A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication

Marc P.C. Fossorier[†], Miodrag J. Mihaljević[•], Hideki Imai[◇], Yang Cui[‡] and Kanta Matsuura[‡] *

Abstract

A novel algorithm for solving the LPN problem is proposed and analyzed. The algorithm originates from the recently proposed advanced fast correlation attacks, and it employs the concepts of decimation, linear combining, hypothesizing and minimum distance decoding. The proposed algorithm appears as more powerful than the best one previously reported known as the BKW algorithm. In fact the BKW algorithm is shown to be a special instance of the proposed algorithm, but without optimized parameters. An improved security evaluation of the HB protocol for RFID authentication is then developed. Employing the proposed algorithm, the security of the HB protocol is reevaluated, implying that the previously reported security margins appear as overestimated.

keywords: cryptanalysis, LPN problem, fast correlation attacks, HB protocol, RFID authentication.

1 Introduction

Motivation for the Work.
Despite certain differences, both the LPN problem and the underlying problem of fast correlation attack can be viewed as the problem of solving an overdefined system of noisy linear equations. However, it appears that the currently reported approaches for solving the LPN problem do not take into account the approaches developed for fast correlation attacks. Accordingly, a goal of this work is to consider employment of fast correlation attack approaches for solving the LPN problem. Another motivation of this work is the security re-evaluation of

*[†]University of Hawaii, Department of Electrical Engineering, 540 Dole St., Holmes Hall 483, Honolulu, HI 96822, USA; [•]Mathematical Institute, Serbian Academy of Sciences and Arts, Kneza Mihaila 35, Belgrade, Serbia; [◇]Faculty of Science and Engineering, Chuo University 1-13-27 Kasuga, Bunkyo-ku, Tokyo, 112-8551, Japan, & Research Center for Information Security (RCIS), National Institute of Advanced Industrial Science and Technology (AIST), Room 1102, Akihabara Daibiru, 1-18-13 Sotokanda, Chiyoda-ku, Tokyo, 101-0021 Japan; [‡]University of Tokyo, Institute of Industrial Science 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan.

the HB protocol for RFID authentication as its security level appears as a direct consequence of the LPN problem hardness.

Summary of the Contributions.
This paper proposes a novel generic algorithm for solving the LPN problem. The proposed algorithm originates from the recently proposed advanced fast correlation attacks and it employs the following concepts: decimation, linear combining, hypothesizing and decoding. However, as opposed to fast correlation attacks, no preprocessing can be performed, which introduces an additional constraint. The following main characteristics of the proposed algorithm have been analytically established: (i) average time complexity; and (ii) average space complexity. The proposed algorithm has been compared with the best previously reported one, namely the BKW algorithm, and its advantages for solving the LPN problem have been pointed out. The proposed algorithm has been applied for security reevaluation of the HB protocol for RFID authentication implying that the previously reported security margins obtained based on the BKW algorithm are overestimated, and more realistic security margins have been derived.

Organization of the Paper.
Section 2 provides a brief review of the LPN problem and specifies it in a form relevant for further analysis. The BKW algorithm is presented in Section 3. A novel algorithm for solving the LPN problem is proposed and analyzed in Section 4. Comparisons between this algorithm and the BKW algorithms are made in Section 5. Security reevaluation of the HB protocol for RFID authentication via employment of the proposed algorithm is given in Section 6.

2 The LPN Problem

Let k be a security parameter. If $\mathbf{x}, \mathbf{g}_1, ..., \mathbf{g}_n$ are binary vectors of length k, let $y_i =< \mathbf{x} \cdot \mathbf{g}_i >$ denote the dot product of $\mathbf{x}$ and $\mathbf{g}_i$ (modulo 2). Given the values $\mathbf{g}_1, y_1$; $\mathbf{g}_2, y_2$; ..., $\mathbf{g}_n, y_n$, for randomly-chosen $\{\mathbf{g}_i\}_i^n$ and $n = O(k)$, it is possible to efficiently solve for $\mathbf{x}$ using standard linear-algebraic techniques.

However, in the presence of noise where each y_i is flipped (independently) with probability p, finding $\mathbf{x}$ becomes much more difficult. We refer to the problem of learning $\mathbf{x}$ in this latter case as the LPN problem.

For a formal definition of this problem, let Ber_p be the Bernoulli distribution with parameter p, $p \in (0, \frac{1}{2})$ (so if a random variable $E \sim \mathrm{Ber}_p$ then $\Pr[E = 1] = p$ and $\Pr[E = 0] = 1-p$) and let $A_{\mathbf{x},p}$ be the distribution defined by

$$\{\mathbf{g} \in \{0,1\}^k;\ e \leftarrow \mathrm{Ber}_p\ :\ (\mathbf{g}, < \mathbf{xg} > \oplus e)\}\ .$$

Also let $A_{\mathbf{x},p}$ denote an oracle which outputs (independent) samples according to this distribution. Algorithm M is said to (t, q, δ)-solve the LPN problem if

$$\Pr[\mathbf{x} \in \{0,1\}^k\ :\ M^{A_{\mathbf{x},p}}(1^k) = \mathbf{x}] \geq \delta\ ,$$

and furthermore M runs in time at most t and makes at most q queries to its oracle. In asymptotic terms, in the standard way, the LPN problem is *hard* if every probabilistic polynomial-time algorithm solves the LPN problem with only negligible probability (where the algorithm's running time and success probability are functions of k).

Note that p is usually taken to be a fixed constant independent of k, as will be the case in this work. The value of p to use depends on a number of tradeoffs and design decisions. Indeed the LPN problem becomes *harder* as p increases. However in certain authentication protocols where the security appears as a consequence of the LPN problem hardness, the larger the value of p is, the more often the honest prover becomes rejected.

The hardness of the LPN_p problem (for constant $p \in (0, \frac{1}{2})$ has been studied in many previous works. Particularly note that the LPN problem can be formulated also as the problem of decoding a random linear block code [1, 14] and has been shown to be NP-complete [1]. Beside the worst-case hardness results, there are numerous studies about the average-case hardness of the problem (see for example [2, 3, 5, 7, 14]). The most relevant result for this work is that the currently best-known algorithm for solving the LPN_p problem requires $t \sim q = 2^{O(k/logk)}$ [3]. More exact estimates of the running time of this algorithm, as well as suggested practical values for k are reported in [8, Appendix D].

In this work, we further investigate the formulation of the LPN_p problem as that of decoding a random linear block code by noticing that the rate k/n of this code is quite low and that the noise level p of the underlying binary symmetric channel is quite high. Both observations also hold for the fast correlation attack for which a similar parallelism with decoding a random linear code has been exploited [15, 10].

For this work, we reformulate the LPN problem after introducing the following notations:

- $\mathbf{x} = [x_i]_{i=1}^{k}$ is a k-dimensional binary vector;
- $\mathbf{G} = [\mathbf{g}_i]_{i=1}^{n}$ is a $k \times n$ binary matrix and each $\mathbf{g}_i = [g_i(j)]_{j=1}^{k}$ is a k-dimensional binary column vector;
- $\mathbf{y} = [y_i]_{i=1}^{n}$ and $\mathbf{z} = [z_i]_{i=1}^{n}$ are n-dimensional binary vectors;
- For each $i = 1, 2, ..., n$, e_i is a realization of a binary random variable E_i such that $Pr(E_i = 1) = p$ and $Pr(E_i = 0) = 1 - p$, and all the random variables E_i are mutually independent.

For given $\mathbf{G}$ and $\mathbf{x}$, the vectors $\mathbf{y}$ and $\mathbf{z}$ are specified as follows:

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{G} \ , \tag{1}$$

$$z_i = y_i \oplus e_i \ , \ i = 1, 2, ..., n \ . \tag{2}$$

Accordingly we have the following formulation of the LPN problem relevant for this paper.

LPN Problem. For given $\mathbf{G}$, $\mathbf{z}$ and p, recover $\mathbf{x}$.

3 The BKW Algorithm

3.1 Preliminaries

In [3], the BKW algorithm has been reported for solving the LPN problem. It is based on the following paradigm:
1. draw s strings (columns of the matrix $\mathbf{G}$);
2. partition the s strings into groups based on their last b bits;

3. choose a string from each partition, add it to every other string within that partition, and then discard it;
4. repeat on the procedure to the second-to-last b bits, third-to-last ... until all that remain are partitions in which only the first b bits are non-zero.

Note that in every step of the algorithm we perform poly$(k, 2^b, s)$ operations, since for each of the partitions (at most 2^b partitions), we add together at most s k-bit strings.

Based on the explanations given in [3], the BKW algorithm is re-written in an explicit algorithmic form in the next section.

3.2 Algorithm

- *Input*
  - matrix $\mathbf{G}$, vector $\mathbf{z}$ and the probability of error p.
- *Initialization*
  - Set the algorithm parameters: integers a and b such that[1] $ab \geq k$.
  - Select the parameter q according to:

    $$q = f((1-2p)^{-2^a}, b)$$

    where $f(\cdot)$ is a polynomial function.
  - Consider each $\mathbf{g_i}$ from $\mathbf{G}$ as consisting of a concatenated segments, labeled as $1, 2, ..., a$, each composed of b bits.
  - Set the algorithm parameter $\alpha = 1$.
- *Processing*

  1. Repeat the following steps (a) - (g) q times :
     (a) Randomly select a subset Ω of $a2^b$ previously not considered columns of $\mathbf{G}$.
     (b) Classify the elements of Ω into at most 2^b categories Ω_j such that all the vectors $\mathbf{g_i} \in \Omega_j$ have the identical segment of b bits labeled as a.
     (c) In each category Ω_j, randomly select a vector and do the following:
     - modify the vectors within the category by performing bit-by-bit XOR-ing of the selected vector with all other vectors in the category, yielding that all the modified vectors have all zeros in the last, a-th segment;
     - remove the selected vector from Ω_j;
     - form the updated / modified Ω as the union of all Ω_i; the expected number of elements in the updated Ω is $(a-1)2^b$.
     (d) Classify the elements of the current set Ω into at most 2^b categories Ω_j such that all the vectors $\mathbf{g_i} \in \Omega_j$ have the identical segment labeled as $a-1$, recalling that all the vectors contain the all zero b-tuple in the segment labeled as a.

[1] The method for selecting the values a and b is not relevant for the algorithm description itself.

(e) In each category Ω_j, randomly select a vector and do the following:
- modify the vectors within the category by performing bit-by-bit XOR-ing of the selected vector with all other vectors in the category, yielding that all the modified vectors have all zeros in the segment $a-1$ (as well as in the segment with label a) ;
- remove the selected vector from Ω_j;
- form the updated / modified Ω as the union of all Ω_i; the expected number of elements in the updated Ω is $(a-2)2^b$.

(f) Repeat $a-3$ times the procedure performed in the previous two steps, so that the last modification of Ω contains on average 2^b vectors with only zeros in all the segments with labels from 2 to a.

(g) For each $\ell = 1, 2, ..., b$, do the following:

i. Based on the vector, if it exists, from the current Ω with all zeros at the positions $1, 2, ..., \ell-1$ and $\ell+1, \ell+2, ..., k$, generate an estimate about x_ℓ which is correct with probability equal to $0.5 + 0.5(1-2p)^{2^{a-1}}$;

ii. If the targeting vector does not exist in the currently considered collection repeat the steps (a) - (f).

2. For each $\ell = 1, 2, ..., b$, do the following:
Employing majority logic based decoding on q individual estimates of x_ℓ, generate the final estimate on x_ℓ which is assumed correct with a probability close to 1 for q large enough.

3. Set $\alpha \to \alpha + 1$ and do the following:
   - if $\alpha > a$ go to Output step;
   - if $\alpha \leq a$ perform the following re-labeling and go to Processing Step 1:
     - $1 \to a$;
     - for each $i = 2, 3, ..., a$, re-label $i \to i-1$.

- *Output*
Estimation of $\mathbf{x}$.

3.3 Complexity Analysis

According to the structure of the BKW algorithm, and the results reported in [3, 8] we have the following statements.

Proposition 1, [3]. The required sample size and the time complexity of the BKW algorithm can be estimated as $f((1-2p)^{-2^a}, 2^b)$ where $f(\cdot)$ is a polynomial function.

Proposition 2, [8]. The BKW algorithm requires a sample of dimension proportional to $a^3 m 2^b$ where $m = \max\{(1-2p)^{-2^a}, b\}$.

Proposition 3, [8]. The time complexity of the BKW algorithm is proportional to $Ca^3 m 2^b$ where $m = \max\{(1-2p)^{-2^a}, b\}$, and C is a constant.

These estimates can be further refined. In particular, Proposition 3 can be put into a more precise form, and the space complexity of the BKW algorithm can be estimated via analysis

of the re-written BKW algorithm as follows.

Required Sample.
The required sample, i.e. the dimension n of the matrix $\mathbf{G}$ for the BKW algorithm execution depends on the following:
• Each execution of Step 1.(a) requires random drawing of $a2^b$ previously not considered columns of the matrix $\mathbf{G}$;
• The structure of the BKW algorithm implies that the expected number of executions of Step 1.(a) executions is proportional to $a^2q = a^2(1-2p)^{-2^a}$.

These considerations imply that the required sample of the BKW algorithm is proportional to $a^3(1-2p)^{-2^a}2^b$ which is in accordance with Proposition 2.
Time Complexity.
The time complexity of the BKW algorithm depends on the following:
• Each repetition of Steps 1.(a) to 1.(f) has time complexity proportional to $a2^b$;
• The expected number of repetitions of Steps 1.(a) to 1.(f) implied by Step 1.(g).(ii) is a (according to [3, 8]);
• Steps 1.(a) to 1.(g) should be repeated q times, with $q = (1-2p)^{-2^a}$;
• Step 3 requires that Steps 1 and 2 should be repeated a times;
• Each bit-by-bit mod2 addition of two k-dimensional vectors with all zeros in the last αb positions has cost proportional to $k - \alpha b$;
• The decoding of a bit of the vector $\mathbf{x}$ involves $(1-2p)^{-2^a}$ parity-checks, implying a complexity proportional to $(1-2p)^{-2^a}$ with a direct approach.

Based on these remarks, Proposition 2 can be reformulated in the following more precise form.

Proposition 4. The average time complexity of the BKW algorithm is proportional to $a^3(k/2)(1-2p)^{-2^a}2^b + k(1-2p)^{-2^a}$.

According to Proposition 4, the decoding time complexity per bit of the BKW algorithm is proportional to $a^3(1-2p)^{-2^a}2^{b-1} + (1-2p)^{-2^a}$.

Space Complexity.
The space complexity of the BKW algorithm is dominated by the dimension of the matrix $\mathbf{G}$ noting that its parameter n should be greater than the required sample.

Proposition 5. The space complexity of the BKW algorithm is proportional to $ka^3(1-2p)^{-2^a}2^b$.

4 A Novel Algorithm for Solving the LPN Problem

The novel algorithm for solving the LPN problem originates from the algorithms developed for the fast correlation attack against certain stream ciphers. However, there are also a few differences. The most substantial difference is that the developed algorithm does not contain any pre-processing phase. As a result, the pre-processing phase employed in fast correlation attacks has to be performed so that its computational cost becomes close to that of the processing phase itself.

4.1 Advanced Fast Correlation Attack Approach

The LPN problem is equivalent to the model of the underlying problem regarding cryptanalysis of certain stream ciphers. For this model of cryptanalysis, a number of powerful fast correlation attacks have been developed including these reported in [13, 4, 6, 12, 11].

In its general presentation, the advanced fast correlation attack (see [11]) is a certain decoding technique based on a pre-processing phase and a processing phase which employs:

- sample decimation;
- mapping based on linear combining;
- hypothesis testing;
- final decoding.

In the following section, a novel algorithm for solving the LPN problem is presented based on these steps [13, 4, 6, 11].

4.2 New Algorithm

The following algorithm uses three parameters b_H, b_0 and w which need to be optimized. The optimization of these values follows the presentation of the algorithm.

- *Input*
  - the matrix $\mathbf{G}$, the vector $\mathbf{z}$ and the probability of error p .
- *Initialization*
  - Select the algorithm parameters: b_H, b_0 and w.
  - Select the integer parameter q such that [6]:
$$q \geq (1-2p)^{-2w} .$$
  - Form the $(k+1) \times n$ matrix $\mathbf{G_e}$ obtained by adding the vector $\mathbf{z}$ to the top row of $\mathbf{G}$ (z_i becomes the 0-th element of the i-th column of $\mathbf{G_e}$).
- *Processing*
  1. **Phase I: Decimation of the matrix $\mathbf{G_e}$**
     (a) Search over the columns of the matrix $\mathbf{G_e}$ and select all the columns $\mathbf{g}_i$ such that $g_i(j) = 0$ for $j = k - b_0 + 1, k - b_0 + 2, ..., k$.
     (b) Form the decimated version $\mathbf{G}^*$ of the matrix $\mathbf{G_e}$ based on the columns selected in the previous step; $\mathbf{G}^*$ is a $(k+1) \times n^*$ matrix with on average, $n^* = 2^{-b_0} n$.
  2. **Phase II: Linear Combining of the Decimated Columns**
     (a) Search for mod2 sums of up to w columns of $\mathbf{G}^*$ such that the resultant vector has any weight in positions $j = 0$ to $j = b_H$ and weight 1 in positions $j = b_H + 1$ to $k - b_0$.

(b) For each $j = b_H + 1$ to $k - b_0$, record at least q such columns to form the matrix Ω_j.

3. **Phase III: Hypothesizing and Partial Decoding**
   Consider all 2^{b_H} possible hypotheses for bits $x_0, \cdots, x_{b_H-1}$ of $\mathbf{x}$ which correspond to rows $j = 1$ to $j = b_H$ in $\mathbf{G}^*$ and perform the following:
   (a) Select a previously not considered hypothesis on the first b_H bits of the vector $\mathbf{x}$; if no one new hypothesis is possible go to the Phase IV.
   (b) For the given hypothesis on the first b_H bits of the vector $\mathbf{x}$, employing the matrix Ω_j, estimate x_j based on

$$(1 \ \ \mathbf{x}) \cdot \Omega_j = \mathbf{0} \tag{3}$$

   for each $j = b_H + 1, \cdots k - b_0$. To this end, a majority rule on the all the decisions x_j needed to satisfy (3) is used.
   (c) Compute the Hamming weight of $(1 \ \mathbf{x}) \cdot \mathbf{G}^*$ to check the validity of the given hypothesis.
   (d) - Memorize the first $k - b_0$ positions of the L most likely vectors $\mathbf{x}$ found so far (list decoding of size L, with $L << 2^{b_H}$);
   - Go to the step (a) of Phase III.

4. **Phase IV: Final Decoding**
   (a) For the L vectors $\mathbf{x}$ recorded in Phase-III, repeat Phase-II based on $\mathbf{G_e}$ (or a punctured version $\mathbf{G_p}$) and Phase-III to estimate the decimated bits $x_{k-b_0}, \cdots, x_{k-1}$.
   (b) Select the most likely vector $\mathbf{x}$ based on the Hamming weight of $(1 \ \mathbf{x}) \cdot \mathbf{G_p}$.

- *Output*
  Estimation of $\mathbf{x}$.

4.3 Complexity Analysis of the Proposed Algorithm

For given parameters b_0, b_H and w, we obtain the following results.

Theorem 1. The average time complexity C of the proposed algorithm is dominated by:

$$C \sim (k - b_0) \left(\binom{n^*/2}{\lceil w/2 \rceil} + 2^{b_H} w \log_2(1-2p)^{-2} \right) \tag{4}$$

Proof. Denote by C_I, C_{II}, C_{III} and C_{IV} the average time complexities of the algorithm phases I to IV, respectively. According to the algorithm structure, the overall time complexity C is given as

$$C = C_I + C_{II} + C_{III} + C_{IV} \ . \tag{5}$$

The identification of the columns of the matrix $\mathbf{G}$ to construct $\mathbf{G}^*$ can be done during the sample collection phase and therefore has high level of parallelism. It follows that $C_I = \mathrm{O}(n)$ but this complexity (which may become dominant compared to C) is discarded as it can be assumed that $\mathbf{G}^*$ is given along with $\mathbf{G}$ at the beginning of the algorithm.

Phase-IV can be viewed as repeating the LPN problem to retrieve b_0 bits instead of $k-b_0$. As a result, for $b_0 < k/2$, C_{IV} can also be discarded as corresponding to solving the same problem, but for a smaller size. It follows $C \sim C_{II} + C_{III}$.

Phase-II can be viewed as constructing parity check equations of weight[2] $w+1$. Using the square-root algorithm proposed in [4] in conjunction with the hashing approach described in [16], the time complexity of this part is $\mathrm{O}\left(\binom{n^*/2}{\lceil w/2 \rceil}\right)$ for each position $j = b_H+1, \cdots, k-b_0$.

Phase-III can be viewed as evaluating the parity check equations found in Phase-II for each position $j = b_H+1, \cdots, k-b_0$. Again using the results of [4] based on Walsh transform, the resulting complexity for q parity check equations is $\mathrm{O}\left(2^{b_H} \log_2 q\right)$, where it is implicitly assumed $b_H \geq \log_2 q$ (which corresponds to most cases of interest). Based on [6], we need $q \geq (1-2p)^{-2w}$, so that for each j, the complexity of Phase-III becomes $\mathrm{O}\left(2^{b_H} w \log_2 (1-2p)^{-2}\right)$.

Joining together the established partial dominating complexities, we obtain the theorem statement.

Theorem 2. The average space complexity M of the proposed algorithm is dominated by:

$$M \sim (k-b_0)\left(n^* + \binom{n^*/2}{\lfloor w/2 \rfloor}\right) \tag{6}$$

Proof. The space complexity to store the matrix $\mathbf{G}^*$ is simply $(k-b_0)n2^{-b_0}$. The memory requirement needed to construct each matrix Ω_j based on the method of [4] is $\mathrm{O}\left(\binom{n^*/2}{\lfloor w/2 \rfloor}\right)$. Finally assuming again that $b_0 < k/2$, the memory requirement for Phase-IV can be discarded as in the worst case, it is of the same order as that considered in Phases I to III. The previous discussion directly imply the theorem statement.

Based on [6], we readily verify that the size of the sample given by Proposition 2 is larger than that required for the proposed algorithm with $w \leq 2^{a-1}$.

4.4 Optimization of the Parameters b_0 and b_H

Based on Theorem 1, we have $C \sim C_{II} + C_{III}$. With respect to the fast correlation attack, it can be observed that C_{II} corresponds to the pre-processing cost (searching for a sufficiently large number of parity check equations of a given form) and C_{III} corresponds to the processing cost (solving the system of parity check equations for a given sample). As a result, for the LPN problem, we have the additional constraint $C_{II} \sim C_{III}$.

For given values w and b_0, the value b_H which minimizes C_{III} is given by [6]

$$b_{H,opt} = 2w \log_2 (1-2p)^{-1} + k - b_0 - \log_2 \binom{n^*}{w}. \tag{7}$$

For this value, we have

$$\log_2 C_{III,opt} \sim 2w \log_2 (1-2p)^{-1} + k - b_0 - \log_2 \binom{n^*}{w} + \log_2 (k-b_0) + \log_2 w + \log_2 \log_2 (1-2p)^{-2}. \tag{8}$$

[2]Recall that columns of $\mathbf{G}$ summing to zero form codewords of the dual code, or equivalently parity check equations.

Table 1: Time complexities C_{II} and C_{III} for the LPN problem with $p = 0.25$.

n	k	w	$b_{0,opt}$	$b_{H,opt}$	$\log_2 C_{II}$	$\log_2 C_{III}$
2^{24}	32	2	16	5 ($> \log_2 q = 4$)	11	11
		4	18	3 ($< \log_2 q = 8$)		
2^{80}	224	4	47	58	70.5	67.9
		6	58	56	67.8	66.6
		8	63	57	66.7	67.7

Equating $C_{III,opt}$ with C_{II} given by

$$C_{II} \sim \log_2(k - b_0) + \log_2 \binom{n^*/2}{\lceil w/2 \rceil} \tag{9}$$

and solving for b_0, we obtain

$$\begin{aligned} b_{0,opt} \approx & \left(\frac{3w}{2} - 1\right)^{-1} \left(\frac{3w}{2}(\log_2 n^* - \log_2 w) - 2w \log_2(1-2p)^{-1}\right. \\ & \left. -k + \frac{w}{2} - \log_2 w - \log_2 \log_2 (1-2p)^{-2}\right) \end{aligned} \tag{10}$$

Based on (7) and (10), Table 1 summarizes the complexities C_{II} and C_{III} for $p = 0.25$ and values of n and k relevant to the LPN approach to attack the HB protocol [8]. For the $(2^{24}, 32)$ code, we observe that while for $w = 2$, C_{II} and C_{III} are efficiently balanced, the optimum value $b_{H,opt}$ for $w = 4$ no longer verifies $b_{H,opt} > w \log_2(1-2p)^{-2}$ so that the approach of [4] has reached its minimum cost. For the $(2^{80}, 224)$ code, we observe that different selections of w provide complexities C_{II} and C_{III} very close after proper optimization of both b_0 and b_H. In that case, selecting the smallest value w minimizes memory requirements.

5 Comparison of the Proposed Algorithm and the BKW Algorithm

It can be observed that the $(a-1)$ repetitions of the procedure to obtain vectors of weight 1 in the BKW algorithm is equivalent to constructing parity-check equations (or codewords in the dual codes) of weight $w = 2^{a-1} + 1$. Interestingly, both approaches require exactly the same number $(1-2p)^{2w}$ of check sums per bit. The results of Table 1 indicate that the choice of w is not critical in minimizing the time complexity of the proposed algorithm. However the techniques used to generate these check sums are very different as well as that to estimate the bits.

Table 2 compares the time and space complexities per information bit obtained from the results of Sections 3 and 4.

In order to establish a clearer comparison, we consider the special case $w = 2^{a-1}$ so that the same number of check sums is required by both approaches and $b = b_H = b_0$ (again this case corresponds to a meaningful comparison when applying these algorithms to the HB

Table 2: Time and space complexities per information bit of the BKW algorithm and the proposed algorithm.

	time complexity	space complexity
BKW algorithm	$\sim (k/b)^3(1-2p)^{-2^{k/b}}2^b$	$\sim (k/b)^3(1-2p)^{-2^{k/b}}2^b$
Proposed algorithm	$\sim \binom{n^*/2}{\lceil w/2 \rceil} + 2^{b_H}\log_2(1-2p)^{-2w}$	$\sim n^* + \binom{n^*/2}{\lfloor w/2 \rfloor}$

Table 3: Time and space complexities per information bit of the BKW algorithm and the proposed algorithm with $w = 2^{a-1}$ and $b = b_H = b_0$.

	time complexity	space complexity
BKW algorithm	$\sim (\log_2 w + 1)^3(1-2p)^{-2w}2^b$	$\sim (\log_2 w + 1)^3(1-2p)^{-2w}2^b$
Proposed algorithm	$\sim (1-2p)^{-w}2^{(k-2b)/2} + 2^b\log_2(1-2p)^{-2w}$	$\sim (1-2p)^{-w}2^{(k-2b)/2}$

protocol). The corresponding complexities are given in Table 3. We observe that even in this non optimized case, the proposed algorithm is more efficient in most cases. In particular for $a = 4$, $(k-2b)/2 = b$ so that a factor $(\log_2 w + 1)^3(1-2p)^{-w}$ is gained both in time and space complexity. The gain $(1-2p)^{-w}$ is mostly due to the application of the square-root algorithm to determine the check sums based on [4, 16]. However, gains even larger than $(\log_2 w + 1)^3(1-2p)^{-w}$ are available by proper selection of the parameters b_0 and b_H due to both decimation and hypothesis testing.

6 Security Re-Evaluation of the HB Protocol for RFID Authentication

Following the framework of a protocol for a person authentication to a computer reported in [7], the HB protocol has been proposed for RFID authentication in [8].

6.1 Security of the HB Protocol and LPN Problem

It is well known that the security of the HB protocol depends on the complexity of solving the LPN problem (see for example [8, 9])

The two main issues regarding the security evaluation of the HB protocol can be summarized as follows:

1. Collecting the sample for cryptanalysis via recording the challenges and responses exchanged during the protocol executions and forming the matrix $\mathbf{G}$ and the vector

$\mathbf{z}$ which define the underlying LPN problem; each challenge is recorded as a certain column $\mathbf{g_i}$ of the matrix $\mathbf{G}$, and its corresponding response is recorded as the element z_i of the vector $\mathbf{z}$.

2. Solving the obtained LPN problem.

Regarding the sample collection for cryptanalysis note the following issues:

- Each authentication session involves the same secret key, and consequently all the available challenge-response pairs can be jointly employed for recovering the secret key;

- Each authentication session involves r mutually independent challenges and responses, providing r columns of the matrix $\mathbf{G}$ and the corresponding r elements of the vector $\mathbf{z}$, so that via collecting the pairs from s authentication sessions, we obtain the matrix $\mathbf{G}$ and the vector $\mathbf{z}$ of dimensions $k \times n$ and n, respectively, where $n = s \cdot r$.

Accordingly, by collecting the information from different authentication sessions which involve the same secret key, we can obtain a huge sample for cryptanalysis, which is suitable for sophisticated processing in order to recover the secret key.

6.2 Security Evaluation of the HB Protocol Based on the BKW and Proposed Algorithms

The security evaluation considered in this section assumes the passive attacking scenario only. Following the approach for security evaluation employed in [8], we consider the security margin which measures the complexity of recovering a bit of the secret key.

As a simple illustrative example, we first consider the security evaluation of the HB protocol with $k = 32$, $p = 0.25$, assuming we can collect a sample of size $n = 2^{24}$. Accordingly, we consider the security margin of the protocol employing the BKW algorithm with parameters $a = 2$ and $b = 16$, as suggested in [8]. Based on Propositions 3 and 4, the expected time complexity of of the BKW algorithm is proportional to $ka^3(1-2p)^{-2^a}2^b = 32 \cdot 2^3 \cdot 2^4 \cdot 2^{16}$, so that the security margin (expected complexity per recovered secret key bit) is 2^{23}. For the proposed algorithm, we select $w = 2$, $b_0 = 16$ and $b_H = 5$ based on Table 1, so that the expected complexity per recovered secret key bit becomes $2 \cdot 2^{11}/16 = 2^8$. As a result, the proposed algorithm reduces the time complexity of the BKW algorithm almost to its cubic root for this simple example.

For security evaluation of the HB protocol with $k = 224$, $n = 2^{80}$ and $p = 0.25$, we have considered the BKW algorithm with $a = 4$ and $b = 56$, (see [8]) , while based on Table 1, we selected $w = 6$, $b_0 = 58$ and $b_H = 56$ for the proposed algorithm. The corresponding security margins are 2^{80} and 2^{61}, respectively. Again a significant gain has been achieved by the proposed algorithm. These results of security evaluation are summarized in Table 4

With respect to the special case considered Table 3, we observe that by proper optimization of all parameters, gains beyond the factor $(\log_2 w + 1)^3(1-2p)^{-w}$ have been achieved by the proposed algorithm over the BKW algorithm as this factor takes values 2^5 and 2^{14} for $k = 32$ and $k = 224$, respectively, while the corresponding gains are 2^{15} and 2^{19}.

Table 4: Security margins of the HB protocol against a passive attack, when the employed key consists of k bits and the employed noise corresponds to $p = 0.25$, based on the BKW and proposed algorithms.

number of secret key bits: k	security margin for the BKW algorithm [8]	security margin for the proposed algorithm
$k = 32$	$\sim 2^{23}$	$\sim 2^{8}$
$k = 224$	$\sim 2^{80}$	$\sim 2^{61}$

References

[1] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, "On the Inherent Intractability of Certain Coding Problems", *IEEE Trans. Info. Theory*, vol. 24, pp. 384-386, 1978.

[2] A. Blum, M. Furst, M. Kearns, and R. Lipton, "Cryptographic Primitives Based on Hard Learning Problems", CRYPTO '93, *Lecture Notes in Computer Science*, vol. 773, pp. 278-291, 1994.

[3] A. Blum, A. Kalai and H. Wasserman, "Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model", *Journal of the ACM*, vol. 50, no. 4, pp. 506-519, July 2003.

[4] P. Chose, A. Joux and M. Mitton, "Fast Correlation Attacks: An Algorithmic Point of View", EUROCRYPT2002, *Lecture Notes in Computer Science*, vol. 2332, pp. 209-221, 2002.

[5] F. Chabaud, "On the Security of Some Cryptosystems Based on Error-Correcting Codes. EUROCRYT '94, *Lecture Notes in Computer Science*, vol. 950, pp. 113-139, 1995.

[6] M.P.C. Fossorier, M.J. Mihaljević and H. Imai, "A Unified Analysis for the Fast Correlation Attack", *2005 IEEE Int. Symp. Inform. Theory - ISIT'2005*, Adelaide, Australia, Sept. 2005, Proceedings, pp. 2012-2015 (ISBN 0-7803-9151-9).

[7] N. Hopper and M. Blum, "Secure Human Identification Protocols", ASIACRYPT 2001, *Lecture Notes in Computer Science*, vol. 2248, pp. 52-66, 2001.

[8] A. Juels and S. Weis, "Authenticating Pervasive Devices with Human Protocols", CRYPTO2005, *Lecture Notes in Computer Science*, vol. 3621, pp. 293-308, 2005.
Updated version available at:
http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/pdfs/lpn.pdf

[9] J. Katz and J.S. Shin, "Parallel and Concurrent Security of the HB and HB+ Protocols", EUROCRYPT2006, *Lecture Notes in Computer Science*, vol. 4004, pp. 73-87, May 2006.

[10] W. Meier and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers," *Journal of Cryptology*, vol. 1, pp. 159-176, 1989.

[11] M.J. Mihaljević, M.P.C. Fossorier and H. Imai, "A General Formulation of Algebraic and Fast Correlation Attacks Based on Dedicated Sample Decimation", AAECC2006, *Lecture Notes in Computer Science*, vol. 3857, pp. 203-214, Feb. 2006.

[12] M.J. Mihaljević, M.P.C. Fossorier and H. Imai, "Cryptanalysis of Keystream Generator by Decimated Sample Based Algebraic and Fast Correlation Attacks", INDOCRYPT2005, *Lecture Notes in Computer Science*, vol. 3797, pp. 155-168, Dec. 2005.

[13] M.J. Mihaljević, M.P.C. Fossorier and H. Imai, "Fast Correlation Attack Algorithm with List Decoding and an Application", FSE2001, *Lecture Notes in Computer Science*, vol. 2355, pp. 196-210, 2002.

[14] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography", *37th ACM Symposium on Theory of Computing*, Proceedings, pp. 84-93, 2005.

[15] T. Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext Only," *IEEE Trans. Comput.*, vol. C-34, pp. 81-85, 1985.

[16] D. Wagner, "A Generalized Birthday Problem," CRYPTO '02, *Lecture Notes in Computer Science*, vol. 2442, pp. 288-304, 2002.