Observations on the LPN Solving Algorithm from Eurocrypt'16

Sonia Bogos and Serge Vaudenay

EPFL

Abstract. In this note we re-evaluate the Eurocrypt'16 paper by Zhang et al. [9] in the area of LPN solving algorithms. We present the history of LPN solving algorithms and give the general description of the algorithm from [9]. While this new algorithm claims to improve all the previous results, we have discovered issues in its analysis. We review inconsistencies in complexity estimates and a misconception of some new reduction algorithm. What we show is that the results of [9] do not provide better performance compared with the results from Asiacrypt'14 [5].

1 LPN

Despite the interesting ideas from [9], we have discovered that there are inconsistencies in the analysis of the LPN solving algorithm they introduce. In this note we explain where the mistakes arise and we provide the updated results with the corrected algorithm.

For our explanations we keep the same notations as the ones from Zhang et al. [9]. We present the definition of LPN as given in [9].

Definition 1 (LPN Problem). *Let Ber_η be the Bernoulli distribution, i.e., if $e \leftarrow Ber_\eta$ then $\Pr[e = 0] = 1-\eta$. Let $\langle \mathbf{x}, \mathbf{g} \rangle$ denote the scalar product of the vectors $\mathbf{x}$ and $\mathbf{g}$, i.e., $\mathbf{x} \cdot \mathbf{g}^{\mathbf{T}}$, where $\mathbf{g}^{\mathbf{T}}$ denotes the transpose of $\mathbf{g}$. Then an LPN oracle $\Pi_{LPN}(k, \eta)$ for an unknown random vector $\mathbf{x} \in \{0,1\}^k$ with a noise parameter $\eta \in (0, \frac{1}{2})$ returns independent samples by*

$$(\mathbf{g} \xleftarrow{\$} \{0,1\}^k, e \leftarrow Ber_\eta : \mathbf{g}, \langle \mathbf{x}, \mathbf{g} \rangle + e).$$

The (k, η)-LPN problem consists of recovering the vector $\mathbf{x}$ according to the samples output by the oracle $\Pi_{LPN}(k, \eta)$. An algorithm $\mathcal{S}$ is called (n, t, m, δ)-solver if $\Pr[\mathcal{S} = x : x \xleftarrow{\$} \{0,1\}^k] \geq \delta$, and it runs in time at most t and memory at most m with at most n oracle queries.

Thus, the LPN problem asks to recover the secret vector $\mathbf{x}$, given the queries which consist of uniformly random vectors and their noisy inner product with $\mathbf{x}$.

In the history of the LPN problem there are several algorithms that solve it in subexponential time, i.e. the algorithm runs in time $2^{\mathcal{O}(\frac{k}{\log k})}$ where k represents the size of the secret. These solving algorithms can be split into two phases: a reduction phase and a solving phase. During the reduction phase, we decrease the size of the secret to $k' \leq k$ and in the solving phase we recover the chunk of secret of k' bits. We can also consider a third phase, the back substitution, where we update the queries according to the result from the solving phase and restart the process with an LPN instance where the secret is of size $k-k'$. We will not consider the back substitution in this note as it is not considered in [9].

The first algorithm to solve LPN, BKW [1], is similar to a Gaussian elimination done on blocks. During the reduction phase, the secret is reduced up to a single bit. The solving phase recovers this bit by applying a majority rule. Improvements of it appeared in [7, 4]. The

algorithm of [7], called LF1, improves the solving phase by the use of the Walsh transform. This work also introduces a new reduction method, called LF2. At Asiacrypt'14, Guo et al. [5] improve the existing algorithms and introduce a new reduction method based on covering codes. An overview of all these algorithms can be found in [2].

2 Chains

The work from [9] is proposing a new LPN solving algorithm that has the following structure: given a (k, η)-LPN instance, the first step of their reduction phase is to do a sample selection. It keeps from the LPN oracle only those samples that have 0 bits on an entire given window of c bits. In this way, from a (k, η)-LPN instance we obtain a $(k - c, \eta)$-LPN instance. This step is called sample selection.

The next step is to change the distribution of the secret. Initially, we work with a uniformly distributed secret. It is a known result [6] that we can change this into a secret with the same distribution of the noise. This step is entitled Gaussian elimination.

With the new secret, the algorithm is further reducing the size of the secret by xoring pairs of vectors that have the same value on a window of b bits. This can be done by applying the BKW reduction step (LF1), LF2 or a new method which is named LF(4). By performing this step t times, we end up to a $(k - c - t * b, \eta')$-LPN instance, where $\eta' = \eta^{2^t}$ for LF1 and LF2 and $\eta' = \eta^{4^t}$ for LF(4). The use of the xoring is named collision procedure.

Using the fact that the distribution of the secret is the same as the one of the initial noise, the algorithm is further guessing k_1 bits of the secret. More precisely, it tries all possible values of the k_1 bit vector that have a Hamming weight smaller than w_1. To make the algorithm succeed, we must run it several times. Afterwards it is applying the covering code reduction to further reduce the LPN instance into a (ℓ, η^{2^t})-LPN instance where we use a $[k_2, \ell]$ code and $k_2 = k - c - t * b - k_1$.

The solving phase consists in applying the Walsh transform on the remaining ℓ bits. Given that the algorithm is guessing k_1 bits of the secret, the Walsh transform has to be instantiated $\sum_{i=0}^{w} \binom{k_1}{i}$ times.

Thus, in order to recover ℓ bits of the secret vector $\mathbf{x}$, the algorithm from [9] performs these steps:

- Sample selection (ss)
- Gaussian elimination (ge)
- Collision procedure (cp)
- Partial secret guessing (psg)
- Covering-coding (cc)
- Subspace hypothesis testing (WHT)

We use this sequence of steps for the $(512, \frac{1}{8})$, $(532, \frac{1}{8})$ and $(592, \frac{1}{8})$-LPN instances. Depending on the collision procedure chosen we have different results. Using the results from [9] we present in Tables 1-3 each reduction step for each instance. By $(k, n) - op - (k', n')$ we denote the fact that we apply one of the reduction phase operations (ss, ge, cp, psg, cc) and from an instance where the secret is of size k and we have n samples we reach an instance where the secret is of size k' and we are left with n' samples. Different results are obtained for LF1, LF2 and LF(4).

We provide below a table with the complexities used for every step of the algorithm from [9], where:

LPN instance	Algorithm using LF1
(512,1/8)-LPN	$(2^{71.291}, 512) - ss(5) - (2^{66.291}, 507) - ge - (2^{66.291}, 507) - cp(63) - (2^{66.135}, 444) - cp(63) - (2^{65.961}, 381) - cp(63) - (2^{65.762}, 318) - cp(63) - (2^{65.533}, 255) - cp(63) - (2^{65.259}, 192) - psg(20, 1) - (2^{65.259}, 172) - cc(172, 62) - (2^{65.259}, 62) - WHT$
(532,1/8)-LPN	$(2^{73.584}, 532) - ss(5) - (2^{68.584}, 527) - ge - (2^{68.584}, 527) - cp(65) - (2^{68.458}, 462) - cp(65) - (2^{68.320}, 397) - cp(65) - (2^{68.168}, 332) - cp(65) - (2^{67.998}, 267) - cp(65) - (2^{67.805}, 202) - psg(20, 1) - (2^{67.805}, 182) - cc(182, 64) - (2^{67.805}, 64) - WHT$
(592,1/8)-LPN	$(2^{79.557}, 592) - ss(4) - (2^{75.557}, 588) - ge - (2^{79.557}, 588) - cp(73) - (2^{79.541}, 515) - cp(73) - (2^{79.526}, 442) - cp(73) - (2^{79.510}, 369) - cp(73) - (2^{79.494}, 296) - cp(73) - (2^{79.478}, 223) - psg(16, 1) - (2^{79.478}, 207) - cc(207, 72) - (2^{79.478}, 72) - WHT$

Table 1. LPN solving algorithm using LF1 [9]

LPN instance	Algorithm using LF2
(512,1/8)-LPN	$(2^{69.987}, 512) - ss(5) - (\mathbf{2^{64.987}, 507}) - \mathbf{ge} - (\mathbf{2^{64.987}, 507}) - cp(64) - (2^{64.974}, 443) - cp(64) - (2^{64.948}, 379) - cp(64) - (2^{64.896}, 315) - cp(64) - (2^{64.792}, 251) - cp(64) - (2^{64.583}, 187) - psg(17, 1) - (2^{64.583}, 170) - cc(170, 62) - (2^{64.583}, 62) - WHT$
(532,1/8)-LPN	$(2^{73.983}, 532) - ss(7) - (2^{67.983}, 522) - ge - (2^{67.983}, 522) - cp(66) - (2^{66.966}, 459) - cp(66) - (2^{66.932}, 393) - cp(66) - (2^{66.863}, 327) - cp(66) - (2^{66.726}, 261) - cp(66) - (2^{66.453}, 195) - psg(17, 1) - (2^{66.453}, 178) - cc(178, 64) - (2^{66.453}, 64) - WHT$
(592,1/8)-LPN	$(2^{77.985}, 592) - ss(4) - (2^{73.985}, 588) - ge - (2^{73.985}, 588) - cp(73) - (2^{73.970}, 515) - cp(73) - (2^{73.940}, 442) - cp(73) - (2^{73.880}, 369) - cp(73) - (2^{73.759}, 296) - cp(73) - (2^{73.519}, 223) - psg(14, 1) - (2^{73.519}, 209) - cc(209, 72) - (2^{73.519}, 72) - WHT$

Table 2. LPN solving algorithm using LF2 [9]

- N is the initial number of queries from the LPN oracle, i.e. $N = 2^c n$
- n is the number of queries after the sample reduction
- k is the initial size of the secret vector
- t is the number of times the collision procedure is applied with parameter b
- a and u are two parameters that are used in the optimization of the Gaussian elimination step
- $n[i]$ represents the number of queries after i calls of the collision procedure. For LF1 $n[i] = n[i-1] - 2^b$, for LF2 $n[i] = \binom{n[i-1]}{2} 2^{-b}$ and for LF(4) we have $n[i] = \binom{n[i-1]}{4} 2^{-b}$
- f is a parameter used in the pre-computation of the collision procedure
- w_1 is an upper bound for the Hamming weight of the secret of size k_1
- m is the cost of performing the xor
- $k_2 = k - c - t * b - k_1$
- ℓ is the size of the secret at the end of the reduction phase

The final complexity of the algorithm is given by the following formula

$$C = C_0 + \frac{PC_{11} + C_1 + C_2 + C_3 + C_4 + C_5}{\Pr(w_1, k_1)},$$

where $\Pr(w_1, k_1) = \sum_{i=0}^{w_1} (1-\eta)^{k_1 - i} \eta^i \binom{k_1}{i}$. PC_{11} represents the cost of updating the queries during the Gaussian elimination. All the steps, except the first one, have to be repeated $\frac{1}{\Pr(w_1, k_1)}$ times because of the assumption that the secret of k_1 bits has a Hamming weight smaller or equal to w_1.

The results from [9] are displayed in Tables 5-7. We added a column to correct the time complexity as detailed hereafter. All the results are expressed in $\log_2$.

LPN instance	Algorithm using LF(4)
(512,1/8)-LPN	$(2^{63.526}, 512) - ss(10) - (2^{53.526}, 502) - ge - (\mathbf{2^{53.526}, 502}) - \mathbf{cp(156)} - (\mathbf{2^{53.519}, 346}) - cp(156) - (2^{53.490}, 190) - psg(16, 1) - (2^{53.490}, 174) - cc(174, 60) - (2^{53.490}, 60) - WHT$
(532,1/8)-LPN	$(2^{70.504}, 532) - ss(15) - (2^{55.504}, 517) - ge - (2^{55.504}, 517) - cp(162) - (2^{55.433}, 355) - cp(162) - (2^{55.149}, 193) - psg(13, 1) - (2^{55.149}, 180) - cc(180, 61) - (2^{55.149}, 61) - WHT$
(592,1/8)-LPN	$(2^{78.513}, 592) - ss(18) - (2^{60.513}, 574) - ge - (2^{60.513}, 574) - cp(177) - (2^{60.468}, 397) - cp(177) - (2^{60.288}, 220) - psg(16, 1) - (2^{60.288}, 204) - cc(204, 68) - (2^{60.288}, 60) - WHT$

Table 3. LPN solving algorithm using LF(4) [9]

Operation	Complexity
Sample selection	$C_0 = N$
Gaussian elimination	$C_1 = (n - k + c)(a + \lceil (k - c)/u \rceil)$
Update of queries from Gaussian elimination	$PC_{11} = (2^s - s - 1)(k - c)a$
Collision procedure (LF1)	$C_2 = \sum_{i=1}^{t} \lceil \frac{k-c+1-ib}{f} \rceil (n - k - c - i2^b)$
Collision procedure (LF2)	$C_2 = \sum_{i=1}^{t} \lceil \frac{k-c+1-ib}{f} \rceil n[i]$
Collision procedure (LF(4))	$C_2 = \sum_{i=1}^{t} (\lceil \frac{k-c+1-ib}{f} \rceil n[i] + (2^b n[i])^{1/3})$
Partial secret guessing	$C_3 = m \sum_{i=1}^{w_1} \binom{k_1}{i}$
Covering-coding	$C_4 = mh$
Subspace hypothesis testing	$C_5 = \ell 2^\ell \sum_{i=0}^{w_1} \binom{k_1}{i}$

Table 4. Complexities used by the LPN algorithm from [9]

3 LF(4)

The authors of [9] introduce a new reduction technique that they call LF(4) which is based on the Wagner algorithm [8]. This reduction method finds four vectors from a list which, when they are xored, give a zero. They use an information theoretic estimate on the required number of vectors for a solution to exist and the complexity results from Wagner.

If we have n vectors of b bits, information theory says we need $n \approx (4! \cdot 2^b)^{\frac{1}{4}}$. For this n, there is an algorithm of complexity $\frac{n^2}{2} \approx \sqrt{6} \cdot 2^{b/2}$ to find a solution (make a list of XOR of two strings then look for a collision). The Wagner algorithm works with the better time complexity of $2^{b/3}$ but needs a larger $n \approx 2^{b/3}$ because it looks for solutions such that the XOR of the first two vectors starts with $\frac{b}{3}$ zero bits. For this, make a list of XOR of two vectors colliding on their first $b/3$ bits then look for collisions of their XOR on the remaining bits. What the authors of [9] use is a data complexity of $2^{b/4}$ and a time complexity of $2^{b/3}$ by invoking Wagner. So, they mix up the two algorithms: they take the best data complexity of the two and the best time complexity of the two without realising that this applies to two different ones. For their algorithm, they need more than one solution so instead of having a complexity of $2^{b/3}$, they will have $2^{b/3} n_{sol}^{1/3}$ with the Wagner algorithm (see C_2 for LF(4) from Table 4), where n_{sol} represents the number of solutions for LF(4). The data complexity for the information theory algorithm becomes $n = (4! \cdot n_{sol} 2^b)^{\frac{1}{4}}$. By looking at our *cp* step from Table 3 in LPN(512,1/8), we can see that $n = n_{sol} = 2^{53.5}$, $b = 156$ is consistent with this formula.

LPN instance	Time	Initial data N	Memory	Pre-computation	Time corrected
$(512, 1/8)$	75.897	71.291	75.281	66.164	81.79
$(532, 1/8)$	78.182	73.584	77.629	68.053	84.06
$(592, 1/8)$	84.715	79.557	84.764	76.391	90.75

Table 5. Performance of LPN solving algorithm using LF1 [9]

LPN instance	Time	Initial data N	Memory	Pre-computation	Time corrected
$(512, 1/8)$	74.732	69.987	73.983	66.164	80.45
$(532, 1/8)$	76.902	73.983	76.028	68.053	82.53
$(592, 1/8)$	83.843	77.985	83.204	76.391	89.46

Table 6. Performance of LPN solving algorithm using LF2 [9]

They claim that their results are confirmed by simulations, but the results they provide miss the time complexity of the simulations.

Using the Wagner algorithm for LF(4) would mean applying LF2 operation two consecutive times.

4 Data Complexity

The data complexity N presented in Tables 5, 6, 7 and in [9] is given in k-bit strings while the memory complexity M is given in bits. So, when seeing a data complexity of $2^{69.987}$ for LF2 with $(512, 1/8)$-LPN instance (Table 6 of [9] and Table 6 in this note), one should understand $2^{78.987}$ bits. This is larger than the claimed time complexity and the claimed memory complexity.

Also, the cost of the first step, the sample selection, is taken to be $C_0 = N$, i.e. the number of queries. During this step, the algorithm discards any vector that is not 0 on a window of c bits. In order to check this, one would require to read the corresponding c bits, leading to a complexity of cN. The work of [9] assumes a random access to the input data. The input data needs not to be scanned nor to be stored. So, the complexity could be $(1 + \frac{1}{2} + \ldots + \frac{1}{2^{c-1}})N$, i.e. twice more than N. But this already hides a cost which is larger than what is claimed.

5 Complexity of Gaussian Elimination Step

In the second step, the algorithm is changing the distribution of the secret to that of the noise. We take the LPN instance of $(512, 1/8)$ for our explanation.

For this step, the memory complexity M essentially consists in storing the useful data in a $k \times n$ matrix G. For our instance, we have a matrix of $k \times n = 2^{73.983}$ bits with $k = 512$ and $n = 2^{64.983}$. The time complexity of $2^{74.732}$ is less than $2M$ as we can see in the Table 6 (and Table 6 of [9]). One operation of this reduction step consists of doing a Gaussian elimination on this big matrix G: compute $\hat{G} = D \cdot G$ for a $k \times k$ matrix D then $z\hat{G}$ for a k-bit vector z. Computing $z\hat{G}$ takes complexity M so [9] manages to compute $D \cdot G$ in less than $k \cdot n$ operations, with $k \cdot n$ being the size of G.

The authors of [9] claim to compute $D \cdot G$ in time $C_1 = (n - k + c)(a + (k - c)/u)$ using many precomputed tables. For LPN(512,1/8), we have $n = 2^{64.987}$ and $k - c = 507$ (See Table 1) and $C_1 = 2^{71.184}$ (they use $a = 10$ and $u = 8$ for this instance). First, they split

LPN instance	Time	Initial data N	Memory	Pre-computation	Time corrected
$(512, 1/8)$	72.844	63.526	68.197	68.020	117.14
$(532, 1/8)$	74.709	70.504	69.528	69.231	120.71
$(592, 1/8)$	81.963	78.513	70.806	69.231	131.28

Table 7. Performance of LPN solving algorithm using LF(4) [9]

$D = (D_1, ..., D_a)$ and $g = (g_1, ..., g_a)$ in a pieces and make the full tables of the $g_i \to D_i \cdot g_i^T$ functions. For each column g^T of G, they just need to compute $a = 10$ lookup tables and the sum $D_1 \cdot g_1^T + ... + D_a \cdot g_a^T$ of a k-bit vectors. If we do it normally, it costs $n \cdot a \cdot k$ bit operations (for all the n columns). This would represent a complexity of $2^{77.304}$ just for the cost of this sum.

Instead, in [9] the sum of the k-bit vectors is computed by assembling k/u sums of a u-bit vectors. To make a sum of a u-bit vectors, they rely on a table giving the sum at a cost of 1 lookup table for a table with 2^{au} entries of u bits.

One problem is that they assume that computing each $D_i \cdot g_i^T$ costs 1 (not even the cost to read the k bits it is supposed to produce) and computing the a-sum of each u-bit chunk costs 1 (not even the cost of reading the $a \cdot u$ bits of index for lookup). This is how they obtain a complexity of $a + k/u$ to treat each column of G. If instead we count the cost of reading the output of the $D_i \cdot g_i^T$ lookup tables, we already have the cost of ak that they wanted to avoid.

The second problem is in the size of the second lookup table which is $u \cdot 2^{au}$ which is too big. For the $(512, 1/8)$ instance of LPN, $u = 8$ and $a = 10$ this would give 2^{83}. To solve this issue, they say that we can only store in the table instances of this sum problem which have non-repeating and non-zero terms, as we know how to simplify other sums without more tables. But this assumes that instead of summing $a = 10$ bytes (since $u = 8$), we first have to eliminate zero-bytes and repetitions of bytes. This would imply to test if the bytes are zero or not and to compare them pairwise, or at least to sort them. The cost of this operation is clearly larger than the cost of summing 10 bytes, which counts at 80 bit-operations.

Considering that Gaussian elimination should rather take a complexity of at least $(n-k)ak$ and that we need to repeat $1/\Pr(w_1, k_1)$ times this procedure, we obtain that solving the $(512, 1/8)$ instance of LPN with LF2 must be larger than $2^{78.81}$ (take $a = k/s$ in Table 6 of [9]). This is only the cost of computing $\hat{G}$ to which we have to add other costs (the pre-computation, computing $z\hat{G}$, and other operations in the solving algorithm). Since a complexity of $2^{79.7}$ with the same type of method was already presented at Asiacrypt 2014 [5], we believe that careful computation on both algorithms with the same method will show there is no better result.

6 Complexity of the Hypothesis Testing

The complexity of the hypothesis testing in [9] is taken to be $\ell 2^\ell$, where ℓ is the size of the secret vector which is recovered. If we consider the complexity in bit operations (as it is done for other operations presented in [9]), the total complexity of computing the Walsh transform is $\ell 2^{\ell \frac{\log_2 n' + 1}{2}}$. This is because the entries of the transform have a magnitude of order $\sqrt{n'}$, where n' represents the number of vectors we have at this phase.

Also, before applying the Walsh transform, we require first to construct the function f on which we apply the transform. Thus, we need to add the cost of scanning the ℓ-bit vectors

which is $n\ell$. The total complexity becomes now $n\ell + \ell 2^{\ell \frac{\log_2 n+1}{2}}$. Also, the Walsh transform is computed for each vector guessed in the partial secret guessing step. For the $(512, 1/8)$ instance, with $n = 2^{64.583}$, $\ell = 62$, $k_1 = 17$, $w_1 = 1$ this gives $2^{77.401}$ which is much larger than the total complexity from Table 6 (and Table 6 from [9]).

7 Conclusion

We reran the results of [9] with the updated formulae. For LF(4) we use the information theoretical algorithm. The exact complexity formulae used for this computation can be found in [3]. We present below the complexity for each reduction step for the LPN instances $(512, 1/8)$, $(532, 1/8)$ and $(592, 1/8)$ with LF1, LF2 and LF(4). These results explain the columns "Time corrected" from Table 5-7.

LF1

(512,1/8)-LPN

$$\begin{aligned}
(2^{71.291}, 512) - ss(5) - (2^{66.291}, 507)\ & comp = 2^{72.245} \\
(2^{66.291}, 507) - ge - (2^{66.291}, 507)\ & comp = 2^{79.225} \\
(2^{66.291}, 507) - cp(63) - (2^{66.135}, 444)\ & comp = 2^{75.277} \\
(2^{66.135}, 444) - cp(63) - (2^{65.961}, 381)\ & comp = 2^{74.930} \\
(2^{65.961}, 381) - cp(63) - (2^{65.762}, 318)\ & comp = 2^{74.535} \\
(2^{65.762}, 318) - cp(63) - (2^{65.533}, 255)\ & comp = 2^{74.076} \\
(2^{65.533}, 255) - cp(63) - (2^{65.259}, 192)\ & comp = 2^{73.527} \\
(2^{65.259}, 192) - psg(20, 1) - (2^{65.259}, 172) & \\
(2^{65.259}, 172) - cc(172, 62) - (2^{65.259}, 62)\ & comp = 2^{72.686} \\
WHT(62)\ & comp = 2^{73.371} \\
& total\ comp = 2^{81.79}
\end{aligned}$$

(532,1/8)-LPN

$$\begin{aligned}
(2^{73.584}, 532) - ss(5) - (2^{68.584}, 527)\ & comp = 2^{74.538} \\
(2^{68.584}, 527) - ge - (2^{68.584}, 527)\ & comp = 2^{81.476} \\
(2^{68.584}, 527) - cp(65) - (2^{68.458}, 462)\ & comp = 2^{77.626} \\
(2^{68.458}, 462) - cp(65) - (2^{68.320}, 397)\ & comp = 2^{77.310} \\
(2^{68.320}, 397) - cp(65) - (2^{68.168}, 332)\ & comp = 2^{76.954} \\
(2^{68.168}, 332) - cp(65) - (2^{67.998}, 267)\ & comp = 2^{76.544} \\
(2^{67.998}, 267) - cp(65) - (2^{67.805}, 202)\ & comp = 2^{76.059} \\
(2^{67.805}, 202) - psg(20, 1) - (2^{67.805}, 182) & \\
(2^{67.805}, 182) - cc(182, 64) - (2^{67.805}, 64)\ & comp = 2^{75.313} \\
WHT(64)\ & comp = 2^{75.597} \\
& total\ comp = 2^{84.06}
\end{aligned}$$

(592,1/8)-LPN

$$
\begin{aligned}
(2^{79.557}, 592) - ss(4) - (2^{75.557}, 588)\ comp &= 2^{80.464} \\
(2^{75.557}, 588) - ge - (2^{79.557}, 588)\ comp &= 2^{88.675} \\
(2^{79.557}, 588) - cp(73) - (2^{79.541}, 515)\ comp &= 2^{84.757} \\
(2^{79.541}, 515) - cp(73) - (2^{79.526}, 442)\ comp &= 2^{84.297} \\
(2^{79.526}, 442) - cp(73) - (2^{79.510}, 369)\ comp &= 2^{83.746} \\
(2^{79.510}, 369) - cp(73) - (2^{79.494}, 296)\ comp &= 2^{83.056} \\
(2^{79.494}, 296) - cp(73) - (2^{79.478}, 223)\ comp &= 2^{82.124} \\
(2^{79.478}, 223) - psg(16, 1) - (2^{79.478}, 207) & \\
(2^{79.478}, 207) - cc(207, 72) - (2^{79.478}, 72)\ comp &= 2^{80.517} \\
WHT(72)\ comp &= 2^{83.443} \\
total\ comp &= 2^{90.75}
\end{aligned}
$$

LF2

(512,1/8)-LPN

$$
\begin{aligned}
(2^{69.987}, 512) - ss(5) - (2^{64.987}, 507)\ comp &= 2^{70.941} \\
(2^{64.987}, 507) - ge - (2^{64.987}, 507)\ comp &= 2^{78.045} \\
(2^{64.987}, 507) - cp(64) - (2^{64.974}, 443)\ comp &= 2^{73.973} \\
(2^{64.974}, 443) - cp(64) - (2^{64.948}, 379)\ comp &= 2^{73.765} \\
(2^{64.948}, 379) - cp(64) - (2^{64.896}, 315)\ comp &= 2^{73.514} \\
(2^{64.896}, 315) - cp(64) - (2^{64.792}, 251)\ comp &= 2^{73.195} \\
(2^{64.792}, 251) - cp(64) - (2^{64.583}, 187)\ comp &= 2^{72.764} \\
(2^{64.583}, 187) - psg(17, 1) - (2^{64.583}, 170) & \\
(2^{64.583}, 170) - cc(170, 62) - (2^{64.583}, 62)\ comp &= 2^{71.993} \\
WHT(62)\ comp &= 2^{73.232} \\
total\ comp &= 2^{80.45}
\end{aligned}
$$

(532,1/8)-LPN

$$(2^{73.983}, 532) - ss(7) - (2^{67.983}, 522)\ comp = 2^{74.972}$$
$$(2^{67.983}, 522) - ge - (2^{67.983}, 522)\ comp = 2^{80.115}$$
$$(2^{67.983}, 522) - cp(66) - (2^{66.966}, 459)\ comp = 2^{76.019}$$
$$(2^{66.966}, 459) - cp(66) - (2^{66.932}, 393)\ comp = 2^{75.808}$$
$$(2^{66.932}, 393) - cp(66) - (2^{66.863}, 327)\ comp = 2^{75.550}$$
$$(2^{66.863}, 327) - cp(66) - (2^{66.726}, 261)\ comp = 2^{75.217}$$
$$(2^{66.726}, 261) - cp(66) - (2^{66.453}, 195)\ comp = 2^{74.756}$$
$$(2^{66.453}, 195) - psg(17, 1) - (2^{66.453}, 178)$$
$$(2^{66.453}, 178) - cc(178, 64) - (2^{66.453}, 64)\ comp = 2^{73.932}$$
$$WHT(64)\ comp = 2^{75.293}$$
$$total\ comp = 2^{82.53}$$

(592,1/8)-LPN

$$(2^{77.985}, 592) - ss(4) - (2^{73.985}, 588)\ comp = 2^{78.892}$$
$$(2^{73.985}, 588) - ge - (2^{73.985}, 588)\ comp = 2^{86.931}$$
$$(2^{73.985}, 588) - cp(73) - (2^{73.970}, 515)\ comp = 2^{83.185}$$
$$(2^{73.970}, 515) - cp(73) - (2^{73.940}, 442)\ comp = 2^{82.978}$$
$$(2^{73.940}, 442) - cp(73) - (2^{73.880}, 369)\ comp = 2^{82.728}$$
$$(2^{73.880}, 369) - cp(73) - (2^{73.759}, 296)\ comp = 2^{82.407}$$
$$(2^{73.759}, 296) - cp(73) - (2^{73.519}, 223)\ comp = 2^{81.969}$$
$$(2^{73.519}, 223) - psg(14, 1) - (2^{73.519}, 209)$$
$$(2^{73.519}, 209) - cc(209, 72) - (2^{73.519}, 72)\ comp = 2^{80.517}$$
$$WHT(72)\ comp = 2^{83.496}$$
$$total\ comp = 2^{89.46}$$

LF4
(512,1/8)-LPN

$$(2^{63.526}, 512) - ss(10) - (2^{53.526}, 502)\ comp = 2^{64.525}$$
$$(2^{53.526}, 502) - ge - (2^{53.526}, 502)\ comp = 2^{66.734}$$
$$(2^{53.526}, 502) - cp(156) - (2^{53.519}, 346)\ comp = 2^{115.024}$$
$$(2^{53.519}, 346) - cp(156) - (2^{53.490}, 190)\ comp = 2^{114.473}$$
$$(2^{53.490}, 190) - psg(16, 1) - (2^{53.490}, 174)$$
$$(2^{53.490}, 174) - cc(174, 60) - (2^{53.490}, 60)\ comp = 2^{60.934}$$
$$WHT(60)\ comp = 2^{70.675}$$
$$total\ comp = 2^{117.14}$$

(532,1/8)-LPN

$$\begin{aligned}
(2^{70.504}, 532) - ss(15) - (2^{55.504}, 517)\ comp &= 2^{71.504}\\
(2^{55.504}, 517) - ge - (2^{55.504}, 517)\ comp &= 2^{68.792}\\
(2^{55.504}, 517) - cp(162) - (2^{55.433}, 355)\ comp &= 2^{119.022}\\
(2^{55.433}, 355) - cp(162) - (2^{55.149}, 193)\ comp &= 2^{118.334}\\
(2^{55.149}, 193) - psg(13, 1) - (2^{55.149}, 180)&\\
(2^{55.149}, 180) - cc(180, 61) - (2^{55.149}, 61)\ comp &= 2^{62.631}\\
WHT(61)\ comp &= 2^{71.743}\\
total\ comp &= 2^{120.71}
\end{aligned}$$

(592,1/8)-LPN

$$\begin{aligned}
(2^{78.513}, 592) - ss(18) - (2^{60.513}, 574)\ comp &= 2^{79.513}\\
(2^{60.513}, 574) - ge - (2^{60.513}, 574)\ comp &= 2^{73.677}\\
(2^{60.513}, 574) - cp(177) - (2^{60.468}, 397)\ comp &= 2^{129.191}\\
(2^{60.468}, 397) - cp(177) - (2^{60.288}, 220)\ comp &= 2^{128.567}\\
(2^{60.288}, 220) - psg(16, 1) - (2^{60.288}, 204)&\\
(2^{60.288}, 204) - cc(204, 68) - (2^{60.288}, 60)\ comp &= 2^{67.956}\\
WHT(60)\ comp &= 2^{79.025}\\
total\ comp &= 2^{131.28}
\end{aligned}$$

Thus, the computations with the correct LF(4) give worse results than LF1 and LF2. This is because an LF(4) with the Wagner algorithm means 2 LF2 consecutive reductions and the LF(4) with the information theory approach brings no improvement. Also, the results of LF1 and LF2 don't bring an improvement over the results from [5].

To conclude, we believe that the results from [9] need to be updated and corrected and that the LF(4) operation is misleading. While the authors presented good ideas in how to improve the LPN solving algorithm, some of their methods (LF(4) and the pre-computation for Gaussian elimination) prove to be wrong.

References

1. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 435–440. ACM, 2000.
2. Sonia Bogos, Florian Tramèr, and Serge Vaudenay. On Solving LPN using BKW and Variants. *Cryptography and Communications*, 2015. (to appear).
3. Sonia Bogos and Serge Vaudenay. Optimization of LPN Solving Algorithms. Cryptology ePrint Archive, Report 2016/288, 2016. http://eprint.iacr.org/.
4. Marc P. C. Fossorier, Miodrag J. Mihaljevic, Hideki Imai, Yang Cui, and Kanta Matsuura. An Algorithm for Solving the LPN Problem and Its Application to Security Evaluation of the HB Protocols for RFID Authentication. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2006.

5. Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN Using Covering Codes. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2014.
6. Paul Kirchner. Improved Generalized Birthday Attack. *IACR Cryptology ePrint Archive*, 2011:377, 2011.
7. Éric Levieil and Pierre-Alain Fouque. An Improved LPN Algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006.
8. David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.
9. Bin Zhang, Lin Jiao, and Mingsheng Wang. Faster Algorithms for Solving LPN. volume 9665 of *Lecture Notes in Computer Science*, pages 168–195, 2016. http://eprint.iacr.org/.