CR10: Cryptanalysis Autumn 2018

Lecture III — 25/09, 2018

Lecturer G. Hanrot, E. Kirshanova *Scribe: L. Assouline*

The Learning Parity with Noise (LPN) Problem, The BKW Algorithm for LPN

1 The LPN Problem

Definition 1. *The LPN_n^τ problem asks to find the secret $s \in \mathbb{Z}_2^n$, given samples of the form $(a_i, b_i =< a_i, s > \oplus e_i) \in \mathbb{Z}_2^n \times \mathbb{Z}_2$, where $a_i \xleftarrow{\$} \mathbb{Z}_2^n$, e_i follows the Bernoulli distribution with parameter τ (i.e. $\Pr(e_i = 1) = \tau$), for $0 < \tau < \frac{1}{2}$. Such e_is are sometimes refered to as "noise", τ being the noise rate. We define the LPN bias $j = \frac{1}{2} - \tau$.*

Remark. 1. *The hardness of LPN is defined by two parameters: n and τ (we know τ). In particular:*

- $\tau = 0 \implies$ *Gaussian elimination*
- $\tau = \frac{1}{2} \implies$ *e_is act as a one-time pad. (s is information theoretically hidden)*
- *if $\tau > \frac{1}{2}$, $\forall i$ take $b_i \oplus 1$, you will get LPN samples with $\tau' = \tau - \frac{1}{2}$.*

2. *(s, e) is uniquely defined with high probability given $\mathcal{O}(n)$ LPN samples.*

3. *Applications of LPN:*
   - *identification schemes ([HB01])*
   - *collision-resistant hash functions*
   - *encryption ([Ale03])*
   - *light weight crypto*

2 Useful bounds

Mitzenmacher & Upfal, "Probability and Computing" [MU05]

Chernoff bound. *Let $e_1, \ldots, e_m$ be independant Bernoulli random variables s.t. $\forall i, \Pr(e_i = 1) = \tau$, let $N = \sum_{i=1}^{m} e_i$, $\mu = \mathbb{E}(N) = m\tau$. Then for $0 < \delta < 1$,*

$$\Pr(N \geq \mu(1+\delta)) \leq e^{\frac{-\mu\delta^2}{3}}$$

$$\Pr(N \leq \mu(1-\delta)) \leq e^{\frac{-\mu\delta^2}{3}}$$

For $x \in \{0,1\}^n$, denote the Hamming weight of x as $wt(x) = \#\{i | x_i = 1\}$.

Fact 2. *For $x \in \{0,1\}^n$ whose entries are iid from Ber_τ, $wt(x)$ follows the Binomial distribution $Bin_{n,\tau}$.*

Corollary 3. *Let e be the error vector coming from m-many LPN_n^τ samples. Then it holds that, for any $\delta \in]0,1[$, $\Pr(wt(e) > m\tau(1+\delta)) \leq e^{\frac{-m\tau\delta^2}{3}}$.*
In particular, for all $\epsilon > 0$, set $\delta = m^{\frac{-\epsilon}{2}}$, $\Pr(wt(e) > \tau(m + m^{\frac{-\epsilon}{2}})) \leq e^{\frac{-\tau m^{1-\epsilon}}{3}}$

Corollary 4 (Majority vote)**.** *Fix a bit $x \in \{0,1\}$. Let $b_i = x \oplus e_i$ where $e_i \leftarrow Ber_\tau$. Denote the bias by $\eta = \frac{1}{2} - \tau$. Then we can decide x with probability at least $1 - e^{-c}$ (where c is a constant), having $m \geq \frac{3}{2}\frac{c(1-2\eta)}{\eta^2}$ many b_is.*

Proof. $x' = \lfloor \frac{1}{m} \sum\limits_{i=1}^{m} b_i \rfloor = \lfloor \frac{1}{m} \sum\limits_{i=1}^{m} x + e_i \rfloor = x + \lfloor \frac{1}{m} \sum\limits_{i=1}^{m} e_i \rfloor$.
If $\frac{1}{m} \sum\limits_{i=1}^{m} e_i < \frac{1}{2}$ then $x' = x$.
Use Chernoff bound with $\delta = \frac{2\eta}{1-2\eta}$, then $\Pr(\frac{1}{m} wt(e) > \frac{1}{2}) < e^{-c}$ if $m \geq \frac{3}{2}\frac{c(1-2\eta)}{\eta^2}$ □

3 Algorithms for LPN_n^τ

Assume:

- we have as many samples as we need
- the secret is unique

3.1 Brute-Force

3.1.1 Over s

Algorithm 1 **Input:** m-many LPN_n^τ samples
Output: s

```
for all s ∈ {0,1}^n do
    if Test(s) then
        return s
    end if
end for
```

with Test:

Test

Input: m-many LPN_n^τ samples $(a_i, b_i)_{i \leq n}$
s: candidate for the solution
$\epsilon > 0$

Output: Reject / Accept

$N = 0$
for $i \in [m]$ **do**
  $N+ = b_i \oplus < a_i, s >$
end for
if $N > \tau(m + \frac{1}{m^{\frac{\epsilon}{2}}})$ **then**
  Reject
else
  Accept
end if

Claim 5. *Test accepts the correct s with probability $> 1 - e^{\frac{-\tau m^{1-\epsilon}}{3}}$, Test rejects a wrong s with probability $> 1 - e^{\frac{-m}{3}(1-\tau)^2}$*

Proof. The first part of the claim follows from corollary 1. With $s \neq s^*, s[1] = s^*[1]$,
$b_i \oplus < a_i, s >=< a_i, s \oplus s^* > +e_i = a_i[1] + e_i$.
$\Pr([a_i[1] + e_i] = 1) = \Pr(a_i[1] = 1). \Pr(e_i = 0) + \Pr(a_i[1] = 0). \Pr(e_i = 1) = \frac{1}{2}(1 - \tau) + \frac{1}{2}\tau = \frac{1}{2}$.
In this case, $N \sim Bin_{m,\frac{1}{2}}$. Apply Chernoff bound with $\tau = \frac{1}{2}$ and $\delta = 1 - 2\tau(1 + \frac{1}{m^{\frac{\epsilon}{2}+1}})$ □

3.1.2 Over e

Algorithm 2

for all integers $t \leq \tau(n + \frac{1}{n^{\frac{\epsilon}{2}}})$ **do**
  for all $e' \in \{0, 1\}^n$ s.t. $wt(e') = t$ **do**
    Solve $A.x = b - e'$ for x // require $A \in \mathcal{GL}_n(\mathbb{F}_2)$
    if $Test(x)$ **then**
      return x (or e')
    end if
  end for
end for

Theorem 6. *Algorithm 1 solves the LPN_n^τ problem with high probability in time $T(\text{brute force}) = \widetilde{\mathcal{O}}(2^n)$ using $m = \mathcal{O}(n)$ samples and $\mathcal{O}(n)$ memory.*
Algorithm 2 solves the LPN_n^τ problem in time $T = \widetilde{\mathcal{O}}(2^{\binom{n}{\tau n}}) = \widetilde{\mathcal{O}}(2^{n.H(\tau)})$ using $\mathcal{O}(n)$ memory samples.

3.2 "Many samples algorithm"

Algorithm 3 Goal: determine s_1

Repeat sampling until you see (a_i, b_i) where $a_i = (1, 0, \ldots, 0) = \vec{e}_1$ // $b_i =< a_i, s > +e_i = s_1 + e_i$

Repeat until we found m-many such samples

$\implies$ we can deduce s_1 if $m \geq \Theta(\frac{1}{(\frac{1}{2}-\tau)^2})$ //cf. Corollary 4 (Majority Vote)

This algorithm requires $m\frac{1}{\Pr(a_i = e_i)}$ many repetitions, i.e. $T = m2^n = \#\text{samples}$.
Try Gaussian elimination to obtain $\vec{e_i}$.
Problem: using naive Gaussian elimination, the vector $\vec{e_i}$ appears as a sum of $\mathcal{O}(n^2)$ a_is.

Lemma 7 (Piling-up lemma)**.** *Let e_i be iid Bernoulli random variables s.t.* $\Pr(e_i = 1) = \tau \ \forall i \in [m]$. *Then* $\sum_{i=1}^{m} e_i \sim Ber_{\frac{1}{2}-\frac{1}{2}(1-2\tau)^m}$

Proof by induction.

1. $m = 1 \to$ OK
2. Assume it holds for $m - 1$
3. $$\begin{aligned}\Pr(\sum_{i=1}^{m} e_i = 1) &= \Pr(\sum_{i=1}^{m-1} e_i = 1).\Pr(e_m = 0) + \Pr(\sum_{i=1}^{m-1} e_i = 0).\Pr(e_m = 1)\\ &= (1-\tau)(\frac{1}{2} - \frac{1}{2}(1-2\tau)^{m-1}) + \tau(\frac{1}{2} + \frac{1}{2}(1-2\tau)^{m-1})\\ &= \frac{1}{2} - \frac{1}{2}(1-2\tau)^m\end{aligned}$$

□

Remark. *In the naive Gaussian elimination, we sum up $\mathcal{O}(n^2)$ LPN_n^τ samples, with bias $\eta = \frac{1}{2} - \tau$, which results in LPN samples with bias $\eta' = \frac{1}{2}(1-2\tau)^{n^2} = c^{-n^2}$. One would need $m \geq c^{n^2}$ many such samples to decide on S_1*

3.3 The [BKW00] algorithm

The algorithm consists of two parts:

1. Block Gaussian elimination (aim: produce $e_1 = \sum_{i \in I} a_i$, $|I| = n^{1-\epsilon}$)
2. Majority vote

Algorithm 4: BKW part 1

Input: a list $\mathcal{L}^{(0)} = \mathcal{L} = \{(a_i, b_i) \text{ LPN samples}\}$ of size m
$k \in \mathbb{Z}$ block size (assume $k|n$)

Output: $I \subseteq [m]$ s.t. $|I| = 2^{\frac{n}{k}}$ and $\sum_{i \in I} a_i = \vec{e_1}$

```
for i ∈ [0, n/k − 2] do
    L^(i+1) ← {}
    Sort L^(i) w.r.t last k nonzero coordinates
    for all j ∈ {0,1}^k do
        choose an element a_j ∈ L^(i) whose last k nonzero coordinates are equal to j
        L^(i+1) = L^(i+1) ∪ {(a_j + a'_j, b_j + b'_j) for all a'_j ≠ a_j whose last nonzero coordinates are = j}
    end for
end for
Sort L^(n/k − 2) w.r.t the last k − 1 coordinates.
Find a pair of elements (x_1, x_2) from L^(n/k − 2) s.t. x_1[1] + x_2[1] = e_1
```

Claim 8. *Taking $m = |\mathcal{L}| = \Theta(poly(n) \cdot 2^k)$ suffices for Algorithm 4 to output $\mathcal{L}^{(\frac{n}{k}-1)}$ of expected size 1 in time $T(\text{Algorithm 4}) = \widetilde{\mathcal{O}}(2^k)$ using memory $M(\text{Algorithm 4}) = \widetilde{\mathcal{O}}(2^k)$.*

Proof sketch. On each step i, we partition $\mathcal{L}^{(i)}$ into at most 2^k classes represented by j.
For each non-empty class (i.e. $\exists$ at least two a_j, a'_j in $\mathcal{L}^{(i)}$), we discard only one of its representatives.
$\implies |\mathcal{L}^{(i+1)}| > |\mathcal{L}^{(i)}| - 2^k$ provided all (almost all) classes are non-empty. We need $\Omega(poly(k) \cdot 2^k)$ elements in $\mathcal{L}^{(i)}$ for $(1 - e^{-n})$-fraction of all classes to contain at least two elements.
$\implies$ we need to start with $|\mathcal{L}^{(0)}| = \Omega(\frac{n}{k} poly(k) \cdot 2^k) = \Omega(poly(n) \cdot 2^k)$ elements.
Elements in $\mathcal{L}^{(i)}$ are uniformly random, conditioned on having zeros on the last $(i-k)$ coordinates. Indeed, let us denote $Y = |\mathcal{L}^{(i)}|$ the random variable giving the size of $\mathcal{L}^{(i)}$. Cf Markov, $\Pr(Y > a) \leq \frac{\mathbb{E}(Y)}{a}$. Taking $a = \mathbb{E}(Y) \cdot poly(n)$ gives us a $1 - \frac{1}{poly(n)}$ probability on the list bound, and hence a $\frac{1}{poly(n)}$ probability on the success of the algorithm. (We could take $a = \mathbb{E}(Y) \cdot 2^{\epsilon n}$ to have an overwhelming probability. In that case, we need to replace $\widetilde{\mathcal{O}}(2^k)$ in the claim by $2^{\mathcal{O}(k)}$. This will however not change the next theorem). See [DRX17] for more details.
The most expensive part of the algorithm is sorting, it takes time $\widetilde{\mathcal{O}}(|\mathcal{L}^{(i)}|)$. □

Algorithm 5: The BKW algorithm
Input: m-many LPN_n^τ samples, $\epsilon > 0$
Output: $s \in \{0,1\}^n$

```
for i ∈ [n] do
    Run Algorithm 4 with k = n/((1−ε)log(n)), N = Θ((1 − 2τ)^(n^(1−ε))) times to obtain (e_i, b_j)_{j≤N}.
    Run the majority vote algorithm to decide on s_i.
end for
```

Theorem 9. *The BKW algorithm 5 solves the LPN_n^τ problem in time $T(BKW) = 2^{\mathcal{O}(\frac{n}{\log(n)})}$ using $M(BKW) = 2^{\mathcal{O}(\frac{n}{\log(n)})}$ memory and LPN samples.*

Proof. Algorithm 4 will output an $LPN_n^{\tau'}$ sample of the form (e_i, b_i) in time $\widetilde{\mathcal{O}}(2^{\frac{n}{(1-\epsilon)\log(n)}})$, with $\tau' = \frac{1}{2} - \frac{1}{2}(1-2\tau)^{2^{(1-\epsilon)\log(n)}} = \frac{1}{2} - \frac{1}{2}C^{n^{1-\epsilon}}$ with C a constant depending on τ.
For the majority vote, we would need to repeat Algorithm 4 $\mathcal{O}(2^{n^{1-\epsilon}})$ times to decide on s_i with constant success probability. □

Remark. *1. A more precise complexity of BKW is:*

$$T = M = \#samples = 2^{\frac{n}{\log(\frac{n}{\tau})}(1+o(1))} \text{ cf [EKM17]}$$

2. *Given $m = n^{1+\epsilon}$ many LPN_n^τ samples, the BKW algorithm's run-time goes up to $T(BKW) = 2^{bigO(\frac{n}{loglog(n)})}$*
(sample "amplification", cf V. Lyubashevsky in [Lyu05])

3. *The "LF2" technique due to Leviel-Fouque [LF06] analyses the BKW algorithm by considering all possible pairs during the zeroizing step.*
Proved by Devadas et al. [DRX17].

Ring-LPN

Take the ring $\mathbb{F}_2[X]/(f)$, with f a degree n polynomial.
Ring-LPN sample $s \in \mathbb{F}_2[X]/(f)$. $(a_i \xleftarrow{\$} \mathbb{F}_2[X]/(f),\ a_i.s_i + e_i \in \mathbb{F}_2[X]/(f))$.
e_i: polynomial with coefficients chosen from Ber_τ.
Open problem: find a better algorithm for Ring-LPN than as for "standard" LPN.

References

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 298–, Washington, DC, USA, 2003. IEEE Computer Society.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *CoRR*, cs.LG/0010022, 2000.

[DRX17] Srinivas Devadas, Ling Ren, and Hanshen Xiao. On iterative collision search for lpn and subset sum. Cryptology ePrint Archive, Report 2017/904, 2017. `https://eprint.iacr.org/2017/904`.

[EKM17] Andre Esser, Robert Kbler, and Alexander May. Lpn decoded. Cryptology ePrint Archive, Report 2017/078, 2017. `https://eprint.iacr.org/2017/078`.

[HB01] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 52–66, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[LF06] Éric Levieil and Pierre-Alain Fouque. An improved lpn algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks*, pages 348–359, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randamization and Computation: Algorithms and Techniques*, APPROX'05/RANDOM'05, pages 378–389, Berlin, Heidelberg, 2005. Springer-Verlag.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, Cambridge, 2005.