Strong Authentication and Strong Integrity (SASI) is not that Strong

Gildas Avoine, Xavier Carpent, and Benjamin Martin

Université catholique de Louvain
Information Security Group
B-1348 Louvain-La-Neuve, Belgium

Abstract. In this work, we present a practical passive attack on SASI, an ultra-lightweight mutual authentication protocol for RFID. This attack can be used to reveal with overwhelming probability the secret ID of the prover by eavesdropping about 2^{17} authentications. The result dismantles SASI and, more generally, provides a new approach that threatens ultra-lightweight authentication protocols.

Key words: RFID, authentication, lightweight cryptography, privacy, passive attack

1 Introduction

The recent ubiquitous deployment of RFID systems raised many concerns about privacy. There is a growing need of lightweight authentication protocols to be implemented on low-cost tags that ensure privacy protection. Some existing solutions involve expensive building blocks, such as hash functions and pseudorandom number generators and do not scale well [12, 19]. More recent proposals focus on extremely lightweight protocols that rely on bitwise operations, additions, or bit rotations. The UMAP family of protocols, by Peris-Lopez, Hernandez-Castro, Estevez-Tapiador and Ribagorda [13, 14, 16], paved the way to this new trend but suffers from teething problems [1–3, 6, 9–11]. In [5], Chien proposed another very lightweight authentication protocol providing Strong Authentication and Strong Integrity, so-called SASI. Security analyses have later highlighted weaknesses in its design, and various attacks have been published. In [7] and [18], the authors present active desynchronization and full-disclosure attacks. In [4], the authors proposed a traceability attack on a compromised tag by linking it with past actions performed on this tag. In [17], the author proposed a traceability attack allowing a passive attacker to guess the least significant bit on the static identifier, roughly one out of four times. Finally in [8], the authors proposed a passive full-disclosure attack against a variant of SASI when the rotation is defined as *modular*, whereas in the original paper, the rotation was defined as *Hamming weight*-based. Moreover, the attack presented in [8] is of theoretical interest because it roughly needs a number of observed runs exponential in the number of unveiled bits of the ID.

In the following, we propose a *passive* full-disclosure attack on SASI which is more efficient than the one in [8] and that works with any definition of the rotation. It requires the attacker to eavesdrop 2^{17} (respectively 2^{19}) in the case of the Hamming-weight rotation (respectively modular rotation) in order to almost certainly disclose the full tag ID. Up to our knowledge, this is the first practical passive full-disclosure attack on SASI.

The rest of this paper is organized as follows. In Section 2, we describe the SASI protocol. In Section 3, we introduce some preliminary tools and solve two subproblems required in the attack. In Section 4, we present the attack itself. In Section 5, we present the efficiency analysis of the attack, as well as some optimizations and experimental results. We present our conclusions in Section 6.

2 The SASI Protocol

SASI [5] is a mutual authentication algorithm designed for ultra-lightweight RFID tags. In such tags, randomness must be provided by the reader, because no pseudorandom number generator (PRNG) is provided in the tag, nor is any cryptographic hash function.

Each tag has a secret static identifier ID, and two secret keys K_1 and K_2, as well as a public index-pseudonym IDS. The latter is used by the reader to identify an entry in its internal database, allowing it to retrieve the identifier and the keys related to this tag. Keys and index pseudonyms are updated after each authentication. All quantities involved, including the submessages exchanged are of fixed length $L = 96$ bits.

The SASI protocol relies on logical OR ($\vee$), logical XOR ($\oplus$), modular addition ($+$), and the rotation $Rot(x, y)$. This operation is defined as a circular left-shift of x of $r(y)$ bits, where:

$$r : [0, 2^L - 1] \rightarrow [0, L - 1].$$

Several rotations can be used with SASI, especially the *modular* rotation, that is $r(y) = y \bmod L$, and *Hamming weight* rotation, with $r(y) = \mathcal{H}(y) \bmod L$, where $\mathcal{H}$ is the Hamming weight function. In the latter, the modulus is there to fold the case where $\mathcal{H}(y) = L$ back to a number in $[0, L-1]$ (a rotation of L bits is the same as a rotation of 0 bits).

Note that r was not precisely defined in [5], and it was pointed out in [4] that the rotation intended to be used in the original version of the protocol is the Hamming-weight one. The modular version was introduced in [8].

The protocol definition is as follows. The reader initiates the authentication by sending a *hello* message to the tag, which answers its current index-pseudonym IDS. The reader uses it to find an entry in its internal database with ID, K_1 and K_2. It then produces two nonces n_1 and n_2, and computes A, B, and C as detailed in Figure 1, and sends these three values to the tag.

From A and B, the tag extracts the nonces and uses them to compute C'. If C matches C', the tag authenticates the reader, and then sends D. The reader computes D' and if it matches the received D, it authenticates the tag.

Finally, each party updates the keys and the index pseudonym for further authentications. An overview of the protocol along with messages and update definitions can be found in Figure 1.

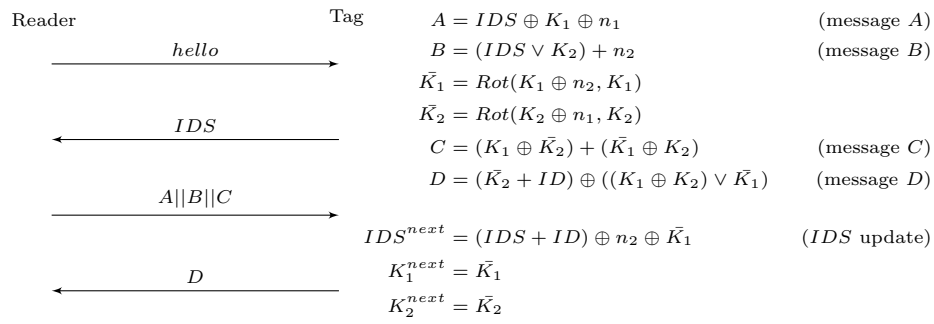

Fig. 1. The SASI protocol

3 Preliminary Tools

In this section we analyze in detail the mechanics of the addition, in order to have a better understanding of equations mixing additions and bitwise operations like logical OR and XOR. We then present two subproblems that are useful in the attack of SASI, but that might be used for other purposes as well.

3.1 Notations and Definitions

In the following, we denote by $[x]_i$ the bit at position i in x. In particular, $[x]_0$ is the least significant bit (LSB) of x, and $[x]_{L-1}$ its most significant bit (MSB). By convention, we say that $[x]_i = 0$ when $i > \lceil \log_2(x) \rceil$. When evoking the i-th bit of x, we refer to $[x]_i$.

We also introduce the following notation for the *carry* of the addition, and the *borrow* of the subtraction, respectively:

$\mathcal{C}(a, b, i)$ denotes the carry at bit i of the sum of a and b, and
$\mathcal{B}(a, b, i)$ denotes the borrow at bit i of the difference of a and b.

Using this notation, we write that the result of the addition of numbers a and b at bit i is:

$$[a + b]_i = [a]_i \oplus [b]_i \oplus \mathcal{C}(a, b, i - 1). \quad (1)$$

Likewise, we write the difference of two numbers a and b at bit i as:

$$[a - b]_i = [a]_i \oplus [b]_i \oplus \mathcal{B}(a, b, i - 1). \quad (2)$$

We compute the *carry* of two numbers a and b at bit index i as:

$$\mathcal{C}(a,b,i) = ([a]_i \wedge [b]_i) \vee \big[([a]_i \vee [b]_i) \wedge \mathcal{C}(a,b,i-1)\big], \tag{3}$$

with the convention that $\mathcal{C}(x,y,i) = 0$ if $i < 0$, for any x and y. Indeed, there is a carry at bit i if either both operands' bits are 1, or if at least one of them is 1 while there is a carry at bit $i-1$. This is no different of the way a computer performs addition.

Similarly, $\mathcal{B}(a,b,i)$ is the *borrow* of the subtraction of b to a at bit i, and is computed as:

$$\mathcal{B}(a,b,i) = ([\overline{a}]_i \wedge [b]_i) \vee \big[([\overline{a}]_i \oplus [b]_i) \wedge \mathcal{B}(a,b,i-1)\big], \tag{4}$$

with the same convention, that is, $\mathcal{B}(x,y,i) = 0$ if $i < 0$, for any x and y. Again, there is a borrow at bit i if either a is 0 and b is 1, or if they are equal but there is a borrow at bit $i-1$.

3.2 Modular Addition

If the + operator is defined as *modular* addition, as often in cryptography, extra care is needed. Namely, when $a \equiv b \pmod N$, that does not necessarily mean that $[a]_i = [b]_i$. Indeed, even though $4 \equiv 1 \pmod 3$, $[4]_0 = 0 \neq [1]_0 = 1$. Recall that $[4]_0$ denotes the bit at index 0 in its base two representation (its LSB). What is true, however, is that if $a \equiv b \pmod N$, then $a \bmod N = b \bmod N$, hence $[a \bmod N]_i = [b \bmod N]_i \; \forall\, i \geq 0$.

In the particular case of $N = 2^L$, then if $a \equiv b \pmod N$, we still have $[a]_i = [b]_i$ if $i < L$. Indeed, when N is a power of 2, computing the remainder is like dropping all the bits above L.

In the practical problems discussed below, we use addition modulo 2^L, and bit indices are smaller than L, so we will not refer to this issue later on.

3.3 First Subproblem

This first subproblem is stated as follows.

Problem 1. Given a and $[a+x]_i$, guess $[x]_i$.

To solve this problem, we use Equation (1), which yields:

$$[x]_i = [a]_i \oplus [a+x]_i \oplus \mathcal{C}(a,x,i-1).$$

We know both $[a]_i$ and $[a+x]_i$, but the carry is unknown. Using Equation (3), we obtain the two following cases. When $[a]_k = 1$:

$$\begin{aligned}\mathcal{C}(a,x,k) &= ([a]_k \wedge [x]_k) \vee \big[([a]_k \oplus [x]_k) \wedge \mathcal{C}(a,x,k-1)\big] \\ &= [x]_k \vee \big([\overline{x}]_k \wedge \mathcal{C}(a,x,k-1)\big) \\ &= [x]_k \vee \mathcal{C}(a,x,k-1).\end{aligned}$$

Likewise, when $[a]_k = 0$:

$$\begin{aligned}\mathcal{C}(a, x, k) &= ([a]_k \wedge [x]_k) \vee \left[([a]_k \oplus [x]_k) \wedge \mathcal{C}(a, x, k-1)\right] \\ &= [x]_k \wedge \mathcal{C}(a, x, k-1).\end{aligned}$$

If we know the distribution of x (usually uniform), we are able compute the probability of $[x]_k$ being 0 or 1, and thus the probability of $\mathcal{C}(a, x, i-1)$ being 0 or 1.

In the general case, we are not sure of the actual values taken by $[x]_i$, although we do have some information as we have seen. That information can be used to guess a possible value for it, which will be correct with a given computable probability. Assuming uniform distribution for x, the possible outputs and their probability of occurring are depicted in Table 1.

Table 1. Possible outputs of $\mathcal{C}(a, x, k)$ and their probability, given a.

$[a]_k$	$\mathcal{C}(a, x, k)$	$\Pr(\mathcal{C}(a, x, k) = 1)$
0	$[x]_k \wedge \mathcal{C}(a, x, k-1)$	$\frac{1}{2} \cdot \Pr(\mathcal{C}(a, x, k-1) = 1)$
1	$[x]_k \vee \mathcal{C}(a, x, k-1)$	$1 - \frac{1}{2} \cdot \Pr(\mathcal{C}(a, x, k-1) = 0)$

We output $[x]_i = [a]_i \oplus [a+x]_i$ if we have probability of carry $\Pr(\mathcal{C}(a, x, i-1) = 1) < \frac{1}{2}$, and $[x]_i = [a]_i \oplus [a+x]_i \oplus 1$ otherwise. The probability of guessing right is computable given the distribution of x.

3.4 Second Subproblem

This second subproblem is stated as follows.

Problem 2. Given a, b, and the relation $a = (b \vee u) + x$, find $[x]_i$ for a given i (u is unknown).

We now see how to get as much information on x as possible. From Equation (2), we have:

$$[x]_i = [a - (b \vee u)]_i = [a]_i \oplus [b \vee u]_i \oplus \mathcal{B}(a, b \vee u, i-1),$$

in which we know $[a]_i$. We also know that, if $[b]_i = 1$, then $[b \vee u]_i = 1$, and if not, we have a 50% chance of guessing the right bit (assuming uniform distribution for u). As for the borrow $\mathcal{B}(a, b \vee u, i-1)$, we can use Equation (4) in the same fashion as we did for the previous subproblem. If $[b]_k = 1$, we have:

$$\begin{aligned}\mathcal{B}(a, b \vee u, k) &= ([\overline{a}]_k \wedge ([b]_k \vee [u]_k)) \vee \left[([\overline{a}]_k \oplus ([b]_k \vee [u]_k)) \wedge \mathcal{B}(a, b \vee u, k-1)\right] \\ &= [\overline{a}]_k \vee \left([a]_k \wedge \mathcal{B}(a, b \vee u, k-1)\right) \\ &= [\overline{a}]_k \vee \mathcal{B}(a, b \vee u, k-1).\end{aligned}$$

However, if $[b]_k = 0$,

$$\begin{aligned}\mathcal{B}(a, b \vee u, k) &= ([\overline{a}]_k \wedge ([b]_k \vee [u]_k)) \vee \big[([\overline{a}]_k \oplus ([b]_k \vee [u]_k)) \wedge \mathcal{B}(a, b \vee u, k-1)\big] \\ &= ([\overline{a}]_k \wedge [u]_k) \vee \big[([\overline{a}]_k \oplus [u]_k) \wedge \mathcal{B}(a, b \vee u, k-1)\big].\end{aligned}$$

Again, we are not always sure of the actual values taken by $[x]_i$, although we can make a good guess with computable probability. Assuming uniform distribution for u, the possible outputs and their probability of occurring are depicted in Table 2.

Table 2. Possible outputs of $\mathcal{B}(a, b \vee u, k)$ and their probability, given a and b.

$[a]_k$	$[b]_k$	$\mathcal{B}(a, b \vee u, k)$	$\Pr(\mathcal{B}(a, b \vee u, k) = 1)$
0	0	$[u]_k \vee \mathcal{B}(a, b \vee u, k-1)$	$1 - \frac{1}{2} \cdot \Pr(\mathcal{B}(a, b \vee u, k-1) = 0)$
0	1	1	1
1	0	$[u]_k \wedge \mathcal{B}(a, b \vee u, k-1)$	$\frac{1}{2} \cdot \Pr(\mathcal{B}(a, b \vee u, k-1) = 1)$
1	1	$\mathcal{B}(a, b \vee u, k-1)$	$\Pr(\mathcal{B}(a, b \vee u, k-1) = 1)$

4 Full-disclosure Attack

4.1 Attack Outline

In this attack scenario, we consider a passive adversary who can only eavesdrop the communications between a reader and a tag, i.e. the submessages A, B, C and D, and IDS. We also assume that the channel between the reader and its database is secure.

The attack is a full-disclosure of the tag's secret ID. It is *probabilistic*, in the sense that the adversary is never 100% sure of the ID recovered. However she can be as close as she wants to this certainty, as long as she has more protocol runs to listen to. It is not dependent of the definition of the rotation, though its efficiency is.

The idea is to build a progressive knowledge on the ID with the information we compute from public quantities. Each information gain requires 3 consecutive successful authentications, but other successful authentications can exist between each information gain.

In order to carry out the attack, we first need to compute the least significant bit (LSB) of the ID, as described in Section 4.2. Once the LSB is retrieved, the attack described in Section 4.3 reveals the remaining bits of ID.

4.2 Attack Initialization

The aim here is to recover the LSB of ID. Therefore, we focus on the case $i = 0$, where the modular addition ($+$) and bitwise XOR ($\oplus$) are the same LSB-wise. Recall that we denoted by $[x]_i$ the i-th bit of x.

Lemma 1. *If* $[IDS]_0 = 1$ *and* $[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = 1$*, we have*

$$[ID]_0 = [B]_0 \oplus [IDS^{next}]_0 \oplus 1.$$

Proof. Let us first look at the submessage (message B) at the LSB:

$$[B]_0 = ([IDS]_0 \vee [K_2]_0) \oplus [n_2]_0.$$

Since $[IDS]_0 = 1$, we have $[n_2]_0 = [B]_0 \oplus 1$, no matter $[K_2]_0$. Moreover, from message definitions (message B), (message C), (message D), and (IDS update), we get the following equalities at the LSB:

$$\begin{aligned}
[B]_0 &= 1 \oplus [n_2]_0, \\
[C]_0 &= [K_1]_0 \oplus [K_2]_0 \oplus [\bar{K_1}]_0 \oplus [\bar{K_2}]_0, \qquad (5)\\
[D]_0 &= [\bar{K_2}]_0 \oplus [ID]_0 \oplus (([K_1]_0 \oplus [K_2]_0) \vee [\bar{K_1}]_0), \qquad (6)\\
[IDS^{next}]_0 &= [IDS]_0 \oplus [ID]_0 \oplus [n_2]_0 \oplus [\bar{K_1}]_0.
\end{aligned}$$

Hence, we have:

$$[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = [K_1]_0 \oplus [K_2]_0 \oplus (([K_1]_0 \oplus [K_2]_0) \vee [\bar{K_1}]_0).$$

Since $[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = 1$, it is impossible that $[\bar{K_1}]_0 = 0$. We thus have:

$$\begin{aligned}
[\bar{K_1}]_0 &= 1, \\
[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 &= [K_1]_0 \oplus [K_2]_0 \oplus 1.
\end{aligned}$$

Now that we know those two quantities, we can get $[\bar{K_2}]_0$ using (5):

$$[\bar{K_2}]_0 = [B]_0 \oplus [D]_0 \oplus [IDS^{next}]_0, \qquad (7)$$

that we then use in (6):

$$\begin{aligned}
[D]_0 &= [\bar{K_2}]_0 \oplus [ID]_0 \oplus (([K_1]_0 \oplus [K_2]_0) \vee [\bar{K_1}]_0), \\
&= [B]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 \oplus [ID]_0 \oplus 1,
\end{aligned}$$

which allows us to conclude.

□

We say that a quantity has a uniform distribution if every element of its domain is equally likely to be instantiated. It is quite easy to see that public quantities IDS, A, B, C, and D have uniform distribution, since their computation involve either a bitwise XOR, or a modular addition with a nonce or with a key (keys also have uniform distributions because they are also updated using a bitwise XOR with a nonce). The domain of these quantities is $[0, 2^L - 1]$, and hence we also have "bitwise" uniform distribution in the sense that every bit has an equal probability to be a zero or a one.

We have seen that getting the LSB of the ID requires two bits to be equal to 1. Since quantities taken into account for this observation have bitwise uniform distribution, the probability of occurrence is $\frac{1}{4}$. The number of runs needed for this observation has a geometric distribution of average 4. The result is quite similar as the one in [17], except here the adversary knows for sure when conditions are met, because they only involve public quantities.

In fact, we can generalize the attack to an arbitrary bit index, but this would need conditions of which probabilities of occurrence are negative exponential in the position of that bit index. Instead, we describe in Section 4.3 a much more efficient attack to retrieve $[ID]_i$ when $i > 0$.

4.3 Attack Details

Now that we have the required tools, we describe the core of the attack more thoroughly. We have seen in Section 4.2 that the adversary can easily recover the least significant bit of the ID of a tag, so we assume that this part of the attack has already been executed, and that we know $[ID]_0$.

At the LSB, (IDS update) becomes:

$$[IDS^{(n+1)}]_0 = [IDS^{(n)}]_0 \oplus [ID]_0 \oplus [n_2^{(n)}]_0 \oplus [\bar{K_1}^{(n)}]_0. \tag{8}$$

We have also seen in Section 4.2 that when $[IDS^{(n)}]_0 = 1$, $[n_2^{(n)}]_0$ is known by computing $[B^{(n)}]_0 \oplus [IDS^{(n)}]_0$. Thus, from Equation (8), we get:

$$[\bar{K_1}^{(n)}]_0 = [ID]_0 \oplus [B^{(n)}]_0 \oplus [IDS^{(n+1)}]_0.$$

The next round, according to the key updating process $K_1^{(n+1)} = \bar{K_1}^{(n)}$, we know $[K_1^{(n+1)}]_0$. Furthermore, if again $[IDS^{(n+1)}]_0 = 1$, we have $[n_2^{(n+1)}]_0 = [B^{(n+1)}]_0 \oplus [IDS^{(n+1)}]_0$. Thus, we know $[K_1^{(n+1)} \oplus n_2^{(n+1)}]_0$. So, since $\bar{K_1}^{(n+1)} = Rot(K_1^{(n+1)} \oplus n_2^{(n+1)}, K_1^{(n+1)})$, we have:

$$[K_1^{(n+1)} \oplus n_2^{(n+1)}]_0 = [\bar{K_1}^{(n+1)}]_{r(K_1^{(n+1)})}.$$

Recall from Section 2 that r denotes the function used in the rotation operation. In most cases, we do not know $r(K_1^{(n+1)})$, since we only know the LSB of $K_1^{(n+1)}$, but we can say that, assuming K_1 has a uniform statistical distribution,

$$\Pr(r(K_1^{(n+1)}) = i) = p_r(i) \qquad \forall\, i \in [0, L-1],$$

where p_r is the probability distribution function of r. For instance, in the case of modular rotation, $r(x) = x \bmod L$, and $p_r(x) = \frac{1}{L}$, and in the case of Hamming weight rotation, $r(x) = \mathcal{H}(x)$, and $p_r(x) = \frac{\binom{L}{x}}{2^L}$. So, with probability $p_r(i)$, we know $[\bar{K_1}^{(n+1)}]_i$. We then use this result in (IDS update):

$$[ID + IDS^{(n+1)}]_i = [IDS^{(n+2)}]_i \oplus [n_2^{(n+1)}]_i \oplus \underbrace{[\bar{K_1}^{(n+1)}]_i}_{[K_1^{(n+1)} \oplus n_2^{(n+1)}]_0}.$$

The computation of $[n_2^{(n+1)}]_i$ has already been discussed in Section 3.4. Finally, we have $[ID + IDS^{(n+1)}]_i$ and $IDS^{(n+1)}$, but this does not necessarily mean that we have $[ID]_i$. However, we can recover some information on $[ID]_i$ using what we know, as seen in Section 3.3. The result is that we obtain $[ID]_i$ with a *certain* probability, that will be quantified in the next section.

An outline of the attack can be seen in Figure 2.

Execute the traceability attack to get $[ID]_0$
repeat
  if $[IDS]_0 = 1$ **then**
    if the previous $[K_1]_0$ was known (that is, previous $[IDS]_0$ was 1) **then**
      Compute n_2 probabilistically (2)
      Compute ID probabilistically given IDS and $IDS + ID$ (1)
      for all $i \in [1, L-1]$ **do**
        Update the knowledge of $[ID]_i$ with the advantages of (1) and (2) and $p_r(i)$
      end for
    end if
    Compute $[K_1^{next}]_0 = [ID]_0 \oplus [B]_0 \oplus [IDS^{next}]_0$
  end if
until all the bits of ID are found with satisfactory probability

---

Fig. 2. Outline of the attack. (1) refers to the first subproblem discussed in Section 3.3, and (2) to the second one, which is discussed in Section 3.4. Note that the "computed" $[K_1^{next}]_0$ is on the attacker side, and that the actual $[K_1^{next}]_0$ is unknown. For clarity reasons, we did not make a difference of notation between computed (or guessed) values and real ones.

In order to provide a more intuitive description of the attack, consider the following game. Having a slightly biased coin, we want to know what side of this coin is biased. We can toss it as many times as we want, but we want to have a good probability of guessing the right side, while minimizing the number of tosses. The smaller the bias is, the harder it is to tell whether it is heads or tails that is the most favorable side. Indeed, if the bias is, for instance 75% for heads and 25% for tails, we already have a good information with 20 tosses. However, if the bias turns out to be 50.01% for heads, and 49.99% for tails, we would need a whole lot more of them.

This is exactly the idea of convergence of the attack, except that we are guessing the biased side of $L-1$ independent coins, and that each toss has a different "weight". Indeed we have seen in the attack that we only get some information when $[IDS]_0 = 1$ for two consecutive runs. Moreover, each information gain has to be weighted with $p_r(i)$, the information we have on n_2, and the one we have on ID given $ID + IDS$. For instance, when $p_r(i)$ is small, the guessed rotation

has a small probability of being right, thus we bring less valuable information than when $p_r(i)$ is big.

5 Efficiency Analysis and Experiments

5.1 Theoretical Analysis

We now analyze the efficiency of the attack, by showing what conclusion we can draw given the set of observations and by linking these observations with the probability of guessing the right ID.

Let us assume that we have at our disposal an oracle that has a secret bit b and a set of $0 \leq q_i \leq 1$ which, upon query, outputs a bit b_i and a value q_i such that $\Pr(b_i = b) = q_i$. We assume that $\Pr(b = 0) = \Pr(b = 1) = \frac{1}{2}$, that all the outputs are independent, and that $q_i \geq \frac{1}{2}$, without loss of generality. Indeed, if one guess is such that $q_i < \frac{1}{2}$, then it would be equivalent to output the opposite bit with complementary probability $1 - q_i > \frac{1}{2}$.

For convenience, let us denote by O the observations, that is the event corresponding to observing the set of bits b_i. Let us also define the sets:

$$S_0 = \{i \in [1, N] \mid b_i = 0\}, \text{ and}$$
$$S_1 = \{i \in [1, N] \mid b_i = 1\}.$$

If we observe $N = |S_0| + |S_1|$ outputs of the oracle, we have:

$$\Pr(O|b = 0) = \prod_{i \in S_0} q_i \prod_{i \in S_1} (1 - q_i), \tag{9}$$

$$\Pr(O|b = 1) = \prod_{i \in S_0} (1 - q_i) \prod_{i \in S_1} q_i. \tag{10}$$

Using Bayes' rule :

$$\begin{aligned}
\Pr(b = 0|O) &= \frac{\Pr(b = 0 \cap O)}{\Pr(O)} \\
&= \frac{\Pr(O|b = 0) \cdot \Pr(b = 0)}{\Pr(O|b = 0) \cdot \Pr(b = 0) + \Pr(O|b = 1) \cdot \Pr(b = 1)} \\
&= \frac{\Pr(O|b = 0)}{\Pr(O|b = 0) + \Pr(O|b = 1)},
\end{aligned}$$

since events $b = 0$ and $b = 1$ are equiprobable. Now we use Equations (9) and (10), and obtain:

$$\begin{aligned}
\Pr(b = 0|O) &= \frac{\prod_{i \in S_0} q_i \prod_{i \in S_1} (1 - q_i)}{\prod_{i \in S_0} q_i \prod_{i \in S_1} (1 - q_i) + \prod_{i \in S_0} (1 - q_i) \prod_{i \in S_1} q_i} \\
&= \frac{1}{1 + \prod_{i \in S_0} \frac{1 - q_i}{q_i} \prod_{i \in S_1} \frac{q_i}{1 - q_i}}.
\end{aligned} \tag{11}$$

An equivalent but more convenient way of seeing this is the following. Instead of outputting probabilities q_i, the oracle can output *advantages* a_i such that $|\Pr(b_i = b) - \Pr(b_i \neq b)| = a_i$. Put differently, we have $a_i = |2q_i - 1|$. In this scenario, Equation (11) becomes:

$$\Pr(b = 0|O) = \frac{1}{1 + \prod_{i \in S_0} \frac{1-a_i}{1+a_i} \prod_{i \in S_1} \frac{1+a_i}{1-a_i}}. \tag{12}$$

Recall from Figure 2 that the information on each bit of the ID is weighted using:

- $p_r(i)$
- the level of trust on $[ID]_i$ given $[ID + IDS]_i$ (subproblem 1)
- the level of trust on $[n_2]_i$ (subproblem 2)

Indeed, each guess on the i-th bit of the ID is correct if:

- the guessed rotation is the correct one
- the guess of $[ID]_i$ given $[ID + IDS]_i$ is correct
- the guess at $[n_2]_i$ is correct

The probability of correctness for the rotation is simply $p_r(k)$, and the probability of correctness for the two subproblems (the level of trust or *advantage* on the quantities n_2 and ID) is computable, given public submessages, as seen in Sections 3.3 and 3.4. If we assume independence between these advantages, we just have to multiply them to obtain an advantage a_i related to the k-th bit on the i-th run. Using Equation (12), we can have a total probability on the value of a bit of the ID, given a certain amount of information materialized by the guesses and advantages on these guesses on that bit.

We have observed experimentally that the average advantages for the first and second subproblems are respectively roughly $\frac{1}{2}$ and $\frac{1}{3}$. This is particularly important for the second subproblem, because we see that simply knowing IDS and B, we can guess roughly one third of n_2.

Recall from Section 4.3 that to execute the inner part of the attack and thus bring information, we require $[IDS]_0 = 1$ for two consecutive runs. Since IDS has a uniform distribution, this will occurs with probability $\frac{1}{4}$. We thus need to multiply the number of runs needed by 4.

5.2 Optimizations and Experimentations

We introduce in this section some optimizations that improve in practice the efficiency of the attack presented in Section 4.3.

First of all, we raise that it is somewhat wasteful to only use $[ID]_0$ while more and more knowledge on ID is revealed along the attack. Progressive knowledge on the ID can not only help solving the first subproblem ($[ID]_i$ from IDS and $[ID + IDS]_i$), but it can also be used as a base, instead of $[ID]_0$ only.

This especially helps with the Hamming-weight rotations where the most and least significant bits are hard to guess using $[ID]_0$ only (because $p_r(i)$ is very small for small or big i).

We have carried out experiments with the two definitions of the rotation and have observed the number of *errors* (number of wrong guesses) on average. When this number is close to 0, it means that we manage to correctly recover the whole ID with good probability, and when it is close to $\frac{L}{2} = 48$, it means that the output guessed ID is not better than if we had guessed one at random. Table 3(a) and Table 3(b) contain the results respectively without and with optimizations (using other $[ID]_i$ as bases). Figure 3 shows the evolution of the quality of the ID recovered (when applying the optimization) with respect N, the number of observed runs.

Table 3. Average number of errors (number of wrong guesses on the ID) respectivelely without (a) and with (b) optimizations. These results were obtained on an average of roughly 500 experiments conducted with a simulated SASI initiated with random secrets. Recall that N is the total number of runs observed, and $L = 96$ is the length of the quantities involved in the protocol.

(a) No optimizations

N	$\mathcal{H}(x) \bmod L$	$x \bmod L$
2^{18}	-	16.230
2^{19}	-	8.011
2^{20}	-	2.973
2^{21}	-	1.375
2^{22}	-	0.628
2^{23}	-	0.154

(b) Optimization applied

N	$\mathcal{H}(x) \bmod L$	$x \bmod L$
2^{14}	22.985	38.741
2^{15}	9.944	33.81
2^{16}	2.908	19.45
2^{17}	0.620	5.436
2^{18}	0.159	1.294
2^{19}	0.041	0.436

Note that in Table 3(a), the results for Hamming-weight rotation are not shown since only the 40 or so middle bits are relevant, as explained above. However, when applying optimizations, we see that it becomes possible to solve it, with slightly better results than with modular rotation. We also see that optimizations reduce the average required number of observed runs, as expected.

Note also that using Equation (12), it is possible to have an idea of the trust on the guessed bits. Hence we can know if more runs are needed.

We can also imagine other optimizations such as the following. The adversary does not know K_1 in whole, but she does know $[K_1]_0$, and for instance in the case of modular rotation, she knows the parity of it, and thus she can reduce the possibilities from L down to $\frac{L}{2}$, which improves the advantage per sample quite a bit. She could also reuse old authentication sessions when she has better knowledge of ID. We did not take these into accout in the experiments for Table 3(b). The point is that we could further reduce the number of runs needed to achieve overwhelming probability, but the first optimization alone is enough to make the attack practical.

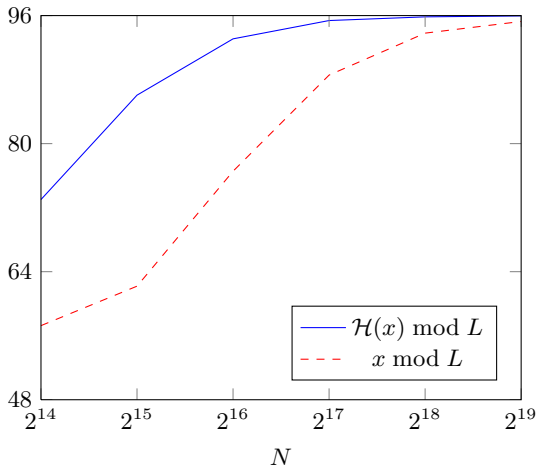

Fig. 3. Average number of bits correctly guessed in ID, for the two usual rotations (optimization applied).

6 Conclusion

In this article we have presented a passive full-disclosure attack on SASI. We have shown that eavesdropping 2^{17} (respectively 2^{19}) in the case of the Hamming-weight rotation (respectively modular rotation) is enough to almost certainly disclose the full tag ID. Up to our knowledge, we provide the first practical full-disclosure attack against SASI with a passive adversary.

We have seen that the submessage B is weak and wraps n_2 quite poorly. We have indeed seen that a passive attacker has an average advantage of roughly one third over each bit of n_2, by simply eavesdropping IDS and B. It shows once more that logical OR ($\vee$) and logical AND ($\wedge$) should by all means be avoided in the external parts of public submessages. We have also seen how to deal with expressions mixing modular addition ($+$) with other bitwise operations, and that they are usually weaker than they appear.

Although rotations are a nice addition to the lightweight family of operations already used in previous similar protocols such as the ones of the UMAP family [16, 14, 13], we have seen that it is by itself not enough to ensure the security of the protocol. The inclusion of a non-triangular operation, however, has made the recovery process much more complex, and a "bottom-up" approach for an attack such as those described in [2, 3] is no longer possible. Consequently, the average number of successful authentications needed to observe is much larger. The Hamming-weight rotation which has been praised in [8] did not prove any stronger than the modular one. The issue with the rotation is that the rotated bits are not changed, and that the set of possible outputs is small (L possibilities).

In conclusion, besides showing several weaknesses in the design of SASI, this attack also introduces a new way of cryptanalysis of ultralightweight authenti-

cation protocols, namely building progressive knowledge on a quantity given a series of observations of non-negligible advantage. Other protocols might suffer from the same weaknesses, and the same approach could be used to analyse their security. Gossamer, by Peris-Lopez et al. [15], was somewhat inspired by SASI and the protocols from the UMAP family, but is more mature and includes features such as double rotations and lightweight PRNG's. Determining whether this protocol is secure and if it can be analaysed using this novel technique remains an open question.

Acknowledgements. This work was partially funded by the Walloon Region Marshall plan through the SPW DG06 Project TRASILUX.

References

1. B. Alomair, L. Lazos, and R. Poovendran. Passive attacks on a class of authentication protocols for RFID. *Information Security and Cryptology-ICISC 2007*, pages 102–115, 2007.
2. M. Bárász, B. Boros, P. Ligeti, K. Lója, and D. Nagy. Breaking LMAP. *Printed handout of Workshop on RFID Security – RFIDSec 07*, July 2007.
3. M. Bárász, B. Boros, P. Ligeti, K. Lója, and D. Nagy. Passive attack against the M2AP mutual authentication protocol for RFID tags. In *Proc. of First International EURASIP Workshop on RFID Technology*, 2007.
4. Tianjie Cao, Elisa Bertino, and Hong Lei. Security Analysis of the SASI Protocol. *IEEE Transactions on Dependable and Secure Computing*, 6:73–77, 2008.
5. Hung-Yu Chien. SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity. *IEEE Transactions on Dependable and Secure Computing*, 4:337 – 340, 2007.
6. H.Y. Chien and C.W. Huang. Security of ultra-lightweight RFID authentication protocols and its improvements. *ACM SIGOPS Operating Systems Review*, 41(4):86, 2007.
7. Paolo D'Arco and Alfredo De Santis. From Weaknesses to Secret Disclosure in a Recent Ultra-Lightweight RFID Authentication Protocol. Cryptology ePrint Archive, Report 2008/470, 2008. `http://eprint.iacr.org/`.
8. J. C. Hernandez-Castro, J. M. E. Tapiador, P. Peris-Lopez, and J.-J. Quisquater. Cryptanalysis of the SASI Ultralightweight RFID Authentication Protocol with Modular Rotations. *ArXiv e-prints*, November 2008.
9. T. Li and R. Deng. Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 238–245, 2007.
10. T. Li and G. Wang. Security analysis of two ultra-lightweight RFID authentication protocols. *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 109–120, 2007.
11. T. Li, G. Wang, and R.H. Deng. Security Analysis on a Family of Ultra-lightweight RFID Authentication Protocols. *Journal of Software*, 3(3):1, 2008.
12. Miyako Ohkubo and Koutarou Suzuki and Shingo Kinoshita. Cryptographic approach to privacy-friendly tags. *RFID Privacy Workshop*, 2003.

13. P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. EMAP: An efficient mutual-authentication protocol for low-cost RFID tags. *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 352–361, 2006.
14. P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. M^2AP: A minimalist mutual-authentication protocol for low-cost RFID tags. *Ubiquitous Intelligence and Computing*, pages 912–923, 2006.
15. P. Peris-Lopez, J. Hernandez-Castro, J. Tapiador, and A. Ribagorda. Advances in Ultralightweight Cryptography for Low-Cost RFID Tags: Gossamer Protocol. *Information Security Applications*, pages 56–68, 2009.
16. P. Peris-Lopez, J.C. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags. In *Proc. of 2nd Workshop on RFID Security*, page 06. Citeseer, 2006.
17. Raphael C.-W. Phan. Cryptanalysis of a New Ultralightweight RFID Authentication Protocol - SASI. *IEEE Transactions on Dependable and Secure Computing*, 6:316–320, 2008.
18. Hung-Min Sun, Wei-Chih Ting, and King-Hang Wang. On the Security of Chien's Ultra-Lightweight RFID Authentication Protocol. *IEEE Transactions on Dependable and Secure Computing*, 99(PrePrints), 2009.
19. T. Van Le, M. Burmester, and B. de Medeiros. Forward-secure RFID authentication and key exchange. *IACR ePrint, February*, 2007.