On the Security of HB# against a Man-in-the-Middle Attack

Khaled Ouafi*, Raphael Overbeck**, and Serge Vaudenay

Ecole Polytechnique Fédérale de Lausanne (EPFL),
CH-1015 Lausanne, Switzerland

Abstract. At EuroCrypt '08, Gilbert, Robshaw and Seurin proposed HB# to improve on HB+ in terms of transmission cost and security against man-in-the-middle attacks. Although the security of HB# is formally proven against a certain class of man-in-the-middle adversaries, it is only conjectured for the general case. In this paper, we present a general man-in-the-middle attack against HB# and RANDOM-HB#, which can also be applied to all anterior HB-like protocols, that recovers the shared secret in 2^{25} or 2^{20} authentication rounds for HB# and 2^{34} or 2^{28} for RANDOM-HB#, depending on the parameter set. We further show that the asymptotic complexity of our attack is polynomial under some conditions on the parameter set which are met on one of those proposed in [8].

Keywords: HB, authentication protocols, RFID.

1 Introduction

Designing secure cryptographic protocols using lightweight components is one of the main challenges of cryptography. Indeed, the emergence of new technology such as radio-frequency identification (RFIDs) with low computation and memory capabilities has stressed the need of such protocols.

These devices require protection from many threats. For example, for a company using RFIDs in inventories and supply-chain management, a RFID tag should be protected from cloning. Biometric passports also have a tight relation with RFIDs since they use contactless chips to communicate and authenticate the passport holder to some authorized authority. Using RFID tags as a replacement of barcodes by many merchant have also raised the issue of traceability and privacy protection. Thus, the need of authentication protocols providing efficiency, security and privacy protection has become a key factor for the future development of this technology. One of the most popular attempts to fulfill this need are the HB family of authentication protocol.

* Supported by a grant of the Swiss National Science Foundation, 200021-119847/1.
** Funded by DFG grant OV 102/1-1.

J. Pieprzyk (Ed.): ASIACRYPT 2008, LNCS 5350, pp. 108–124, 2008.

The HB Family. Originally introduced by Hopper and Blum [11], the HB protocol aims at authenticating RFID tags to a reader using very lightweight operations while reducing its security to a well-known $\mathcal{NP}$-hard problem: the learning parity with noise (LPN) problem [1]. In fact, this protocol only requires a matrix multiplication and some basic XOR operations. But Juels and Weis [12] showed later that HB is insecure against adversaries able to interact with tags by impersonating readers and then proposed a new variant immune against this type of attacks: HB$^+$. As these two protocols were initially studied in a scenario of a sequential executions, Katz and Shin [13] extended both security proofs of HB and HB$^+$ to a more general concurrent and parallel setting. However, as Gilbert, Robshaw and Sibert noted in [6], the security of HB$^+$ is compromised if the adversary is given the ability to modify messages going from the reader to the tag. This model was later known as the GRS security model.

Since then, many HB-like protocols aiming security in the GRS model were proposed. Most notably, we mention the works of Bringer, Chabanne and Dottax on HB^{++} [3], Munilla and Peinado on HB-MP [15] and Duc and Kim on HB* [4]. But all these protocols were proven to be insecure in the GRS model, as all of them were successfully cryptanalyzed by Gilbert, Robshaw and Seurin in [7].

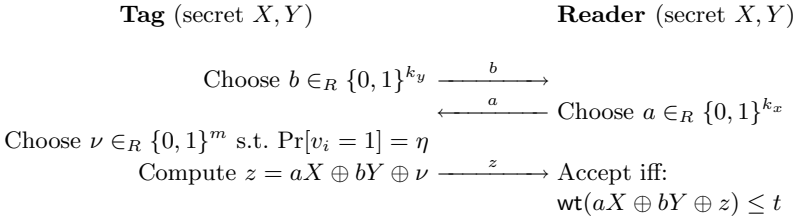

Fig. 1. The RANDOM-HB# and HB# protocols. In RANDOM-HB#, $X \in \mathbb{F}_2^{k_x \times m}$ and $Y \in \mathbb{F}_2^{k_y \times m}$ are random matrices, in HB# they are Toeplitz matrices. wt denotes the Hamming weight.

At EuroCrypt '08, Gilbert, Robshaw and Seurin [8,9], proposed a new variant of HB$^+$ named RANDOM-HB# and its optimized version HB#. In these protocols, the tag and the reader share some secret matrices X and Y. During an authentication instance, both issue challenges of k_y-bit and k_x-bit length respectively and the final response of the tag is a m-bit message disturbed by a noise vector in which every bit has a probability η of being 1.

The details of the RANDOM-HB# and HB# protocols are outlined in Figure 1 and the proposed parameters (inspired from the results of [14]) in Table 1. The difference between these two versions lies in the structure of the secret matrices X and Y: while in RANDOM-HB# these two are completely random, thus needing $(k_x + k_y)m$ bits of storage, HB# reduces this amount to $k_x + k_y + 2m - 2$ by using Toeplitz matrices for X and Y.

Besides generating two random vectors ν and b, the operations performed by the tag to authenticate itself are very cheap: it only needs two matrix

Table 1. HB$^{\#}$ Parameter sets proposed in [8,9]. P_{FR} and P_{FA} denote the false rejection and false acceptance rates respectively. In the set III, the Hamming weight of the error vector ν generated by the tag is smaller than t.

Parameter set	k_x	k_y	m	η	t	P_{FR}	P_{FA}
I	80	512	1164	0.25	405	2^{-45}	2^{-83}
II	80	512	441	0.125	113	2^{-45}	2^{-83}
III	80	512	256	0.125	48	0	2^{-81}

multiplications to compute aX and bY which can be implemented using basic AND and XOR operations along with two bitwise XOR operations between two m-bit vectors. In some variant, the tag generates a random error vector ν until it has weight no larger than t requiring the tag to be able to compute a Hamming weight wt.

RANDOM-HB$^{\#}$ is also accompanied with a proof of security in the GRS security model if the parameters satisfy the condition $m\eta \leq t \leq m/2$. Under the conjecture that the Toeplitz-MHB puzzle is hard, HB$^{\#}$ is also secure in the same model. However, both protocols only provide "strong arguments" in favor of their resistance against man-in-the middle adversaries and formally proving their security in such a model was left as an open problem.

Our Contribution. In this paper, we present an attack against RANDOM-HB$^{\#}$ and HB$^{\#}$ in a general man-in-the-middle attack where the adversary is given the ability to modify all messages. The idea of our attack is to modify the messages of a session according to values obtained from a passive attack where the adversary eavesdrops on a protocol session between a reader and the tag.

Through this paper, we will denote b and z (resp. a) the values sent by the tag (resp. the reader) and $\hat{b}$ and $\hat{z}$ (resp. $\hat{a}$) the value received by the reader (resp. the tag) after corruption by the adversary. Thus the tag computes $z = \hat{a}X \oplus bY \oplus \nu$ while the reader checks that $\mathsf{wt}(aX \oplus \hat{b}Y \oplus \hat{z}) \leq t$.

Outline. Our paper is organized as follows. First, we show how it is possible to mount a man-in-the-middle attack against HB$^{\#}$ by proposing an algorithm able to compute the Hamming weight of the errors introduced by the tag in a session $(\bar{a}, \bar{b}, \bar{z})$. Then, we provide a complexity analysis of this initial attack needed by the man-in-the-middle to fully recover the secret matrices of RANDOM-HB$^{\#}$ and HB$^{\#}$. Afterwards, we present our optimized attack in Section 4 and give the complexity results applied to parameter sets I and II of HB$^{\#}$ of Table 1. After that, we investigate some open proposals to limit the Hamming weight of the error vector in HB-like protocols and present an attack against the parameter set III of HB$^{\#}$ shown in Table 1. At last, we show the lower bounds on the parameters for which our attack does not work.

2 Basic Attack

In this section, we show that, contrarily to what was conjectured in [8,9], both RANDOM-HB# and HB# are vulnerable against man-in-the-middle attacks by presenting a (non-optimized) attack.

2.1 Principle

The core of our attack is Algorithm 1 in which Φ denotes the cumulative distribution function of the normal distribution. It shows how an adversary able to modify messages going in both directions can compute the Hamming weight of the error vector $\bar{\nu} = \bar{a}X \oplus \bar{b}Y \oplus \bar{z}$ denoted $\bar{w} = \mathsf{wt}(\bar{\nu})$ introduced in a triplet $(\bar{a}, \bar{b}, \bar{z})$. The crucial observation is that since $z = \hat{a}X \oplus bY \oplus \nu$, at in each for-loop of Algorithm 1, the reader computes the Hamming weight $\mathsf{wt}(\nu \oplus \bar{\nu})$ of

$$aX \oplus \hat{b}Y \oplus \hat{z} = aX \oplus (\bar{b} \oplus b)Y \oplus (\bar{z} \oplus z) = (\hat{a}X \oplus bY \oplus z) \oplus (\bar{a}X \oplus \bar{b}Y \oplus \bar{z}) = \nu \oplus \bar{\nu}$$

and accepts iff $\mathsf{wt}(\nu \oplus \bar{\nu}) \leq t$.

Algorithm 1. Approximating $\bar{w}$

Input: $\bar{a}, \bar{b}, \bar{z}$, n
Output: $P^{-1}\left(\frac{c}{n}\right)$, an approximation of $\bar{w} = \mathsf{wt}(\bar{a}X \oplus \bar{b}Y \oplus \bar{z})$
where $P(\bar{w}) = \Pr[\mathsf{wt}(\nu \oplus \bar{\nu}) \leq t] = \Phi(\frac{t-(m-\bar{w})\eta-\bar{w}(1-\eta)}{\sqrt{m\eta(1-\eta)}})$
Processing:
1: Initialize $c \leftarrow 0$
2: **for** $i = 1 \ldots n$ **do**
3: During a protocol, set $\hat{a} \leftarrow a \oplus \bar{a}$, $\hat{b} \leftarrow b \oplus \bar{b}$ and $\hat{z} \leftarrow z \oplus \bar{z}$
4: **if** reader accepts **then**
5: $c \leftarrow c + 1$
6: **end if**
7: **end for**

Correctness. We show, that the output of Algorithm 1 is indeed an estimation of $\mathsf{wt}(\nu \oplus \bar{\nu})$. The probability p that a bit of $(\nu \oplus \bar{\nu})$ is 1 is given by:

$$p = \Pr[(\nu \oplus \bar{\nu})_i = 1] = \begin{cases} \eta & \text{if } \bar{\nu}_i = 0 \\ 1 - \eta & \text{if } \bar{\nu}_i = 1. \end{cases}$$

Hence, $m - \bar{w}$ bits of $(\nu \oplus \bar{\nu})$ follow a Bernoulli distribution of parameter η and the other $\bar{w}$ bits follow a Bernoulli distribution of parameter $1 - \eta$, thus $\mathsf{wt}(\nu \oplus \bar{\nu})$ follows a binomial distribution. Because of the independence of all bits, the expected value and variance of $\mathsf{wt}(\nu \oplus \bar{\nu})$ are given by $\mu = (m - \bar{w})\eta + \bar{w}(1 - \eta)$ and $\sigma^2 = m\eta(1 - \eta)$ respectively.

We now define the function P as $P(\bar{w}) = \Pr[\mathsf{wt}(\nu \oplus \bar{\nu}) \leq t]$. By the definition of the standard normal cumulative distribution function Φ and the central limit theorem, we have that

$$P(\bar{w}) \approx \Phi(u), \text{ with } u = \frac{t-\mu}{\sigma} . \tag{1}$$

The random variable $\frac{c}{n}$ thus follows a normal distribution with expected value $P(\bar{w})$ and variance $\frac{1}{n}P(\bar{w})(1-P(\bar{w}))$. To decide whether $\mathsf{wt}(\bar{\nu}) = \bar{w}$ or not, the estimate $\frac{c}{n}$ for $P(\mathsf{wt}(\bar{\nu}))$ has to be good enough. The difference of the probabilities is at least $P(\bar{w}+1) - P(\bar{w}) \approx P'(\bar{w})$ which we can compute as

$$P'(\bar{w}) \approx -\frac{1-2\eta}{\sqrt{m\eta(1-\eta)}}\Phi'(u) = -\frac{1-2\eta}{\sqrt{m\eta(1-\eta)}} \times \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} .$$

By taking

$$n = \frac{\theta^2}{r^2} R(\bar{w}) \quad \text{with} \quad R(\bar{w}) = 2\frac{P(\bar{w})(1-P(\bar{w}))}{(P'(\bar{w}))^2} , \tag{2}$$

the probability that $|\frac{c}{n} - P(\bar{w})| > r|P'(\bar{w})|$ is $2\Phi(-\theta\sqrt{2}) = \mathsf{erfc}(\theta)$. With θ high enough, $\frac{c}{n}$ yields a estimate of $P(\bar{w})$ with precision $\pm rP'(\bar{w})$. Thus, Algorithm 1 is correct if n is chosen large enough.

Choice of Input. To determine a reasonable choice for the input n, we have to fix values for r and θ. If we can assume that $\bar{w} = \mathsf{wt}(\bar{\nu})$ is an integer close to some value w_0, we can call Algorithm 1 and $r = \frac{1}{2}$ to infer $\bar{w} = \lceil P^{-1}(\frac{c}{n}) \rfloor$ with error probability $\mathsf{erfc}(\theta)$ (here, $\lceil \cdot \rfloor$ refers to normal rounding). On the other hand, if we know that $\bar{w} \in \{w_0 - 1, w_0 + 1\}$, we can choose $r = 1$ to infer $\bar{w}$ by the closest value to $P^{-1}(\frac{c}{n})$. The error probability is $\frac{1}{2}\mathsf{erfc}(\theta)$. In both cases, Algorithm 1 is an oracle of complexity $n = \frac{\theta^2}{r^2}R(w_0)$ that can be used to compute $\bar{w}$ given $\bar{a}, \bar{b}, \bar{z}$ and succeeding with an probability of error smaller than $\mathsf{erfc}(\theta)$.

Since we have to recover ℓ secret bits by Algorithm 1, $\mathsf{erfc}(\theta)$ should be less than the inverse of the number of secret bits ℓ. Using the approximation $\Phi(-x) \approx \varphi(x)/x$ when x is large (so $\Phi(-x)$ is small) we obtain

$$\theta = \sqrt{\ln \ell} \quad \Longrightarrow \quad \mathsf{erfc}(\theta) = 2\Phi(-\theta\sqrt{2}) \approx 2\frac{\varphi(\theta\sqrt{2})}{\theta\sqrt{2}} = \frac{e^{-\theta^2}}{\theta\sqrt{\pi}} < \frac{1}{\ell} ,$$

and thereby a reasonable choice for θ.

Recovering the whole secret key. Algorithm 2 shows how to recover the secret key by building a system of linear equations with the help of Algorithm 1.

Clearly the complexity of Algorithm 2 is $\theta^2(4R(\bar{w}) + mR(\bar{w}))$ and we have to call it ℓ/m times on independent $(\bar{a}, \bar{b})$ pairs to fully recover X and Y, where ℓ is the length of the secret key (Note that $\ell = (k_x + k_y)m$ in RANDOM-HB$^{\#}$ and $\ell = k_x + k_y + 2m - 2$ in HB$^{\#}$). The expected number of errors in the equation system defining X and Y is $\ell \cdot \mathsf{erfc}(\theta)$. The probability that a passive attack gives an $(\bar{a}, \bar{b})$ linearly dependent from the i previous ones is $\frac{2^{i-1}}{2^{k_x+k_y}}$. The number of

Algorithm 2. Getting linear equations for X and Y

Input: $\bar{a}, \bar{b}, \bar{z}$ and $\bar{w}_{\mathsf{est}}$ the expected weight of $\bar{\nu} = \bar{a}X \oplus \bar{b}Y \oplus \bar{z}$
Output: A linear equation $\bar{a}X \oplus \bar{b}Y = \bar{c}$
Processing:
1: Initialize m-bit vector $\bar{c} \leftarrow \bar{z}$
2: Call Algorithm 1 on input $(\bar{a}, \bar{b}, \bar{z}, n = 4\theta^2 R(\bar{w}_{\mathsf{est}}))$ to get $\bar{w}$
3: **for** $i = 1 \dots m$ **do**
4: Flip bit i of $\bar{z}$ to get $\bar{z}'$
5: Call Algorithm 1 on input $(\bar{a}, \bar{b}, \bar{z}', n = \theta^2 R(\bar{w}))$ to get $\bar{w}'$
6: **if** $\bar{w}' = \bar{w} - 1$ **then**
7: $\bar{c}_i \leftarrow \bar{c}_i \oplus 1$
8: **end if**
9: **end for**

passive attacks to get the inputs for Algorithm 2 is thus and can be neglected in comparison to the ℓ/m calls of Algorithm 2.

$$C = \sum_{i=1}^{\lceil \ell/m \rceil} \frac{1}{1 - \frac{2^{i-1}}{2^{k_x + k_y}}} < 2 + \frac{\ell}{m} \tag{3}$$

Computational complexity. The computational complexity of the given attack is quite low in comparison to the number of authentications needed: For each call of Algorithm 1 we have at most n incrementation of a counter and one evaluation of P^{-1}. For RANDOM-HB$^{\#}$, after running Algorithm 2 we have m linear binary equation systems in $k_x + k_y$ variables (one for each row of the matrix $[X^\top | Y^\top]$), which can thus be solved in $O(m(k_x + k_y)^3)$ operations. This number is negligible in comparison to the number of authentications needed to perform Algorithm 2 and is even lower for HB$^{\#}$. Throughout the paper we thus measure the complexity of our attack in terms of (intercepted) authentications between the tag and the reader.

2.2 Asymptotic Complexity Analysis

The complexity of the attack is related to the complexity of Algorithm 2 which is in its turn related to the complexity of Algorithm 1. Thus, the main component of the attack affecting the overall complexity is the input n in Algorithm 1. Equation (2) yields that $n = O((\theta^2 e^{\frac{u^2}{2}})/(1-2\eta)^2)$ so the complexity of our attack is exponential in u^2 as we can use a θ logarithmic in ℓ.

Parameters with optimal complexity. The minimal value of n is reached when $u = 0$ which happens when the estimated value $\bar{w}_{\mathsf{est}}$ of $\mathsf{wt}(\bar{\nu})$ is

$$\bar{w}_{\mathsf{est}} = \bar{w}_{\mathsf{opt}} = \frac{t - m\eta}{1 - 2\eta} \ .$$

In this case we obtain

$$P(\bar{w}_{\mathsf{opt}}) = \frac{1}{2} ,$$
$$P'(\bar{w}_{\mathsf{opt}}) = -\frac{1-2\eta}{\sqrt{2\pi m\eta(1-\eta)}},$$
$$R(\bar{w}_{\mathsf{opt}}) = \frac{\pi m}{4}\left(\frac{1}{(1-2\eta)^2} - 1\right).$$

Obviously, our attack has optimal complexity if we can call Algorithm 2 on input of valid triplets $(\bar{a}, \bar{b}, \bar{z})$ with $\mathsf{wt}(\bar{\nu}) = \bar{w}_{\mathsf{opt}}$, only. As clearly, for most parameter sets the latter is not true for random triplets obtained by passive attacks, we would like to manipulate errors in $\bar{z}$ to reach an expected value of $\bar{w}_{\mathsf{opt}}$. Unfortunately, due to the hardness of the LPN problem, we cannot *remove* errors from $\bar{z}$ if $\bar{w} > \bar{w}_{\mathsf{opt}}$. However, if $\bar{w} \leq \bar{w}_{\mathsf{opt}}$ then we can *inject* errors in $\bar{z}$ so that the resulting vector has an expected weight of $\bar{w}_{\mathsf{opt}}$ and the attack remains polynomial. This case happens when:

$$m\eta \leq \frac{t - m\eta}{1-2\eta} \quad \Longleftrightarrow \quad t \geq 2m\eta(1-\eta) ,$$

using the approximation $\bar{w}_{\mathsf{est}} \approx m\eta$ when a valid triplet $(\bar{a}, \bar{b}, \bar{z})$ is obtained by a passive attack and the false rejection rate of the HB$^{\#}$ protocol is negligible. Thus in this case, our attack remains optimal.

Categorization of parameter sets. We have seen, that for $u = 0$, our attack has subquadratic running time. However, even if $u = O(\sqrt{\ln \ell})$, we obtain a polynomial time attack. Thus, from Formula (2) we distinguish three cases:

1. *Subquadratic complexity:* If $t \geq 2m\eta(1-\eta)$ the attack has a complexity of $O(\frac{\ell \ln \ell}{(1-2\eta)^2})$ since Algorithm 1 is called $O(\ell)$ times.
2. *Polynomial complexity:* $t = 2m\eta(1-\eta) - c\sqrt{m\eta(1-\eta)}, c = O(\sqrt{\ln \ell})$: the above complexity is multiplied by an e^{c^2} factor. Thus, Algorithm 1 is still polynomial.
3. *Exponential complexity:* All other cases.

Depending on the category of the parameter set, there are different strategies to find the triplets $(\bar{a}, \bar{b}, \bar{z})$ which serve as input for Algorithm 2 (and thus Algorithm 1). We present those strategies in the following and give numbers for the according parameter sets.

2.3 Strategy for the Case $t \geq 2m\eta(1-\eta)$

Thanks to the hypothesis $t \geq 2m\eta(1-\eta)$, we have that $\bar{w}_{\mathsf{opt}} \geq \bar{w} = m\eta$. Thus, the best strategy is to optimize the complexity of Algorithm 1 by having a triplet $(\bar{a}, \bar{b}, \bar{z})$ with an error vector of expected Hamming weight $\bar{w}_{\mathsf{opt}}$. Using a triplet $(\bar{a}, \bar{b}, \bar{z})$ obtained from a passive attack, we can flip the last $(\bar{w}_{\mathsf{opt}} - m\eta)/(1-2\eta)$ bits of $\bar{z}$ to get $\bar{\nu}$ of expected Hamming weight $\bar{w}_{\mathsf{opt}}$ and then use the attack described previously.

Application to parameter vector II. As these parameters are in the case $t \geq 2m\eta(1-\eta)$, we can use Algorithm 2 in its optimum complexity to attack both RANDOM-HB# and HB#. After computing $\bar{w}_{\mathsf{opt}} = 77.167$, $P'(\bar{w}_{\mathsf{opt}}) = 0.0431$, $R(\bar{w}_{\mathsf{opt}}) = 269.39$ and the expected value of $\bar{w} = m\eta = 55$, we have to flip $f = 29$ bits to get an expected value close to $\bar{w}_{\mathsf{opt}}$. For RANDOM-HB# the number of bits to retrieve is $\ell = (k_x + k_y)m = 261\,072$ for which we can use $\theta = 3.164$. The total complexity is $\ell\theta^2 R(\bar{w}_{\mathsf{opt}}) = 2^{29.4}$. In the case of HB# the number of secret bits is $\ell = k_x + k_y + 2m - 2 = 1\,472$ for which we use $\theta = 2.265$ and end up with complexity of $\ell\theta^2 R(\bar{w}_{\mathsf{opt}}) = 2^{21}$.

2.4 Strategy for t Close to $2m\eta(1-\eta)$

The case $t < 2m\eta(1-\eta)$ is trickier to address since the expected value of $\bar{w}$ becomes *greater* than w_{opt}. To achieve the same complexity as the previous case we would have to reduce the Hamming weight of $\bar{\nu}$ which is infeasible in polynomial time due to the hardness of the LPN problem.

However, if t is a only a little less than $2m\eta(1-\eta)$ then the expected value of $\bar{w}$ is not far from w_{opt}. So, we can use Algorithm 2 without flipping any bit of $\bar{z}$ and the complexity is still polynomial. To further speed up the attack, we can remove errors from $\bar{z}$ in step 9 of Algorithm 2 until we reach $\bar{w} = w_{\mathsf{opt}}$ which we can expect to happen at iteration $i = \left\lceil \frac{\bar{w}_{\mathsf{est}} - \bar{w}_{\mathsf{opt}}}{\bar{w}_{\mathsf{est}}} \right\rceil$.

Application to parameter set I. For parameter set I we have $t < 2m\eta(1-\eta)$. We first compute $\bar{w}_{\mathsf{est}} = m\eta = 291$, $\bar{w}_{\mathsf{opt}} = 228$, $P'(\bar{w}_{\mathsf{opt}}) = 0.0135$, $R(\bar{w}_{\mathsf{est}}) = 15\,532$ and $R(\bar{w}_{\mathsf{opt}}) = 2742.6$. For RANDOM-HB#, the number of key bits is $\ell = (k_x + k_y)m = 689\,088$ and $\theta = 3.308$ is enough to guarantee that $\mathsf{erfc}(\theta) \leq \frac{1}{689\,088}$. We obtain a total complexity of $\ell\theta^2(\frac{\bar{w}_0 - \bar{w}_{\mathsf{opt}}}{\bar{w}_{\mathsf{est}}} R(\bar{w}_{\mathsf{est}}) + \frac{\bar{w}_{\mathsf{opt}}}{\bar{w}_{\mathsf{est}}} R(\bar{w}_{\mathsf{opt}})) = 2^{35.4}$. For HB#, we have $\ell = k_x + k_y + 2m - 2 = 2\,918$ secret bits to retrieve, so $\theta_2 = 2.401$ is enough and we get a total complexity of $\ell\theta^2(\frac{\bar{w}_0 - \bar{w}_{\mathsf{opt}}}{\bar{w}_{\mathsf{est}}} R(\bar{w}_{\mathsf{est}}) + \frac{\bar{w}_{\mathsf{opt}}}{\bar{w}_{\mathsf{est}}} R(\bar{w}_{\mathsf{opt}})) = 2^{26.6}$.

2.5 Strategy for Lower t

The case of lower t, the false acceptance rate will be very low but the false rejection rate of HB# becomes high (e.g. 0.5 for $t = m\eta$; Please remember that for $t < m\eta$, HB# is no longer provable secure in the GRS security model.) so that it would require more than one authentication in average for the tag to authenticate itself. The main advantage of this approach is that the complexity of Algorithm 1 becomes exponential. Here, we present a better strategy than calling Algorithm 2 with an triplet $(\bar{a}, \bar{b}, \bar{z})$ obtained by a simple passive attack.

Our goal is to call Algorithm 2 with a $\bar{w}_{\mathsf{est}}$ as low as possible. During the protocol, we can set $(\hat{a}, \hat{b}, \hat{z})$ to $(a, b, z \oplus \bar{\nu})$ with $\bar{\nu}$ of weight $\bar{w}$ until the reader accepts $\hat{z}$. Then, we launch our attack with $(\bar{a}, \bar{b}, \bar{z}) = (a, b, z)$. A detailed description is shown in Algorithm 3.

Algorithm 3. Getting (a, b, z) with low Hamming weight

Input: $\bar{w}$
Output: (a, b, z) such that $(aX \oplus bY \oplus z)$ has low weight.
Processing:
1: Pick random vector $\bar{\nu}$ of Hamming weight $\bar{w}$
2: **repeat**
3: During a protocol with messages (a, b, z), set $\hat{z} = z \oplus \bar{\nu}$
4: **until** reader accepts

The probability that $\hat{z}$ gets accepted by the verifier is $P(\bar{w})$ which can be written in an equivalent way to Equation (1) as:

$$P(\bar{w}) = \sum_{j=0}^{t} \left(\binom{m-\bar{w}}{j} \eta^j (1-\eta)^{m-\bar{w}-j} \cdot \sum_{i=0}^{t-j} \binom{\bar{w}}{i} \eta^{\bar{w}-i} (1-\eta)^i \right) \tag{4}$$

For an accepted $\hat{z}$, the $m - \bar{w}$ positions not in the support of $\bar{\nu}$ are erroneous with probability

$$\eta_{\bar{w}} = \frac{\sum_{j=0}^{t} \left(j \binom{m-\bar{w}}{j} \eta^j (1-\eta)^{m-\bar{w}-j} \sum_{i=0}^{t-j} \binom{\bar{w}}{i} \eta^{\bar{w}-i} (1-\eta)^i \right)}{(m-\bar{w}) P(\bar{w})}. \tag{5}$$

On the other hand, the other positions of $\hat{z}$ in the support of $\bar{\nu}$ are non-zero with probability

$$\eta^{\circ}_{\bar{w}} = \frac{\sum_{j=0}^{t} \left(\binom{m-\bar{w}}{j} \eta^j (1-\eta)^{m-\bar{w}-j} \cdot \sum_{i=0}^{t-j} i \binom{\bar{w}}{i} \eta^{\bar{w}-i} (1-\eta)^i \right)}{\bar{w} P(\bar{w})}. \tag{6}$$

Thus, because of the high false rejection rate, if $\hat{z}$ gets accepted in our MIM-Attack with $(\bar{a}, \bar{b}, \bar{z}) = (0, 0, \bar{\nu})$, we can expect that the error vector ν, introduced in (a, b, z) the output of Algorithm 3, has weight $\bar{w}_{\mathsf{est}} = (m - \bar{w})\eta_{\bar{w}} + \bar{w}(1 - \eta^{\circ}_{\bar{w}})$.

Application to parameter set II with $t = 55$. Assume that for the parameter set II we set $t = m\eta \approx 55$. Then, an accepted vector obtained by a passive attack will most likely have weight $\bar{w}_{\mathsf{est}} = (m - \bar{w})\eta_0 + \bar{w}(1 - \eta^{\circ}_0) \approx 50$ and it will take $4\theta^2 R(\bar{w}_{\mathsf{est}}) = 2^{30}$ operations to determine its correct weight. Calling Algorithm 3, e.g., with $\bar{w} = 41$, we get (a, b, z) with error vector ν of weight $\bar{w}_{\mathsf{est}} = (m - \bar{w})\eta_{41} + \bar{w}(1 - \eta^{\circ}_{41}) \approx 33$ in $\frac{1}{P(\bar{w})} = 2^{20}$ authentications and can recover the weight of ν in another $4\theta^2 R(33) = 2^{20}$ operations with Algorithm 1. We determined the optimal input $\bar{w}$ by exhaustive search minimizing the sum of the complexity of the consecutive execution of Algorithms 3 and Algorithm 1.

The following table we consider parameter sets I and II with modified t. It shows the costs to learn one bit about the secret key, i.e. calling Algorithm 1 with a random vector obtained by a passive attack in comparison to calling Algorithm 3 first and then Algorithm 1 with its output. Note, that recovering successive bits is always cheaper.

Table 2. Attack cost for the initial bit of the shared key for HB# applied to $t = \lceil \eta m \rfloor$

Parameter set	Algorithm 1	Algorithms 3 + 1
I	2^{78}	$2^{58.5}$
II	2^{30}	2^{21}

3 Optimizing the Attack

In this Section, we present our best attack on RANDOM-HB# and HB#. First, we optimize Algorithm 2. Using an adaptive solution to the weighing problem [5] we show how to efficiently recover the error vector. Then, we present our full attack.

3.1 Optimizing Algorithm 2

The problem we are solving in Algorithm 2 can be formulated as follows: given a m-bit vector ν of Hamming weight w and an oracle measuring the sum of some selected bits (Algorithm 1), what is the minimal number of measurements to fully recover ν?

The naïve solution to this problem employed in Algorithm 2 takes m measurements. A more sophisticated solution to to fully recover a vector ν of arbitary weight was already given by Erdős and Rényi in [5]. They show that the minimal number of measurements required is upper-bounded by $(m \log_2 9)/\log_2 m$. To recover ν in the given complexity, they define a fixed series of measurements for each m. However, in our case, the vector ν is known to be of small weight ($\leq m\eta$), which allows us to improve on the solution by Erdős and Rényi. Our proposal, Algorithm 4, does not use a fixed series of measurements but takes into account the partial information obtained by all previous measurements.

To determine the error positions in a k-bit window by measuring the weight, Algorithm 4 uses a divide-and-conquer strategy: it splits the vector into two windows of the same length then measures each of them. For those parts which do not have full or zero weight it then applies this strategy recursively leading to a lower number of measurements comparing to measuring a k-bit window bit by bit as Algorithm 2 does.

The number of invocations of Algorithm 1, $C_w(k)$, to fully recover a k-bit window with known Hamming weight w by Algorithm 4 is

$$C_w(k) = \begin{cases} 0 & \text{if } w = 0 \text{ or } w = k \\ 1 + \sum_{i=0}^{k/2} \frac{\binom{\lfloor k/2 \rfloor}{i}\binom{\lceil k/2 \rceil}{w-i}}{\binom{k}{w}} \left(C_i(\lfloor k/2 \rfloor) + C_{w-i}(\lceil k/2 \rceil)\right) & \text{otherwise} \end{cases}$$

Let $C(k)$ be the average number of invocations of Algorithm 1 to first determine the number of errors in a k-bit window and then recover their positions using Algorithm 4:

$$C(k) = 1 + \sum_{w=0}^{k} C_w(k) \binom{k}{w} \eta^w (1-\eta)^{k-w}$$

Algorithm 4. Finding errors in $|J|$-bit windows

Input: $\bar{a}, \bar{b}, \bar{z}$, $\bar{w} = \mathsf{wt}(\bar{a}X \oplus \bar{b}Y \oplus \bar{z})$, a set $J \subseteq \{0, 1, \cdots m\}$ and w_J the number of non-zero $(\bar{a}X \oplus \bar{b}Y \oplus \bar{z})_j$, $j \in J$
Output: $I \subseteq J$ containing the j with non-zero $(\bar{a}X \oplus \bar{b}Y \oplus \bar{z})_j$, $j \in J$.
Processing:
1: **if** $w_J = 0$ **then**
2: $I \leftarrow \emptyset$
3: **else if** $w_J = |J|$ **then**
4: $I \leftarrow J$
5: **else**
6: Choose $J_1 \subseteq J$ such that $|J_1| = \lceil |J|/2 \rceil$.
7: Set ν' the m-vector with $\nu'_j = 1$ iff $j \in J_1$
8: Call Algorithm 1 on input $(\bar{a}, \bar{b}, \bar{z} \oplus \nu', n = 4\theta^2 R(\bar{w}))$ to get w'.
9: Call Algorithm 4 with $(\bar{a}, \bar{b}, \bar{z}, \bar{w}, J_1, w_{J_1} = (\bar{w} + |J_1| - w')/2)$ to get I_1
10: Call Algorithm 4 with $(\bar{a}, \bar{b}, \bar{z}, \bar{w}, J \setminus J_1, w_J - w_{J_1})$ to get I_2
11: $I \leftarrow I_1 \cup I_2$
12: **end if**

Table 3. Complexity of measuring a 16-bit window for parameter set II

	Parameter Set I		Parameter Set II	
k	$C(k)\frac{16}{k}$	Cost measurement	$C(k)\frac{16}{k}$	Cost measurement
2	11	$2^{15.95}$	9.75	$2^{12.43}$
4	9.72	$2^{15.96}$	7.404	$2^{12.49}$
8	9.51	$2^{15.99}$	6.71	$2^{12.75}$
16	9.51	$2^{16.11}$	6.69	$2^{13.90}$

We note that $C(k)/k$ is minimal when k is a power of 2. Although, it is clear from Table 3 that the number of measurements decreases when k increases, the cost of measuring the weight of a k-bit window also increases faster with k, so a good tradeoff is to use $k = 8$.

Now that we have an efficient algorithm to find error positions in fixed size windows, we introduce Algorithm 5 which takes benefit from Algorithm 4 to optimize the number of measurements needed to localize the introduced errors and output m linear equations. Algorithm 5 splits the error vector introduced in a triplet $(\bar{a}, \bar{b}, \bar{z})$ to m/k k-bit windows, each one of these is then recovered using Algorithm 4. Additionally, using the learned bits, it adjusts $\bar{z}$ so that the next measurements cost less. The number of calls to Algorithm 4 we need before we reach $\bar{w} = \bar{w}_{\mathsf{opt}}$, is then

$$i = \begin{cases} \frac{\bar{w}_{\mathsf{opt}} - \bar{w}_{\mathsf{est}}}{k(m - \bar{w}_{\mathsf{est}})} m & \text{if } \bar{w}_{\mathsf{opt}} \geq \bar{w}_{\mathsf{est}} \\ \frac{\bar{w}_{\mathsf{est}} - \bar{w}_{\mathsf{opt}}}{k \cdot \bar{w}_{\mathsf{est}}} m & \text{if } \bar{w}_{\mathsf{opt}} \leq \bar{w}_{\mathsf{est}} \end{cases}$$

Algorithm 5. Optimizing Algorithm 2

Input: $\bar{a}, \bar{b}, \bar{z}$ and $\bar{w}_{\mathsf{est}}$ the expected value of $\bar{\nu} = \bar{a}X \oplus \bar{b}Y \oplus \bar{z}$, k
Output: A linear equation $\bar{a}X \oplus \bar{b}Y = \bar{c}$
Processing:
1: Initialize m-bit vector $\bar{c} \leftarrow \bar{z}$
2: Initialize $M \leftarrow \emptyset$
3: Call Algorithm 1 on input $(\bar{a}, \bar{b}, \bar{z}, n = 4\theta^2 R(\bar{w}_{\mathsf{est}}))$ to get $\bar{w}$
4: Define a set $\mathcal{S}$ of $J_i = \{ik + 1, \ldots, \min((i+1)k, m)\},\ i = 1 \ldots \lceil \frac{m}{k} \rceil$
5: **repeat**
6: Choose $J \in \mathcal{S}$
7: Call Algorithm 1 on input $(\bar{a}, \bar{b}, \bar{z} \oplus J, n = \theta^2 R(\bar{w}))$ to get $\bar{w}' = \mathsf{wt}(\bar{\nu}\ \mathsf{AND}\ J)$
8: Call Algorithm 4 with $(\bar{a}, \bar{b}, \bar{z}, \bar{w}, J, w_J = (\bar{w} + |J| - \bar{w}')/2)$ to get I
9: Set $\bar{c}_i \leftarrow \bar{c}_i \oplus 1$ for all $i \in I$
10: $M \leftarrow M \cup I$
11: Remove J from $\mathcal{S}$
12: **if** $\bar{w} > \bar{w}_{\mathsf{opt}}$ **then**
13: Flip $\min(|I|, \bar{w} - \bar{w}_{\mathsf{opt}})$ bits $\bar{z}_i$ for which $i \in I$
14: $\bar{w} \leftarrow \bar{w} - \min(|I|, \bar{w} - \bar{w}_{\mathsf{opt}})$
15: **else if** $\bar{w} < \bar{w}_{\mathsf{opt}}$ **then**
16: Flip $\min(|J \setminus I|, \bar{w}_{\mathsf{opt}} - \bar{w})$ bits $\bar{z}_i$ for which $i \in J \setminus I$
17: $\bar{w} \leftarrow \bar{w} + \min(|J \setminus I|, \bar{w}_{\mathsf{opt}} - \bar{w})$
18: **end if**
19: **until** $\mathcal{S}\emptyset$

So the full complexity of Algorithm 5 is given by

$$N = \theta^2 \left(iR(\bar{w}_{\mathsf{est}}) + \left\lceil \frac{m}{k} - i \right\rfloor R(\bar{w}_{\mathsf{opt}}) \right) C(k) \ .$$

3.2 Final Algorithm

The final attack is described in Algorithm 6. The idea is to get a vector with low expected weight using Algorithm 3 and then find all the erroneous positions inserted by the tag to obtain m linear equations and iterate this until we get enough equations to solve and find the secrets X and Y. To get the lower complexity, we can flip the last bits of $\bar{z}$ so that we end up with an expected weight of $\bar{w}_{\mathsf{opt}}$. We note that introducing errors in a full segment as defined by Step 4 of Algorithm 5 does not increase the needed number of measurements as $C_w(k) = C_{k-w}(k)$. Using Formula (3), we deduce the full complexity in terms of intercepted authentications as

$$\left\lceil \frac{\ell}{m} \right\rceil \theta^2 \left(iR(\bar{w}_{\mathsf{est}}) + \left\lceil \frac{m}{k} - i \right\rfloor R(\bar{w}_{\mathsf{opt}}) \right) C(k) \ + \ (2 + \frac{\ell}{m}) \frac{1}{P(w)} \ . \qquad (7)$$

Application to parameter set I. With input $k = 8$ and $w = 300$ we obtain $P(w) = 2^{-7}$, $\bar{w}_{\mathsf{est}} = 273$ and $\bar{w}_{\mathsf{opt}} = 228$, $i = 24$, $R(\bar{w}_{\mathsf{opt}}) = 2742.6$, $R(\bar{w}_{\mathsf{est}}) = 7\,026.4$. So the full complexity of the attack is then given by Equation

Algorithm 6. Final attack on RANDOM-HB$^{\#}$ and HB$^{\#}$

Input: k, w
Output: X, Y the secrets of the tag
Processing:
1: Initialize $\mathcal{S} \leftarrow \emptyset$
2: **for** $i = 1 \dots 2 + \left\lceil \frac{\ell}{m} \right\rceil$ **do**
3: Call algorithm 3 on input w to get $\bar{a}, \bar{b}, \bar{z}$ with an error vector of expected weight $\bar{w}_{\mathsf{est}} = (m-w)\eta_w + w(1-\eta_w^{\circ})$
4: **if** $\bar{w}_{\mathsf{opt}} > \bar{w}_{\mathsf{est}}$ **then**
5: Flip the last $(\bar{w}_{\mathsf{opt}} - m\eta)/(1-2\eta)$ bits of $\bar{z}$
6: Set $\bar{w}_{\mathsf{est}} \leftarrow \bar{w}_{\mathsf{opt}}$
7: **end if**
8: Call Algorithm 5 on input $(\bar{a}, \bar{b}, \bar{z}, \bar{w}_{\mathsf{est}}, k)$ to get m linear equations
9: Insert linear equations in $\mathcal{S}$
10: **end for**
11: Solve $\mathcal{S}$

(7) with θ and ℓ as in Section 2.4. This is 2^{25} sessions for HB$^{\#}$ and $2^{33.8}$ for RANDOM-HB$^{\#}$.

Application to parameter set II. In this case, we have $k = 8$, $w = 0$ and $\bar{w}_{\mathsf{est}} = 55$. We flip 29 bits to obtain an error vector of expected weight $\bar{w}_{\mathsf{opt}} = 77$, which yields $R(\bar{w}_{\mathsf{opt}}) = 269.39$ and $i = 0$. The complexity is $2^{19.7}$ sessions for HB$^{\#}$ and $2^{28.1}$ for RANDOM-HB$^{\#}$.

4 Attacking Parameter Vectors without False Rejections

To thwart the previous attacks without taking parameter sets with huge m or high false rejection rate, we could change the protocol so that the prover generates a vector ν of constant or bounded Hamming weight like it was proposed for parameter set III. In this section we will show that this leads to different attacks.

Assume that the prover accepts (a, b, z) iff $w = \mathsf{wt}(aX \oplus bY \oplus z) = t$, then from this triplet the attacker learns

$$\bigoplus_{i=1}^{m} (aX \oplus bY)_i = \bigoplus_{i=1}^{m} z_i \oplus \begin{cases} 1 & \text{if } t \text{ odd} \\ 0 & \text{if } t \text{ even} \end{cases}$$

It is possible to recover the matrices X and Y by sending $z \oplus \bar{\nu}$ instead of the Tag's response z to the Reader, where $\bar{\nu}$ is a m-bit vector of Hamming weight 2. Doing so, the attacker learns

$$(aX \oplus bY)\bar{\nu}^{\top} = z\bar{\nu}^{\top} \oplus \begin{cases} 1 & \text{if } \hat{z} \text{ accepted} \\ 0 & \text{if } \hat{z} \text{ rejected} \end{cases}$$

since the verifier accepts $\hat{z}$ on challenges a, b if there was exactly one error in the flipped positions, which is the case with probability $\binom{m-w}{1}\binom{w}{1}/\binom{m}{2}$.

The above approach may be generalized to the case where the Hamming weight of ν is bounded in the original protocol, i.e. if the verifier accepts if $w \leq t$ and the prover discards error vectors which are going to be rejected. This was suggested for the parameter vector III. Again, the attacker can replace the Tag's answer z by $z \oplus \bar{\nu}$ where $\bar{\nu}$ is of weight 2. Now, the attackers response $z \oplus \bar{\nu}$ gets rejected iff $w \in \{t-1, t\}$ and the attacker flipped two non-erroneous positions. Thus, in the case of a rejection, the attacker learns

$$(aX \oplus bY)_i = z_i, \quad \bar{\nu}_i \neq 0$$

which happens with probability

$$q = \frac{\sum_{i=0}^{1} \binom{m}{t-i} \eta^{t-i} (1-\eta)^{m-t+i} \frac{\binom{m-t+i}{2}}{\binom{m}{2}}}{\sum_{i=0}^{t} \binom{m}{i} \eta^i (1-\eta)^{m-i}}$$

Application to parameter set III. For the parameter vector III, the attacker learns two bits about the secret key every $1/q = 2^{9.02} \approx 512$ iterations. This is 16 times faster than an attack by Algorithm 1 and needs only $\ell \cdot 2/q = 2^{26}$ authentications to recover a RANDOM-HB# secret key (2^{19} for HB#).

5 Lower Bounds on Secure Parameters

In this section, we investigate the lower bounds on the parameter sets for which our attack is not effective. We say that HB# is secure if recovering one bit of information about the secret key requires an attack with complexity (in terms of protocol sessions) within an order of magnitude of at least 2^s and time complexity "reasonably comparable".

Let us assume that Algorithm 3 succeeds with a total error weight of $t = \mathsf{wt}(\nu \oplus \bar{\nu})$ when the added error vector has weight $\bar{w}$. To obtain this vector, the attacker limited to 2^{80} operations can choose the input $\bar{w}$ in any way, such that $1/P(\bar{w}) = 1/\Phi(\frac{t-\mu}{\sigma}) \leq 2^{80}$. Since $\Phi(-10.2) \approx 2^{-80}$ we can be sure, that the $\bar{w}$ chosen by the attacker satisfies that

$$\begin{aligned}
\tfrac{t-\mu}{\sigma} = \quad & \tfrac{t-(m-\bar{w})\eta-\bar{w}(1-\eta)}{\sqrt{m\eta(1-\eta)}} \geq -10.2 \\
\Leftrightarrow\ & (m-\bar{w})\eta + \bar{w}(1-\eta) \leq 10.2\sqrt{m\eta(1-\eta)} + t \\
\Leftrightarrow\ & -\bar{w}\eta + \bar{w}(1-\eta) \leq 10.2\sqrt{m\eta(1-\eta)} + t - m\eta \\
\Leftrightarrow\ & \bar{w}(1-2\eta) \leq -m\eta + t + 10.2\sqrt{m\eta(1-\eta)} \\
\Leftrightarrow\ & \bar{w} \leq \tfrac{1}{1-2\eta}(10.2\sqrt{m\eta(1-\eta)} + t - m\eta)\ .
\end{aligned} \tag{8}$$

Fixing $t = \lfloor m\eta \rceil$ for which our attack has the maximal complexity, we get the lowest value for a secure m, thus $\bar{w} = \frac{10.2\sqrt{m\eta(1-\eta)}}{1-2\eta}$.

We can now calculate the value $\bar{w}_{\mathsf{est}}$ by using equations (4), (5) and (6) and then by using Formula (2) with $r = 1/2$ and $\theta = 1/2$ (which leads to $\mathsf{erfc}(\theta) = 0.4795$) we can estimate the total cost of the attack. By using an exhaustive

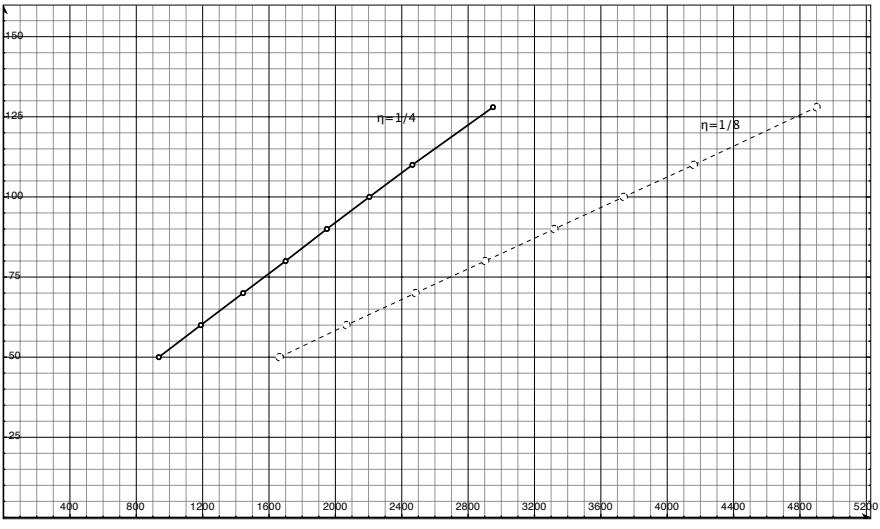

Fig. 2. Security level in logarithmic scale in comparison to m when $t = m\eta$

search on m we obtain that $m = 1\,697$ for $\eta = 1/4$ and $m = 2\,903$ for $\eta = 1/8$ is the lowest choice achieving 2^{80}-security and 50% of false rejection rate. The full results with the intermediates values are summarized in Table 4.

Table 4. Lowest values of m and $t = \lfloor m\eta \rceil$ for which our attack is not effective

η	m	t	$\bar{w}$	$\bar{w}_{\mathsf{est}}$	$\eta_{\bar{w}}$	$\eta^{\circ}_{\bar{w}}$	$1/P(\bar{w})$	n
0.25	1697	424	364	340	$2^{-2.73}$	$2^{-0.7}$	2^{80}	2^{80}
0.125	2903	363	242	229	$2^{-3.93}$	$2^{-0.36}$	2^{80}	2^{80}

Following this method we obtain the graphs of Fig. 2 showing how the security scales with growing m. To reach this security with a more acceptable false rejection rate (ideally negligible), it requires m to be higher.

6 Conclusion

In this article, we proved that the conjecture about the security of Random-HB$^{\#}$ and HB$^{\#}$ is wrong. We presented a basic attack against these protocols that allows to retrieve the shared secret between a reader and a tag. We showed a lower bound on the parameter set for which our attack is not effective but such parameters are unpractical to use in RFID tags.

Although it may not be the most effective for all versions, our attack is valid against all anterior protocols of the HB family.

Table 5. Summary of the complexity of our attacks

Parameter Set	k_X	k_Y	m	η	t	RANDOM-HB#	HB#
I	80	512	1164	0.25	405	2^{34}	2^{25}
II	80	512	441	0.125	113	2^{28}	2^{20}
III(w bounded)	80	512	256	0.125	48	2^{26}	2^{19}

There are still new versions in the HB family. PUF-HB, proposed by Hammouri and Sunar [10] uses a physical unclonable function but does not carry any proof of security against man-in-the-middle attacks within. Indeed, a closer look reveals several possible points of attack for a man in the middle like flipping the last bit in the challenge vector a. On the other side, Trusted-HB, proposed by Bringer and Chabanne [2], is proved secure against general man-in-the-middle attacks. However, this comes at the cost of adding a check on the integrity of the error vector using a secure cryptographic hash function which on its own would be sufficient to allow authentication by shared secrets.

Acknowledgment

We would like to thank Henri Gilbert, Matt Robshaw and Yannick Seurin for fruitful discussions.

References

1. Berlekamp, E.R., McEliece, R., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory 24(3), 384–386 (1978)
2. Bringer, J., Chabanne, H.: Trusted-HB: a low-cost version of HB$^+$ secure against man-in-the-middle attacks. CoRR, abs/0802.0603 (2008)
3. Bringer, J., Chabanne, H., Dottax, E.: HB^{++}: a lightweight authentication protocol secure against some attacks. In: Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2006), Lyon, France, June 29, pp. 28–33. IEEE Computer Society, Los Alamitos (2006)
4. Duc, D.N., Kim, K.: Securing HB$^+$ against GRS man-in-the-middle attack. In: Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security, Sasebo, Japan, January 23-26, p. 123 (2007)
5. Erdős, P., Rényi, A.: On two problems of information theory. Publ. Math. Inst. Hung. Acad. Sci. 8(21), 229–243 (1963)
6. Gilbert, H., Robshaw, M., Sibert, H.: Active attack against HB$^+$: a provably secure lightweight authentication protocol. IEEE Electronics Letters 41(21), 1169–1170 (2005)
7. Gilbert, H., Robshaw, M.J.B., Seurin, Y.: Good variants of HB$^+$ are hard to find. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 156–170. Springer, Heidelberg (2008)

8. Gilbert, H., Robshaw, M.J.B., Seurin, Y.: HB$^{\#}$: Increasing the security and efficiency of HB^{+}. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 361–378. Springer, Heidelberg (2008)
9. Gilbert, H., Robshaw, M.J.B., Seurin, Y.: HB$^{\#}$: Increasing the security and efficiency of HB^{+}, full version. Cryptology ePrint Archive, Report 2008/028 (2008)
10. Hammouri, G., Sunar, B.: PUF-HB: A tamper-resilient HB based authentication protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 346–365. Springer, Heidelberg (2008)
11. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
12. Juels, A., Weis, S.A.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
13. Katz, J., Shin, J.S.: Parallel and concurrent security of the HB and HB^{+} protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006)
14. Levieil, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
15. Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. Computer Networks 51(9), 2262–2267 (2007)