1]_m$). So, after conducting the above attack, the adversary advantage for the group of the tags that are in the minority considering its $[id]_m$ is:

$$Adv_A^{\mathsf{UNT}}(q,1) = |(\frac{1}{2} + \frac{1}{2} \cdot (1-\gamma)) - \frac{1}{2}| = \frac{1}{2} \cdot (1-\gamma) \geq \frac{1}{4}$$

Summarising both cases, if $m > 0$ – and $0 < \gamma < 1$ consequently – the advantage of the adversary is significant and the privacy location compromised. More precisely, the adversary's advantage is bounded as below:

$$\frac{1}{4} \leq Adv_A^{\mathsf{UNT}}(q,1) \leq \frac{1}{2}$$

4 *id* Disclosure Attack

In this section we analyze the security of the mutual authentication protocol proposed by Tan *et al.* [20] when an RFID system uses this mutual authentication protocol combined with one of the search protocols presented in Section 2 –

except the second improved search protocol sketched in Fig 4. We show how an attacker can disclose the most valuable information stored on tags memory, the unique identifier id. To success in our attack, we make the following assumption about the hash function used in the authentication and search protocols. We assume the usage of an iterated hash function based on Merkle-Damgård (MD) [4,15] with a compression function belongs to a group of PGV compression functions [18] for which the message block M_i is used as the key in the underlying block cipher E.

It must be noted that our assumption on the hash function does not compromise the assumption made in the original paper [20]. More precisely, the authors consider $h(x)$ as a one-way hash function, which is not contradiction of what we assume in our analysis. More over, the construction assumed, that is MD, is very well-known and used in popular hash functions such as MD5 or SHA family. On the other hand, it is a common approach to analyze the security of a provable secure hash-based scheme with an ideal hash function by instantiating it with a real hash function. For instance, in [9] Gauravaram and Knudsen shown an approach to use the Dean's method of finding expandable messages [6,13] for finding a second preimage in the Merkle-Damgård hash function to forge a signature scheme based on a RMX-hash function [10] which uses the Davies-Meyer compression functions.

In Appendix D the hash function model is introduced in order to facilitate the understanding of our proposed attack.

4.1 Proposed Attack

In this section we consider the mutual authentication protocol and the basic search protocol (see Fig. 1 and Fig. 2, respectively). To disclose the id, when the hash function uses MD construction with Davies Mayer (DM) [5] one-way compression function, the adversary (A) does as follows:

1. A eavesdrops on a search session between R_i and T_j and records the query $(X = \{h(f(r_i, t_j)\|n_r) \oplus id_j,\ n_r,\ r_i\})$ sent by R_i.
2. A supplants R_i in a mutual authentication session:
   (a) A starts the protocol by sending a *request* to T_j.
   (b) T_j replies with a random value n_j.
   (c) A sends {X[2], X[3]}, where $X[i]$ symbolizes the i-th value of X.
   (d) A captures the response of the tag $(Y = \{h(f(r_i, t_0))_m, h(f(r_i, t_j)\|n_r\|n_j) \oplus id_j\})$.
3. A knows $X[1] = h(f(r_i, t_j)\|n_r) \oplus id_j$ and $Y[2] = h(f(r_i, t_j)\|n_r\|n_j) \oplus id_j$ and the random numbers $\{n_r, n_j\}$.
4. Taking into account the MD structure and DM construction – our initial assumption –, A uses $X[1]$ and $Y[2]$ to disclose id_j as follows:

a) A computes the XOR between the two values captured:

$$\begin{aligned} & X[1] \oplus Y[2] \\ &= h(f(r_i, t_j)\|n_r) \oplus h(f(r_i, t_j)\|n_r\|n_j) \\ &= h(f(r_i, t_j)\|n_r) \oplus E_{n_j}(h(f(r_i, t_j)\|n_r)) \\ & \quad \oplus h(f(r_i, t_j)\|n_r) \\ &= E_{n_j}(h(f(r_i, t_j)\|n_r)) \end{aligned}$$

b) A obtains $h(f(r_i, t_j)\|n_r)$ value:

$$h(f(r_i, t_j)\|n_r) = E_{n_j}^{-1}(X[1] \oplus Y[2])$$

c) Finally, A discloses the static identifier id_j:

$$\begin{aligned} id_j &= h(f(r_i, t_j)\|n_r) \oplus X[1] \\ &= E_{n_j}^{-1}(X[1] \oplus Y[2]) \oplus X[1] \end{aligned}$$

So, the adversary can disclose the static identifier of a tag after observing an execution of the search protocol and impersonating a reader in a session of the mutual authentication scheme. Then, just by computing XOR operations and a decryption in which the key is known, the static identifier is revealed, compromising the privacy information.

Remark 1. A similar attack can be executed to disclose the id_j of T_j when the mutual authentication is used in conjunction with the first and the third improved search protocols, sketched on Fig. 3 and Fig. 5 respectively.

Remark 2. A similar attack can be run to disclose the id_j of T_j when the underlying hash function is used with other one-way compression functions that according [2] are indexed by 6, 7 and 8 in Table 2 in Appendix D.

Remark 3. The above attack may not work if the hash function is used along with *MD strengthening* [14]. However, the usage of MD strengthening in this application could be an unrealistic assumption because efficiency is a vital requirement and we only work with messages of length at most three blocks, assuming that $f(r_i, t_j)$ has the same length as n_j, n_r and the message block length of compression function. We note that, for the given protocols, any call to the hash function are of the form $(h(f(r_i, t_j))$, $(h(f(r_i, t_j)\|n_r)$, $(h(f(r_i, t_j)\|n_r\|n_j)$ or $(h(f(r_i, t_j)\|n_j\|n_r)$.

5 Traceability Attack on Search Protocols

Besides RFID authentication protocols, other schemes for performing different RFID operations are used. Searching protocols are one of these mentioned operations and facilitate the seeking of an specific tag from a large population of tags. To provide privacy and security, the scheme has to fulfil two requirements:

1) RFID tags have to authenticate readers before answering; 2) RFID readers have to make sure than only genuine tags receive and understand its query. In other words, tags only have to answer to authenticated readers and readers only have to query authenticated tags.

In Fig. 2, the basic search protocol is sketched. As the authors state in the original article [20], the scheme does not offer protection against traceability attacks. The traceability attack made reference here is quite different from the one introduced in Section 3. Basically, the ultimate aim that an attacker pursues here is the detection of the presence or absence of an specific tag. We now describe how an attacker successes against the basic search protocol. First, the adversary (A) eavesdrops on an execution of the protocol between a reader and a group of tags – A only has to detect and capture the values from the query and the answer. Then, A replies the captured query, the target tag answers – query is legitimate – and finally A captures the answer. Although the obtained value is different from the previous one – result of using the nonce n_t – A can detect the presence/absence of the target tag because only the pursued tag knows the secret information $\{t_j, id_j\}$ necessary to check the legitimacy of the query and generate a valid answer. The attack can be extended by the information obtained by physical observation. For that purpose, A isolates each tag in the group, replies the captured query and waits for the answer.

In [20], several improved search protocols are proposed to minimize the impact of this sort of tracking. Nevertheless, the authors fail in their attempt because the new versions are insecure as the basic protocol. More precisely, in this section we present traceability attacks on the first and the third improved search protocols (see Fig. 3 and Fig. 5 for details) and concerning the second improved protocol an attack was already presented in Section 3.2.

5.1 Traceability Attack on the First Improved Search Protocol

To avoid the reply of previous captured messages and facilitate the success of traceability attacks, the reader has to be forced to use a different random number n_r for each new query. RFID tags can keep a record of the recent challenges used to accomplish this task. More precisely, the authors considered enough that tags only store the last random number used – denoted as *oldn* in Fig. 3. The authors claimed that an adversary can not track a tag because the adversary needs two successful queries. Nevertheless, this claim is incorrect and an adversary (A) can exploit the symmetric of the protocol (query/reply). To mount a traceability attack, A does as described below:

1. A eavesdrops a transaction between R_i and T_i and captures the query $\{X = h(f(r_i, t_j)\|n_r) \oplus id_j,\ n_r,\ r_i\}$ and the answer $\{Y = h(f(r_i, t_j)\|n_t) \oplus id_j,\ n_t\}$ transmitted over the insecure radio channel.
2. A generates a legitimate query from the captured information: $\{Y, X[3]\} = \{h(f(r_i, t_j)\|n_t) \oplus id_j,\ n_t,\ r_i\}$, where $X[i]$ represents the i-th element of X. Specifically,
   (a) From the captured query, the identifier of the reader is copied ($r_i = X[3]$).

   (b) From the captured reply, a fresh authentication token ($Y = \{h(f(r_i, t_j) \| n_t) \oplus id_j,\ n_t\}$) using a new random number n'_r is obtained (i.e. $n'_r = n_t$).
3. To trace T_i, A replies $\{Y, X[3]\} = \{h(f(r_i, t_j) \| n'_r) \oplus id_j,\ n'_r,\ r_i\}$, where $n'_r = n_t$.
4. If any tag T^* answers to the adversary, A detects the presence of T_j again. As the query/answer is based on the knowledge of the private information $\{t_j, id_j\}$ linked unequivocally with one tag, only the target tag could check the query and send the answer.

Alternatively, an adversary (A) can exploit the fact that only one random number is stored on tags memory. To conduct a traceability attack, the next steps are followed by A:

1. A attacks a transaction between R_i and T_i:
   (a) A captures the query ($X = \{h(f(r_i, t_j) \| n_r) \oplus id_j,\ n_r,\ r_i\}$) sent by R_i and stores this value.
   (b) A frustrates the successful execution of the protocol by altering T_i answer to some random value.
2. As consequence of the above error, the reader repeats the search and A does not interfere in the protocol.
   (a) R_i sends query $X' = \{h(f(r_i, t_j) \| n'_r) \oplus id_j,\ n'_r,\ r_i\}$.
   (b) T_j checks X', updates $oldn = n'_r$ and replies $Y' = \{h(f(r_i, t_j) \| n'_t) \oplus id_j,\ n'_t\}$
3. To trace T_i, A replies X which uses a fresh n_r random value.
4. If any tag T^* answers to the adversary, A detects the presence of T_j again. In fact T_j answers – assuming it still presents there– because X is a valid token and a fresh value ($n_r \neq oldn (= n'_r)$)

Summarizing, following one of these strategies, an attacker puts at stake the privacy location. The risk is indeed maximal since the success probability is 1 and the complexity of running the attack is negligible. The attacks can be avoided by following well-known guidelines for designing cryptographic protocols. For instance Principles 4 and 5 in Abadi and Needham's guidelines are broken [1].

5.2 Traceability Attack on the Third Improved Search Protocol

The last solution the authors proposed to avoid traceability attacks consists on using noise to mask the reply of the target tag. In the protocol, each tag which receives a query and mismatches the request ($id \neq id_j$) will reply with some probability (λ). The authors stated that an adversary (A) does not obtain useful information to track a tag by replying a previous query since any tag could reply. Nevertheless, we show how is possible to conduct an efficient traceability attack against this protocol. The main observation for this attack is that when R_i searches T_j, then the target tag T_j will answer with probability 1 while any mismatched tag will reply with probability λ. To mount a traceability attack, assuming a population of N tags in the range, A does as follows:

1. A eavesdrops a transaction between R_i and T_j and captures the query ($X = \{h(f(r_i, t_j)\|n_r) \oplus id_j,\ n_r,\ r_i\}$).
2. A checks the presence of the target tag N_s times. Specifically,
   For $s = 1$ to N_s:
   (a) A replies the query X.
   (b) A counts the number of answered obtained, where c represents the counter.
      i. If T_j is present, it answers $Y = \{h(f(r_i, t_j)\|n'_t) \oplus id_j,\ n'_t\}$. The tag detects an answer and increments the counter ($c = c + 1$).
      ii. For the rest of the tags ($N - 1$), each tag answers $Y = \{rand, n'_t\}$ with probability λ. When a tag replies, A detects the answer and increments the counter ($c = c + 1$).
3. Finally, A uses this simple but quite effective decision rule:

$$\begin{cases} \text{If } c \geq Ns + (N-1) \cdot Ns \cdot \lambda & T_j \text{ is present} \\ \text{If } c < Ns + (N-1) \cdot Ns \cdot \lambda & T_j \text{ is not present} \end{cases}$$

On the above algorithm, at each sending of a query a mismatched tag answers with a probability of λ while the target tag replies with a probability of 1. If T_j is not present and A repeatedly replies N_s queries, A obtains on average $(N-1) \cdot N_s \cdot \lambda$ answers. When T_j is present, the above mentioned value will be $N_s + (N-1) \cdot N_s \cdot \lambda$. Hence, $(N-1) \cdot N_s \cdot \lambda$ can be interpreted as the average value of the "noise" inserted by the mismatched tags. So, the attacker running the above attack can track the target tag, just by counting the number of answers received. Nevertheless, to determine the success probability, we should consider the possible errors:

- $Error_1$: It denotes the case when T_j is present but the above threshold is not satisfied. In this case the algorithm will not trace the tag properly and Pr_{Error_1} represents the probability of this false alarm.
- $Error_2$: It denotes a case when T_j is not present but the above threshold is satisfied. In this case the algorithm wrongly alarms that T_j is present while it is not there. The probability of this false alarm is denoted as Pr_{Error_2}.

$Error_1$ only can happen if T_j is not the target tag. Hence, A has N mismatched tags in the range and each tag will reply with probability λ at each query. It can be modeled as a random process with a binomial distribution with parameters $p = \lambda$ and $n = N \cdot Ns$. $Error_1$ happens if:

$$c \geq Ns + (N-1) \cdot Ns \cdot \lambda$$

Hence, this error can be estimated as follows:

$$Pr_{Error_1} =$$

$$\sum_{i=Ns+(N-1)\cdot Ns\cdot\lambda}^{N\cdot Ns} \binom{N \cdot Ns}{i} \times \lambda^i \times (1-\lambda)^{(N\cdot Ns)-i}$$

On the other hand, $Error_2$ can only happen if T_j is there. Hence, A has $N-1$ mismatched tags in the range that will reply to each query with probability λ. It can be modeled as a random process with a binomial distribution with parameters $p = \lambda$ and $n = (N-1) \cdot Ns$. $Error_2$ happens when for Ns queries, the mismatched tags in the range reply less times than $(N-1) \cdot Ns \cdot \lambda$. Hence, Pr_{Error_2} can be estimated as described below:

$$Pr_{Error_2} =$$

$$\sum_{i=0}^{(N-1)\cdot Ns\cdot\lambda-1} \binom{(N-1)\cdot Ns}{i} \times \lambda^i \times (1-\lambda)^{((N-1)\cdot Ns)-i}$$

Assuming large values for N, it can be approximated as follows:

$$Pr_{Error_2} \cong$$

$$\sum_{i=0}^{(N-1)\cdot Ns\cdot\lambda-1} \binom{N\cdot Ns}{i} \times \lambda^i \times (1-\lambda)^{(N\cdot Ns)-i}$$

Given that $Error_1$ and $Error_2$ will never happen together and defining γ as the probability of the event that T_j is present, the total error probability (Pr_{Error}) can be determined as detailed below:

$$Pr_{Error} = Pr_{Error_1} \times (1-\gamma) + Pr_{Error_2} \times \gamma \leq$$

$$Pr_{Error_1} + Pr_{Error_2}$$

On the other hand, for a binomial distribution with parameters $p = \lambda$ and $n = N \cdot Ns$ repetition, we have the following equality:

$$\sum_{i=0}^{N\cdot Ns} \binom{N\cdot Ns}{i} \times \lambda^i \times (1-\lambda)^{(N\cdot Ns)-i} = 1$$

So, the probability of success ($Pr_{Suc.} = 1 - Pr_{Error}$) can be estimated as follows:

$$Pr_{Suc.} \cong$$

$$\sum_{i=(N-1)\cdot Ns\cdot\lambda}^{Ns+(N-1)\cdot Ns\cdot\lambda-1} \binom{N\cdot Ns}{i} \times \lambda^i \times (1-\lambda)^{(N\cdot Ns)-i}$$

In Table 1 the $Pr_{Suc.}$ for several values of λ, Ns and N has been depicted. It shows that the success probability of attack is considerable. The complexity of the described attack is N_s, which represents the number of queries an adversary has to send and then count the number of answers obtained.

Table 1. The success probability of the traceability attack on the third improved search protocol for different values of λ, N and Ns . In this table H and L denote $Ns + (N-1) \cdot Ns \cdot \lambda - 1$ and $(N-1) \cdot Ns \cdot \lambda$ respectively.

λ	Ns	N	H	L	$N \cdot Ns$	$Pr_{suc.}$
0.1	10	10	18	9	100	0.6745
0.1	100	10	189	90	1000	0.8666
0.1	200	10	379	180	2000	0.9385
0.1	10	100	108	99	1000	0.3731
0.1	100	100	1089	990	10000	0.6211
0.1	200	100	2179	1980	20000	0.6847
0.1	10	1000	1008	999	10000	0.1313
0.1	100	1000	10089	9990	100000	0.3708
0.1	200	1000	20179	19980	200000	0.46934
0.05	10	10	13.5	4.5	100	0.5635
0.05	100	10	144	45	1000	0.7853
0.05	200	10	289	90	2000	0.86
0.05	10	100	58.5	49.5	1000	0.4097
0.05	100	100	594	495	10000	0.5971
0.05	200	100	1189	990	20000	0.6317
0.05	10	1000	508.5	499.5	10000	0.1603
0.05	100	1000	5094	4995	100000	0.4454
0.05	200	1000	10189	9990	200000	0.5161
0.01	10	10	9.9	0.9	100	0.634
0.01	100	10	108	9	1000	0.6683
0.01	200	10	217	18	2000	0.7041
0.01	10	100	18.9	9.9	1000	0.5358
0.01	100	100	198	99	10000	0.5536
0.01	200	100	397	198	20000	0.5661
0.01	10	1000	108.9	99.9	10000	0.3183
0.01	100	1000	1098	999	100000	0.5159
0.01	200	1000	2197	1998	200000	0.5209

6 Impersonation Attacks on Search Protocols

Tan *et al.* [20] claim that the search protocols are resistant again cloning attacks. More precisely, they consider the skimming attack described in [11]. The attacker (A) starts querying T_j and capturing a response. She then copies the response on a fake $RFID$ tag ($\hat{T}_j$). A successes in her attempt when she tricks R_i into believing that $\hat{T}_j$ is T_j. The authors state that as result of using fresh random numbers n_r generated by the challenger (R_i), previous responses are not valid and counterfeit tags are detected.

We now show how the first and the third improved search protocols are vulnerable to impersonation attacks, fooling the reader about the presence of the target tag. More precisely, we propose an attack on-the-fly exploiting the symmetric between a query and an answer. That is, in the original protocols the authors ignored the Principle 4 for designing cryptographic protocols [1]. To conduct the attack, A follows the next steps:

1. A eavesdrops the query $X = \{h(f(r_i, t_j) \| n_r) \oplus id_j,\ n_r,\ r_i\}$ sent by R_i and stores this tuple.
2. In the future when A receives a request $X'$from R_i, she can impersonate T_j by replying $Y = \{X[1], X[2]\}$, where X[i] symbolizes the i-th element of X.
3. Finally, when R_i checks Y it is convinced of the presence of T_j.

So, an attacker just replies messages and simulates the presence of the target tag with a 100% of success. In the original protocols, the mistake is two fold: first the random number n_r generated by R_i (the challenger) does not take part in the response and secondly the frame of the query/answer message is symmetric.

7 Conclusions

In [20], the authors dealt with the design of an authentication protocol without requiring a permanent connection to a central database, which is a thought provoking challenge. Furthermore, Tan *et al.* introduced the problem of efficiently searching for an specific tag in a large population of tags. Nevertheless, we show how these proposals are insecure because an attacker can compromise the privacy of confidential information (*id* disclosure attack) and put at risk the privacy of location (traceability attack). Moreover, an attacker can easily trace and supplant a tag, ruining the usefulness of the search protocols. The complexity of the proposed attacks is negligible and the adversary's success probability is significant – sometimes even maximal as in the *id* disclosure or in the impersonation attack.

References

1. Abadi, M., Needham, R.M.: Prudent Engineering Practice for Cryptographic Protocols. IEEE Trans. Software Eng. 22(1), 6–15 (1996)
2. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
3. Phan, R.C.-W.: Cryptanalysis of a New Ultralightweight RFID Authentication Protocol –SASI. IEEE Transactions on Dependable and Secure Computing 6, 316–320 (2009)
4. Damgård, I.B.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
5. Davies, D.W., Price, W.L.: The Application of Digital Signatures Based on Public-Key Cryptosystems. In: Proc. Fifth Intl. Computer Communications Conference, pp. 525–530 (October 1980)
6. Dean, R.D.: Formal Aspects of Mobile Code Security. PhD thesis, Princeton University (1999)
7. Feldhofer, M., Rechberger, C.: A Case Against Currently Used Hash Functions in RFID Protocols. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM Workshops 2006. LNCS, vol. 4277, pp. 372–381. Springer, Heidelberg (2006)
8. FIPS. Secure Hash Standard. National Institute for Standards and Technology, pub-NIST:adr (August 2002)
9. Gauravaram, P., Knudsen, L.R.: On Randomizing Hash Functions to Strengthen the Security of Digital Signatures. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 88–105. Springer, Heidelberg (2009)
10. Halevi, S., Krawczyk, H.: Strengthening Digital Signatures Via Randomized Hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
11. Juels, A.: Strengthening EPC Tags Against Cloning. In: Proc. of WiSe 2005, pp. 67–76. ACM Press (2005)
12. Juels, A., Weis, S.: Defining Strong Privacy for RFID. In: Proc. of PerCom 2007, pp. 342–347. IEEE Computer Society Press (2007)
13. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)

14. Lai, X., Massey, J.L.: Hash Functions Based on Block Ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
15. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
16. National Institute of Standards and Technology. Secure hash standard (SHS). FIPS Publication 180 (May 1993)
17. Preneel, B.: Analysis and Design of Cryptographic Hash Functions. Thesis (Ph.D.), Katholieke Universiteit Leuven, Leuven, Belgium (January 1993)
18. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
19. Rivest, R.L.: RFC 1321: The MD5 Message-Digest Algorithm. Internet Activities Board (April 1992)
20. Tan, C.C., Sheng, B., Li, Q.: Secure and Serverless RFID Authentication and Search Protocols. IEEE Transactions on Wireless Communications 7(4), 1400–1407 (2008)

Appendix

A Notation

Through the paper, we use the following notation:

- R_i: RFID reader i.
- r_i: Static identifier of R_i.
- T_i: RFID tag i.
- id_j: Static identifier of T_i.
- t_i: Secret of T_i.
- n_r: Random number generated by the reader.
- n_t: Random number generated by the tag.
- L_i: Access list for R_i.
- n: Number of entries in L_i.
- $h(x)$: One-way hash function.
- $f(x, y)$: Concatenation of x and y, then applying $h(.)$, $h(x||y)$.
- CA: Trusted party, responsible for authenticating readers and deploying tags.
- m: Number of bits defined by CA, $m < l$.
- l: Output length of hash $h(.)$.
- $A \rightarrow B$: Sending a message from A to B.

B Tan *et al.* Protocols

$$
\begin{aligned}
R_i \rightarrow T_j &: request && (1)\\
R_i \leftarrow T_j &: n_j && (2)\\
R_i \rightarrow T_j &: n_i, r_i && (3)\\
R_i \leftarrow T_j &: h(f(r_i,t_j))_m, h(f(r_i,t_j)||n_i||n_j) \oplus id_j && (4)\\
R_i &: \text{Hash every entry in } L_i \text{ and check} \\
&\quad \text{if first } m \text{ bits match } h(f(r_i,t_j))_m && (5)\\
R_i &: \text{Checks } L_i \text{ for matching } h(f(r_i,t_j))_m && (6)\\
R_i &: \text{Determine } h(f(r_i,t_j)||n_i||n_j), \text{ obtain } id_j && (7)
\end{aligned}
$$

Fig. 1. The mutual authentication protocol proposed by Tan *et al.* [20]

$$
\begin{aligned}
R_i \rightarrow T* &: h(f(r_i,t_j)||n_r) \oplus id_j, n_r, r_i && (1)\\
T* &: \text{Derive } h(f(r_i,t)||n_r) \text{ and XOR with} \\
&\quad h(f(r_i,t_j)||n_r) \oplus id_j && (2)\\
&: \text{If } id = id_j && (3)\\
R_i \leftarrow T_j &: \quad h(f(r_i,t_j)||n_t||n_r) \oplus id_j, n_t && (4)
\end{aligned}
$$

Fig. 2. The basic search protocol proposed by Tan *et al.* [20]

$$
\begin{aligned}
R_i \rightarrow T* &: h(f(r_i,t_j)||n_r) \oplus id_j, n_r, r_i && (1)\\
T* &: \text{Deriving } h(f(r_i,t)||n_r) \text{ and XOR with} \\
&\quad h(f(r_i,t_j)||n_r) \oplus id_j && (2)\\
&: \text{If } id = id_j \text{ and } n_r \neq oldn, \\
&\quad \text{update } oldn = n_r && (3)\\
R_i \leftarrow T_j &: \quad h(f(r_i,t_j)||n_t) \oplus id_j, n_t && (4)
\end{aligned}
$$

Fig. 3. The first improved search protocol proposed by Tan *et al.* [20]

$$
\begin{aligned}
R_i \rightarrow T* &: \text{Broadcast } [id_j]_m, r_i, n_r && (1)\\
T* &: \text{If } id_m = [id_j]_m && (2)\\
R_i \leftarrow T_j &: \quad h(f(r_i,t_j)||n_r||n_t) \oplus id_j, n_t && (3)\\
R_i &: \text{Determines } f(r_i,t_j) \text{ from } L, \text{ obtain } id_j && (4)
\end{aligned}
$$

Fig. 4. The second improved search protocol proposed by Tan *et al.* [20]

$$
\begin{aligned}
R_i \rightarrow T* &: \text{Broadcast } h(f(r_i,t_j)||n_r) \oplus id_j, n_r, r_i && (1)\\
T* &: \text{Derive } h(f(r_i,t)||n_r) \text{ and XOR with} \\
&\quad h(f(r_i,t_j)||n_r) \oplus id_j && (2)\\
&: \text{If } id = id_j : \\
&\quad R_i \leftarrow T_j : h(f(r_i,t_j)||n_t) \oplus id_j, n_t && (3)\\
&: \text{Else :} \\
&\quad R_i \leftarrow T_j : (rand, n_t) \text{ with prob. } \lambda && (4)
\end{aligned}
$$

Fig. 5. The third improved search protocol proposed by Tan *et al.* [20]

C Privacy Model

In RFID schemes, tags (T) and readers (R) interact in protocol sessions. In general terms, the adversary (A) controls the communications between all the

participants and interacts passively or actively with them. Specifically, A can run the following queries:

- Execute(R_i, T_j, k) query. This models a passive attacker. A eavesdrops on the channel, and gets read access to the exchanged messages between R_i and T_j in session k of a genuine protocol execution.
- Test(k', T_0, T_1) query. This does not model any ability of A, but it is necessary to define the untraceability test. When this query is invoked for session k', a random bit is generated $b \in \{0,1\}$. Then, the tokens $\{X_b, Y_b, Z_b, \ldots\}$ from the set $\{\{X_0, Y_0, Z_0, \ldots\}, \{X_1, Y_1, Z_1, \ldots\}\}$ corresponding to tags $\{T_0, T_1\}$ are given to $\mathcal{A}$.

Upon definition of the adversary's abilities, the untraceability problem can be defined as a game G divided into three phases:

- Phase-1 (Learning): A can make any number of Execute queries, which facilitates the eavesdropping of exchanged messages – modelling a passive attacker – over the insecure radio channel.
- Phase-2 (Challenge): A chooses two fresh tags $\{T_0, T_1\}$ whose associated identifiers and keys are $\{id_0, t_0\}$ and $\{id_1, t_1\}$, respectively. He then sends a Test(k', T_0, T_1) query. As a result, and depending on a chosen random bit $b \in \{0,1\}$, A is given the tokens $\{X_b, Y_b, Z_b, \ldots\}$ from the set $\{\{X_0, Y_0, Z_0, \ldots\}, \{X_1, Y_1, Z_1, \ldots\}\}$.
- Phase-3 (Guessing): A ends the game and outputs a bit $\widetilde{b}$ as its conjecture of the value of b.

A's success in winning G is equivalent to the success of breaking the untraceability property offered by the protocol. So the advantage of A in distinguishing whether the messages correspond to T_0 or T_1 is defined as below:

$$Adv_A^{\mathsf{UNT}}(q, kr) = |Pr[\widetilde{b} = b] - \frac{1}{2}|$$

where q is a security parameter (i.e. the bit length of the key shared between the tag and the reader) and kr is the number of times $\mathcal{A}$ runs an Execute query.

Definition 1. *An RFID protocol in an RFID system (S= {R_i, T_0, T_1, ...}) in which an adversary A can invoke* {Execute*(R_i, T_j, k),* Test*(k', T_0, T_1)*} *in a game G, offers resistance against traceability if:*

$$Adv_A^{\mathsf{UNT}}(q, kr) < \varepsilon(q, kr) \tag{3}$$

$\varepsilon(.)$ *being some negligible function.*

D Hash Function Model

A cryptographic hash function maps messages of arbitrary length to fixed-length message digests (hash values). Commonly, to compress messages of arbitrary

length to fixed-length digests, a fixed-input-length compression function is used in a mode of operation. The most commonly used mode of operation is the Merkle-Damgård hash construction [4,15].

The Merkle-Damgård (MD) is a well known hash construction, which is used in almost all popular hash functions such as MD5 [19], SHA-1 [16] and SHA-2 [8]. Given a message $M = M_1 \| ... \| M_l$ and $g : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ as a compression one-way function, MD hash function computes the hash value of M as follows:

$$H^g(M) = g(g(...g(IV, M_1), ...), M_l)$$

where IV is the initial value.

In this paper we assume that the hash functions used in the mutual authentication protocol and the search protocols are based on Merkle-Damgå construction with *fixed IV* but without employing *MD strengthening*. It should be noted that the MD construction has a security reduction [4,15], showing how a collision for the hash function entails a collision for the compression function. This is achieved by including the length of the message as part of the message padding. This technique is called *MD strengthening* [14]. The security reduction of the MD construction is valid for arbitrary initial values (IVs). On the other hand, Damgård [4] also observed that a similar reduction is possible in an iterated hash function construction when the IV is fixed but the message length is not appended. In [17], Preneel recommended to fix the IV as well as to employ MD strengthening, and we can find this approach in many standard hash functions (e.g., SHA and RIPEMD families). Nevertheless, the security of the hash function is not compromise whether the IV is fixed and MD strengthening is not used. This second approach is what we do in our proposed attack. Moreover, this approach sounds more realistic for an RFID application as we can avoid extra calls to the compression function.

Compression one-way functions are the cryptographic primitives that hashes fixed-length-input messages to the fixed-length hash values. These primitives are generally used as the building blocks in a mode of operation (e.g. MD) to design hash functions that process messages of arbitrary length. A common approach to build one-way compression functions is to use block ciphers. Preneel, Govaerts and Vandewalle (PGV) [18] showed sixty-four ways of building compression function modes from block ciphers and twelve of these were shown to be both collision and (second) preimage resistant. Henceforth, these schemes are known as PGV compression functions. Some years later, Black, Rogaway and Shrimpton [2] formally showed that these twelve compression functions are collision and (second) preimage resistant in the ideal cipher model. These compression functions are represented in Table 2, where E symbolizes an ideal block cipher. A general form of these compression functions are as described below:

$$g(H_i, M_i) = E_{K_E}(PT_E) \oplus U$$

where the plaintext PT_E, the key K_E and the feed-forward value U belong to the set $\{M_i, H_i, M_i \oplus H_i, v\}$ and v is a known constant value.

Table 2. PGV compression functions. The column with $\imath$ represents the PGV schemes that are collision and (second) preimage resistant [2]. The column with $\jmath$ represents the numbering of the compression functions as in [18], which was also put forward in [2]. In this Table, $w_i = m_i \oplus H_{i-1}$ and v is a constant value.

$\imath$ [2]	$\jmath$ [18]	PGV: $h_i =$	$\imath$ [2]	$\jmath$ [18]	PGV: $h_i =$
	1	$E_{m_i}(m_i) \oplus v$		33	$E_{m_i}(w_i) \oplus v$
	2	$E_{H_{i-1}}(m_i) \oplus v$		34	$E_{H_{i-1}}(w_i) \oplus v$
	3	$E_{w_i}(m_i) \oplus v$		35	$E_{w_i}(w_i) \oplus v$
	4	$E_v(m_i) \oplus v$		36	$E_v(w_i) \oplus v$
	5	$E_{m_i}(m_i) \oplus m_i$		37	$E_{m_i}(w_i) \oplus m_i$
1	6	$E_{H_{i-1}}(m_i) \oplus m_i$	4	38	$E_{h_i}(w_i) \oplus m_i$
9	7	$E_{w_i}(m_i) \oplus m_i$		39	$E_{w_i}(w_i) \oplus m_i$
	8	$E_v(m_i) \oplus m_i$		40	$E_v(w_i) \oplus m_i$
	9	$E_{m_i}(m_i) \oplus H_{i-1}$	8	41	$E_{m_i}(w_i) \oplus H_{i-1}$
	10	$E_{H_{i-1}}(m_i) \oplus H_{i-1}$		42	$E_{H_{i-1}}(w_i) \oplus H_{i-1}$
11	11	$E_{w_i}(m_i) \oplus H_{i-1}$		43	$E_{w_i}(w_i) \oplus H_{i-1}$
	12	$E_v(m_i) \oplus H_{i-1}$		44	$E_v(w_i) \oplus H_{i-1}$
	13	$E_{m_i}(m_i) \oplus w_i$	6	45	$E_{m_i}(w_i) \oplus w_i$
3	14	$E_{H_{i-1}}(m_i) \oplus w_i$	2	46	$E_{H_{i-1}}(w_i) \oplus w_i$
	15	$E_{w_i}(m_i) \oplus w_i$		47	$E_{w_i}(w_i) \oplus w_i$
	16	$E_v(m_i) \oplus w_i$		48	$E_v(w_i) \oplus w_i$
	17	$E_{m_i}(H_{i-1}) \oplus v$		49	$E_{m_i}(v) \oplus v$
	18	$E_{H_{i-1}}(H_{i-1}) \oplus v$		50	$E_{H_{i-1}}(v) \oplus v$
	19	$E_{w_i}(H_{i-1}) \oplus v$		51	$E_{w_i}(v) \oplus v$
	20	$E_v(H_{i-1}) \oplus v$		52	$E_v(v) \oplus v$
	21	$E_{m_i}(H_{i-1}) \oplus m_i$		53	$E_{m_i}(v) \oplus m_i$
	22	$E_{H_{i-1}}(H_{i-1}) \oplus m_i$		54	$E_{H_{i-1}}(v) \oplus m_i$
12	23	$E_{w_i}(H_{i-1}) \oplus m_i$		55	$E_{w_i}(v) \oplus m_i$
	24	$E_v(H_{i-1}) \oplus m_i$		56	$E_v(v) \oplus m_i$
5	25	$E_{m_i}(H_{i-1}) \oplus H_{i-1}$		57	$E_{m_i}(v) \oplus H_{i-1}$
	26	$E_{H_{i-1}}(H_{i-1}) \oplus H_{i-1}$		58	$E_{H_{i-1}}(v) \oplus H_{i-1}$
10	27	$E_{w_i}(H_{i-1}) \oplus H_{i-1}$		59	$E_{w_i}(v) \oplus H_{i-1}$
	28	$E_v(H_{i-1}) \oplus H_{i-1}$		60	$E_v(v) \oplus H_{i-1}$
7	29	$E_{m_i}(H_{i-1}) \oplus w_i$		61	$E_{m_i}(v) \oplus w_i$
	30	$E_{H_{i-1}}(H_{i-1}) \oplus w_i$		62	$E_{H_{i-1}}(v) \oplus w_i$
	31	$E_{w_i}(H_{i-1}) \oplus w_i$		63	$E_{w_i}(v) \oplus w_i$
	32	$E_v(H_{i-1}) \oplus w_i$		64	$E_v(v) \oplus w_i$

In this article, we consider a group of PGV compression functions that use M_i as the K_E. For instance, one of these compression functions is known as Davies Mayer (DM) [5], which is indexed by Preneel *et al.* [18] as the 25^{th} scheme and by Black *et al.* [2] as the 5^{th} scheme. Given a block cipher E, this compression function accepts the i^{th} chaining value H_i and a message block M_i and compresses these as follows:

$$g(H_i, M_i) = E_{M_i}(H_i) \oplus H_i$$

Other secure schemes that we consider in this article are the schemes 6^{th}, 7^{th} and 8^{th} – following the Black *et al.* [2] indexing.