

# How to Sequentialize Independent Parallel Attacks? Biased Distributions Have a Phase Transition

Sonia Bogos<sup>(✉)</sup> and Serge Vaudenay

EPFL, 1015 Lausanne, Switzerland  
{soniamihaela.bogos, serge.vaudenay}@epfl.ch

**Abstract.** We assume a scenario where an attacker can mount several independent attacks on a single CPU. Each attack can be run several times in independent ways. Each attack can succeed after a given number of steps with some given and known probability. A natural question is to wonder what is the optimal strategy to run steps of the attacks in a sequence. In this paper, we develop a formalism to tackle this problem. When the number of attacks is infinite, we show that there is a magic number of steps  $m$  such that the optimal strategy is to run an attack for  $m$  steps and to try again with another attack until one succeeds. We also study the case of a finite number of attacks.

We describe this problem when the attacks are exhaustive key searches, but the result is more general. We apply our result to the learning parity with noise (LPN) problem and the password search problem. Although the optimal  $m$  decreases as the distribution is more biased, we observe a phase transition in all cases: the decrease is very abrupt from  $m$  corresponding to exhaustive search on a single target to  $m = 1$  corresponding to running a single step of the attack on each target. For all practical biased examples, we show that the best strategy is to use  $m = 1$ . For LPN, this means to guess that the noise vector is 0 and to solve the secret by Gaussian elimination. This is actually better than all variants of the Blum-Kalai-Wasserman (BKW) algorithm.

## 1 Introduction

We assume that there are an infinite number of independent keys  $K_1, K_2, \dots$  and that we want to find at least one of these keys by trials with minimal complexity. Each key search can be stopped and resumed. The problem is to find the optimal strategy to run several partial key searches in a sequence. In this optimization problem, we assume that the distributions  $D_i$  for each  $K_i$  are known. We denote  $D = (D_1, D_2, \dots)$ . Consider the problem of guessing a key  $K_i$ , drawn following  $D_i$ , which is not necessarily uniform. We assume that we try all key values exhaustively from the first to the last following a fixed ordering. If we stop the

---

S. Bogos—Supported by a grant of the Swiss National Science Foundation, 200021\_143899/1.

key search on  $K_i$  after  $m$  trials, the sequence of trials is denoted by  $ii \cdots i = i^m$ . It has a worst-case complexity  $m$  and a probability of success which we denote by  $\Pr_D(i^m)$ .

Instead of running parallel key searches in sequence, we could consider any other attack which decomposes in *steps* of the same complexity and in which each step has a specific probability to be the succeeding one. We assume that the  $i$ th attack has a probability  $\Pr_D(i^m)$  to succeed within  $m$  steps and that each step has a complexity 1. The fundamental problem is to wonder how to run steps of these attacks in a sequence so that we minimize the complexity until one attack succeeds. For instance, we could run attack 1 for up to  $m$  steps and decide to give up and try again with attack 2 if it fails for attack 1, and so on. We denote by  $s = 1^m 2^m 3^m \cdots$  this strategy. Unsurprisingly, when the  $D_i$ 's are the same, the average complexity of  $s$  is the ratio  $\frac{C_D(1^m)}{\Pr_D(1^m)}$  where  $C_D(1^m)$  is the expected complexity of the strategy  $1^m$  which only runs attack 1 for  $m$  steps<sup>1</sup> and  $\Pr_D(1^m)$  is its probability of success.

Traditionally, when we want to compare single-target attacks with different complexity  $C$  and probability of success  $p$ , we use as a rule of the thumb to compare the ratio  $\frac{C}{p}$ . Quite often, we have a continuum of attacks  $C(m)$  with a number of steps limited to a variable  $m$  and we tune  $m$  so that  $p(m)$  is a constant such as  $\frac{1}{2}$ . Indeed, the curve of  $m \mapsto \frac{C(m)}{p(m)}$  is often decreasing (so has an L shape) or decreasing then increasing (with a U shape) and it is optimal to target  $p(m) = \frac{1}{2}$ . But sometimes, the curve can be increasing with a  $\Gamma$  shape. In this case, it is better to run an attack with very low probability of success and to try again until this succeeds. In some papers, e.g. [14], we consider  $\min \frac{C(m)}{p(m)}$  as a complexity metric to compare attacks. Our framework justifies this choice.

LPN and Learning with Errors (LWE) [21] are two appealing problems in cryptography. In both cases, the adversary receives a matrix  $V$  and a vector  $C = Vs + D$  where  $s$  is a secret vector and  $D$  is a noise vector. For LPN, the best solving algorithm was presented in Asiacrypt 2014 [12]. It brings an improvement over the well-known BKW [5] and its variants [11, 15]. The best algorithm has a sub-exponential complexity.

Assuming that  $V$  is invertible, by guessing  $D$  we can solve  $s$  and check it with extra equations. So, this problem can be expressed as the one of guessing a correct vector  $D$  of small weight, which defines a biased distribution. Here, the distribution of  $D$  corresponds to the weighted concatenation of uniform distributions among vectors of the same weight. We can thus study this problem in our formalism. This was used in [8]. This algorithm is also cited in [6] and by Lyubashevsky<sup>2</sup>.

Both LPN and LWE fall in the aforementioned scenario of guessing a  $k$ -bit biased noise vector by a simple transformation. Work on breaking cryptosystems with biased keys was also done in [18].

<sup>1</sup>  $C_D(1^m)$  can be lower than  $m$  since there is a probability to succeed before reaching the  $m$ th step.

<sup>2</sup> <http://www.di.ens.fr/~lyubash/talks/LPN.pdf>.

The guessing game that we describe in our paper also matches well the password guessing scenario where an attacker tries to gain access to a system by hacking an account of an employee. There exists an extensive work on the crypt-analytic time-memory tradeoffs for password guessing [2–4, 13, 19, 20], but the game we analyse here requires no pre-computation done by the attacker.

*Our Results.* We develop a formalism to compare strategies and derive some useful lemmas. We show that when we can run an infinite number of independent attacks of the same distribution, an optimal strategy is of the form  $1^m 2^m 3^m \dots$  and it has complexity

$$\min_m \frac{C_D(1^m)}{\Pr_D(1^m)}$$

for some “magic” value  $m$ . This justifies the rule of the thumb to compare attacks with different probabilities of success.

When the probability that an attack succeeds at each new step decreases (e.g., because we try possible key values in decreasing order of likelihood), there are two remarkable extreme cases:  $m = n$  (where  $n$  is the maximal number of steps) corresponds to the normal single-target exhaustive search with a complexity equal to the *guesswork entropy* [17] of the distribution;  $m = 1$  corresponds to trying attacks for a single step until it works, with complexity  $2^{-H_\infty}$ , where  $H_\infty$  is the *min-entropy* of the distribution.

When looking at the “magic” value  $m$  in terms of the distribution  $D$ , we observe that in many cases there is a phase transition: when  $D$  is very close to uniform, we have  $m = n$ . As soon as it becomes slightly biased, we have  $m = 1$ . There is no graceful decrease from  $m = n$  to  $m = 1$ .

We also treat the case where we have a finite number  $|D|$  of independent attacks to run. We show that there is an optimal “magic” sequence  $m_1, m_2, \dots$  such that an optimal strategy has form

$$1^{m_1} 2^{m_1} \dots |D|^{m_1} 1^{m_2} 2^{m_2} \dots |D|^{m_2} \dots$$

The best strategy is first to run all attacks for  $m_1$  steps in a sequence then to continue to run them for  $m_2$  steps in a sequence, and so on.

Although our results look pretty natural, we show that there are distributions making the analysis counter-intuitive. Proving these results is actually non trivial.

We apply this formalism to LPN by guessing the noise vector then performing a Gaussian elimination to extract the secret. The optimal  $m$  decreases as the probability  $\tau$  to have an error in a parity bit decreases from  $\frac{1}{2}$ . For  $\tau = \frac{1}{2}$ , the optimal  $m$  corresponds to a normal exhaustive search. For  $\tau < \frac{1}{2} - \frac{\ln 2}{2k}$ , where  $k$  is the length of the secret, the optimal  $m$  is 1: this corresponds to guessing that we have no noise at all. So, there is a phase transition.

Furthermore, for LPN with  $\tau = k^{-\frac{1}{2}}$ , which is what is used in many cryptographic constructions, the obtained complexity is  $\text{poly} \cdot e^{\sqrt{k}}$  which is much better than the usual  $\text{poly} \cdot 2^{\frac{k}{\log_2 k}}$  that we obtain for variants of the BKW algorithm [6].

More generally, we obtain a complexity of  $\text{poly} \cdot e^{-k \ln(1-\tau)}$ . It is not better than the BKW variants for constant  $\tau$  but becomes interesting when  $\tau < \frac{\ln 2}{\log_2 k}$ .

When the number of samples is limited in the LPN problem with  $\tau = k^{-\frac{1}{2}}$ , we can still solve it with complexity  $e^{\mathcal{O}(\sqrt{k}(\ln k)^2)}$  which is better than  $e^{\mathcal{O}(\frac{k}{\ln k})}$  with the BKW variants [16].

For LWE, the phase transition is similar, but the algorithm for  $m = 1$  is not better than the BKW variants. This is due to the 0 noise having a much lower probability in LWE (which is  $1 - \tau$  for LPN) in the discrete Gaussian distribution in  $\mathbb{Z}_q$ .

For password search, we tried several empirical distributions of passwords and obtained again that the optimal  $m$  is  $m = 1$ . So, the complexity is  $2^{-H_\infty}$ .

Besides the 3 problems we study here, we believe that our results can prove to be useful in other cryptographic applications.

*Structure of the Paper.* Section 2 formalizes the problem and presents a few useful results. In Sect. 3 we characterize the optimal strategies and show they can be given a special regular structure. We then apply this in Sect. 4 with LPN and password recovery. Due to lack of space, we do the same for LWE in the full version of this paper. We study the phase transition of the “magic” number  $m$  in Sect. 5 and conclude in Sect. 6.

## 2 The STEP Game

In this section we introduce our framework through which we address the fundamental question of what is the best strategy to succeed in at least one attack when we can step several independent attacks. Let  $D = (D_1, D_2, \dots)$  be a tuple of independent distributions. If it is finite,  $|D|$  denotes the number of distributions. We formalize our framework as a game where we have a ppt adversary  $\mathcal{A}$  and an oracle that has a sequence of keys  $(K_1, K_2, \dots)$  where  $K_i \leftarrow D_i$ . At the beginning, the oracle assigns the keys according to their distribution. These distributions are known to the adversary  $\mathcal{A}$ . The adversary will test each key  $K_i$  by exhaustive search following a given ordering of possible values. We can assume that values are sorted by decreasing order of likelihood to obtain a minimal complexity but this is not necessary in our analysis. We only assume a fixed order. So, our framework generalizes to other types of attacks in which we cannot choose the order of the steps. Each test on  $K_i$  corresponds to a step in the exhaustive search for  $K_i$ . In general, we write “ $i$ ” in a sequence to denote that we run one new step of the  $i$ th attack. The sequence of “ $i$ ”s defines a strategy  $s$ . It can be finite or not. The sequence of steps we follow is thus a sequence of indices. For instance,  $i^m$  means “run the  $K_i$  search for  $m$  steps”. The oracle is an algorithm that has a special command:  $\text{STEP}(i)$ . When queried with the command  $\text{STEP}(i)$ , the oracle runs one more step of the  $i^{\text{th}}$  attack ( so, it increments a counter  $t_i$  and tests if  $K_i = t_i$ , assuming that possible key values are numbered from 1). If this happens then the adversary wins. The adversary wins as soon as one attack succeeds (i.e., he guesses one of the keys from the sequence  $K_1, K_2, \dots$  ).

**Definition 1 (Strategies).** Let  $D$  be a sequence of distributions  $D = (D_1, \dots, D_{|D|})$  (where  $|D|$  can be infinite or not). A strategy for  $D$  is a sequence  $s$  of indices between 1 and  $|D|$ . It corresponds to Algorithm 1. We let  $\Pr_D(s)$  be the probability that the strategy succeeds and  $C_D(s)$  be the expected number of STEP when running the algorithm until it stops. We say that the strategy is full if  $\Pr_D(s) = 1$  and that it is partial otherwise.

---

**Algorithm 1.** Strategy  $s$  in the STEP game

---

- 1: initialize attacks  $1, \dots, |D|$
  - 2: **for**  $j = 1$  to  $|s|$  **do**
  - 3:     STEP( $s_j$ ): run one more step of the attack  $s_j$  and stop if succeeded
  - 4: **end for**
  - 5: stop (the algorithm fails)
- 

For example for  $s = 11223344 \dots$ , Algorithm 1 tests the first two values for each key.

**Definition 2 (Distributions).** A distribution  $D_i$  over a set of size  $n$  is a sequence of probabilities  $D_i = (p_1, \dots, p_n)$  of sum 1 such that  $p_j \geq 0$  for  $j = 1, \dots, n$ . We assume without loss of generality that  $p_n \neq 0$  (Otherwise, we decrease  $n$ ). We can equivalently specify the distribution  $D_i$  in an incremental way by a sequence  $D_i = [p'_1, \dots, p'_n]$  (denoted with square brackets) such that

$$p'_j = \frac{p_j}{p_j + \dots + p_n} \qquad p_j = p'_j(1 - p'_1) \dots (1 - p'_{j-1})$$

for  $j = 1, \dots, n$ .

We have  $\Pr_D(i^j) = p_1 + \dots + p_j = 1 - (1 - p'_1) \dots (1 - p'_j)$ , the probability of the  $j$  first values under  $D_i$ .

When considering the key search, it may be useful to assume that distributions are sorted by decreasing likelihood. We note that the equivalent condition to  $p_j \geq p_{j+1}$  with the incremental description is  $\frac{1}{p'_j} + j \leq \frac{1}{p'_{j+1}} + j + 1$ , for  $j = 1, \dots, n - 1$ .

We define the distribution that the keys are not among the already tested ones.

**Definition 3 (Residual Distribution).** Let  $D = (D_1, \dots, D_{|D|})$  be a sequence of distributions and let  $s$  be a strictly partial strategy for  $D$  (i.e.,  $\Pr_D(s) < 1$ ). We denote by “ $\neg s$ ” the residual distribution in the case where the strategy  $s$  does not succeed, i.e., the event  $\neg s$  occurs.

We let  $\#occ_s(i)$  denote the number of occurrences of  $i$  in  $s$ . We have

$$D|\neg s = \left( D_1|\neg 1^{\#occ_s(1)}, \dots, D_{|D|}|\neg |D|^{\#occ_s(|D|)} \right)$$

where  $D_i | \neg i^{t_i} = [p'_{i,t_i+1}, \dots, p'_{i,n_i}]$  if  $D_i = [p'_{i,1}, \dots, p'_{i,n_i}]$ . Hence, defining distributions in the incremental way makes the residual distribution being just a shift of the original one.

We write  $\Pr_D(s' | \neg s) = \Pr_{D | \neg s}(s')$  and  $C_D(s' | \neg s) = C_{D | \neg s}(s')$ .

Next, we prove a list of useful lemmas in order to compute complexities, compare strategies, etc.

**Lemma 4 (Success Probability).** *Let  $s$  be a strategy for  $D$ . The success probability is computed by*

$$\Pr_D(s) = 1 - \prod_{i=1}^{|D|} \Pr_{D_i}(\neg i^{\#\text{occ}_s(i)})$$

*Proof.* The failure corresponds to the case where for all  $i$ ,  $K_i$  is not in  $\{1, \dots, \#\text{occ}_s(i)\}$ . The independence of the  $K_i$  implies the result.  $\square$

**Lemma 5 (Complexity of Concatenated Strategies).** *Let  $ss'$  be a strategy for  $D$  obtained by concatenating the sequences  $s$  and  $s'$ . If  $\Pr_D(s) = 1$ , we have  $\Pr_D(ss') = \Pr_D(s)$  and  $C_D(ss') = C_D(s)$ . Otherwise, we have*

$$\begin{aligned} \Pr_D(ss') &= \Pr_D(s) + \left(1 - \Pr_D(s)\right) \Pr_D(s' | \neg s) \\ C_D(ss') &= C_D(s) + \left(1 - \Pr_D(s)\right) C_D(s' | \neg s) \end{aligned}$$

*Proof.* The first equation is trivial from the definition of residual distributions and conditional probabilities.

The prefix strategy  $s$  succeeds with probability  $\Pr_D(s)$ . Let  $c$  be the complexity of  $s$  conditioned to the event that  $s$  succeeds. Clearly, the complexity of  $ss'$  conditioned to this event is equal to  $c$ . The complexity of  $ss'$  conditioned to the opposite event is equal to  $|s| + C_D(s' | \neg s)$ . So,  $C_D(ss') = \Pr_D(s)c + (1 - \Pr_D(s))(|s| + C_D(s' | \neg s))$ . The complexity of  $s$  conditioned to that  $s$  fails is equal to  $|s|$ . So,  $C_D(s) = \Pr_D(s)c + (1 - \Pr_D(s))|s|$ . From these two equations, we obtain the result.  $\square$

**Lemma 6 (Complexity with Incremental Distributions).** *Let  $D_i = [p'_{i,1}, \dots, p'_{i,n_i}]$  and let  $s$  be a strategy for  $D = (D_1, D_2, \dots)$ . We have*

$$\begin{aligned} \Pr_D(s) &= 1 - \prod_{t'=1}^{|s|} (1 - p'_{s_{t'}, \#\text{occ}_{s_1 \dots s_{t'}}(s_{t'})}) \\ C_D(s) &= \sum_{t=1}^{|s|} \prod_{t'=1}^{t-1} (1 - p'_{s_{t'}, \#\text{occ}_{s_1 \dots s_{t'}}(s_{t'})}) \end{aligned}$$

*Proof.* By induction, the probability that the strategy fails on the first  $t-1$  steps is  $q_t = \prod_{t'=1}^{t-1} (1 - p'_{s_{t'}, \#\text{occ}_{s_1 \dots s_{t'}}(s_{t'})})$ . We can express  $C_D(s) = \sum_{t=1}^{|s|} q_t$ . So, we can deduce  $\Pr_D(s)$  and  $C_D(s)$ .  $\square$

*Example 7.* For  $D_1 = (p_1, \dots, p_n) = [p'_1, \dots, p'_n]$  and  $m \leq n$ , due to Lemma 6 we have

$$\Pr_D(1^m) = p_1 + \dots + p_m = 1 - (1 - p'_1) \cdots (1 - p'_m)$$

and

$$\begin{aligned} C_D(1^m) &= \sum_{t=1}^m \prod_{j=1}^{t-1} (1 - p'_j) \\ &= \sum_{t=1}^m (p_t + \dots + p_n) = p_1 + 2p_2 + \dots + mp_m + mp_{m+1} + \dots + mp_n \end{aligned}$$

The second equality uses the relations from Definition 2.

We want to concatenate an isomorphic copy  $w$  of a strategy  $v$  to another strategy  $u$ . For this, we make sure that  $w$  and  $u$  have no index in common.

**Definition 8 (Disjoint Copy of a Strategy).** *Two strategies  $v$  and  $w$  are isomorphic if there exists an injective mapping  $\varphi$  such that  $w_t = \varphi(v_t)$  for all  $t$  and  $D_{\varphi(i)} = D_i$  for all  $i$ . So,  $C_D(v) = C_D(w)$ . Let  $u$  and  $v$  be two strategies for  $D$ . Whenever possible, we define a new strategy  $w = \text{new}_u(v)$  such that  $v$  and  $w$  are isomorphic and  $w$  has no index in common with  $u$ .*

*We can define it by recursion: if  $w_1 = \varphi(v_1), \dots, w_{t-1} = \varphi(v_{t-1})$  are already defined and  $\varphi(v_t)$  is not, we set it to the smallest index  $i$  (if exists) which does not appear in  $u$  nor in  $w_1, \dots, w_{t-1}$  and such that  $D_i = D_{v_t}$ .*

For instance, if  $v = 1^m$ , all  $D_i$  are equal, and  $i$  is the minimal index which does not appear in  $u$ , we have  $\text{new}_u(v) = i^m$ .

**Lemma 9 (Complexity of a Repetition of Disjoint Copies).** *Let  $s$  be a non-empty strategy for  $D$ . We define new strategies  $s_{+1}, s_{+2}, \dots$ , disjoint copies of  $s$ , by recursion as follows:  $s_{+r} = \text{new}_{s_{+1} \dots s_{+(r-1)}}(s)$ . We assume that  $s_{+1}, s_{+2}, \dots, s_{+(r-1)}$  can be constructed. If  $\Pr_D(s) = 0$ , then*

$$C_D(ss_{+1}s_{+2} \cdots s_{+(r-1)}) = r \cdot C_D(s).$$

*Otherwise, we have*

$$C_D(ss_{+1}s_{+2} \cdots s_{+(r-1)}) = \frac{1 - (1 - \Pr_D(s))^r}{\Pr_D(s)} C_D(s)$$

*For  $r$  going to  $\infty$ , we respectively obtain  $C_D(ss_{+1}s_{+2} \cdots) = +\infty$  and*

$$C_D(ss_{+1}s_{+2} \cdots) = \frac{C_D(s)}{\Pr_D(s)}$$

For instance, for  $s = 1^m$  and  $D_i$  all equal, the disjoint isomorphic copies of  $s$  are  $s_{+r} = (1+r)^m$ . I.e., we run  $m$  steps the  $(1+r)$ th attack. So,  $ss_{+1}s_{+2} \cdots s_{+(r-1)} = 1^m 2^m \dots r^m$ .

*Proof.* We prove it by induction on  $r$ . This is trivial for  $r = 1$ . Let  $\bar{s}_r = s s_{+1} s_{+2} \cdots s_{+r}$ . If it is true for  $r - 2$ , then

$$C_D(\bar{s}_{r-1}) = C_D(\bar{s}_{r-2}) + (1 - \Pr(\bar{s}_{r-2}))C_D(s_{+(r-1)}|\neg\bar{s}_{r-2})$$

$$= \begin{cases} \frac{1 - (1 - \Pr_D(s))^{r-1}}{\Pr_D(s)} C_D(s) + (1 - \Pr_D(\bar{s}_{r-2}))C_D(s_{+(r-1)}|\neg\bar{s}_{r-2}) & \text{if } \Pr_S(s) > 0 \\ (r - 1) \cdot C_D(s) + (1 - \Pr_D(\bar{s}_{r-2}))C_D(s_{+(r-1)}|\neg\bar{s}_{r-2}) & \text{if } \Pr_S(s) = 0 \end{cases}$$

Clearly, we have  $1 - \Pr_D(\bar{s}_{r-2}) = (1 - \Pr_D(s))^{r-1}$  and  $C_D(s_{+(r-1)}|\neg\bar{s}_{r-2}) = C_D(s)$ . So, we obtain the result.  $\square$

*Example 10.* For all  $D_i$  equal, if we let  $s = 1^m$ , we can compute

$$C_D(1^m 2^m \cdots r^m) = \frac{1 - (1 - \Pr_D(1^m))^r}{\Pr_D(1^m)} C_D(1^m)$$

$$= \frac{1 - (p_{m+1} + \cdots + p_n)^r}{p_1 + \cdots + p_m} (p_1 + 2p_2 + \cdots + mp_m + mp_{m+1} + \cdots + mp_n)$$

We now consider  $r = \infty$ . For an infinite number of i.i.d distributions we have

$$C_D(1^m 2^m \cdots) = \frac{C_D(1^m)}{\Pr_D(1^m)}$$

$$= \frac{p_1 + 2p_2 + \cdots + mp_m + mp_{m+1} + \cdots, mp_n}{p_1 + \cdots + p_m}$$

$$= \frac{\sum_{i=1}^m ip_i + m(1 - p_1 + \cdots + p_m)}{p_1 + \cdots + p_m}$$

$$= G_m + m \left( \frac{1}{\Pr_{D_i}(1^m)} - 1 \right)$$

where  $G_m = C_{D_1|1^m}(1^m)$  and  $D_1|1^m = (\frac{p_1}{\Pr_{D_1}(1^m)}, \dots, \frac{p_m}{\Pr_{D_1}(1^m)})$ . If  $D_1$  is ordered,  $G_m$  corresponds to the guesswork entropy of the key with distribution  $D_1|1^m$ .

We can see two extreme cases for  $s = 1^m 2^m \cdots$ . On one end we have a strategy of exhaustively searching the key until it is found, i.e. take  $m = n$ . On the other extreme we have a strategy where the adversary tests just one key before switching to another key, i.e.  $m = 1$ . For the sequences  $s = 12 \cdots$  and  $s = 1^n 2^n \cdots$ , i.e.  $m = 1$  and  $m = n$ , when  $D_1$  is ordered by decreasing likelihood, we obtain the following expected complexity:

$$m = 1 \Rightarrow C_D(12 \cdots) = \frac{1}{p_1} = 2^{-H_\infty(D_1)}$$

$$m = n \Rightarrow C_D(1^n 2^n \cdots) = C_D(1^n) = G_n,$$

where  $H_\infty(D_1)$  and  $G_n$  denote the min-entropy and the guesswork entropy of the distribution  $D_1$ , respectively.

We now define a way to compare partial strategies.



**Definition 11 (Strategy Comparison).** *We define*

$$\min C_D(s) = \inf_{s'; \Pr_D(ss')=1} C_D(ss')$$

the infimum of  $C_D(ss')$ , i.e. the greatest of its lower bounds. We write  $s \leq_D s'$  if and only if  $\min C_D(s) \leq \min C_D(s')$ . A strategy  $s$  is optimal if  $\min C_D(s) = \min C_D(\emptyset)$ , where  $\emptyset$  is the empty strategy (i.e. the strategy running no step at all).

So,  $s$  is better than  $s'$  if we can reach lower complexities by starting with  $s$  instead of  $s'$ . The partial strategy  $s$  is optimal if we can still reach the optimal complexity when we start by  $s$ .

**Lemma 12 (Best Prefixes are Best Strategies).** *If  $u$  and  $v$  are permutations of each other, we have  $u \leq_D v$  if and only if  $C_D(u) \leq C_D(v)$ .*

*Proof.* Note that  $\Pr_D(u) = 1$  is equivalent to  $\Pr_D(v) = 1$ . If  $\Pr_D(u) = 1$ , it holds that  $\min C_D(u) = C_D(u)$  and  $\min C_D(v) = C_D(v)$ . So, the result is trivial in this case. Let us now assume that  $\Pr_D(u) < 1$  and  $\Pr_D(v) < 1$ . For any  $s'$ , by using Lemma 5 we have

$$C_D(us') = C_D(u) + \left(1 - \Pr_D(u)\right) C_D(s'|\neg u)$$

So,

$$\inf_{s'; \Pr_D(us')=1} C_D(us') = C_D(u) + \left(1 - \Pr_D(u)\right) \inf_{s'; \Pr_D(us')=1} C_D(s'|\neg u)$$

The same holds for  $v$ . Since  $u$  and  $v$  are permutations of each other, we have  $D|\neg u = D|\neg v$ . So,  $\Pr_D(us') = \Pr_D(vs')$  and  $C_D(s'|\neg u) = C_D(s'|\neg v)$ . Hence,  $\inf C_D(s'|\neg u) = \inf C_D(s'|\neg v)$ . Furthermore, we have  $\Pr_D(u) = \Pr_D(v)$ . So,  $\min C_D(u) \leq \min C_D(v)$  is equivalent to  $C_D(u) \leq C_D(v)$ .  $\square$

### 3 Optimal Strategy

The question we address in this paper is: what is the optimal strategy for the adversary so that he obtains the best complexity in our STEP formalism? That is, we try to find the optimal sequence  $s$  for Algorithm 1. At a first glance, we may think that a *greedy* strategy always making a step which is the most likely to succeed is an optimal strategy. We show below that this is wrong. Sometimes, it is better to run a series of unlikely steps in one given attack because we can then run a much more likely one of the same attack after these steps are complete. However, criteria to find this strategy are not trivial at all.

The greedy algorithm is based on looking at the  $i$  for which the next applicable  $p_j$  in  $D_i$  is the largest. With our formalism, this defines as follows.

**Definition 13 (Greedy Strategy).** *Let  $s$  be a strategy for  $D$ . We say that  $s$  is greedy if*

$$\Pr(s_t|\neg s_1 \cdots s_{t-1}) = \max_i \Pr(i|\neg s_1 \cdots s_{t-1})$$

for  $t = 1, \dots, |s|$ .

The following example shows that the greedy strategy is not always optimal.

*Example 14.* We take  $|D| = \infty$  and all  $D_i$  equal to  $D_i = (\frac{2}{3}, \frac{7}{36}, \frac{5}{36}) = [\frac{2}{3}, \frac{7}{12}, 1]$ . After testing the first key, we have  $D|\neg 1 = (D', D_2, D_3, \dots)$  with  $D' = (\frac{7}{12}, \frac{5}{12}) = [\frac{7}{12}, 1]$ . Since  $\frac{2}{3} > \frac{7}{12}$ , the greedy algorithm would then test a new key and continue testing new keys. I.e., we would have  $s = 1234 \dots$  as a greedy strategy. By applying Lemma 5, the complexity is solution to  $c = 1 + \frac{1}{3}c$ , i.e.,  $c = \frac{3}{2}$ . However, the one-key strategy  $s = 111$  has complexity

$$\frac{2}{3} + 2\frac{7}{36} + 3\frac{5}{36} = \frac{53}{36} < \frac{3}{2}$$

so the greedy strategy is not the best one.

*Remark:* The above counterexample works even when  $|D|$  is finite. If we take  $D = (D_1, D_2)$  with  $D_i = (\frac{2}{3}, \frac{7}{36}, \frac{5}{36}) = [\frac{2}{3}, \frac{7}{12}, 1]$ , the greedy approach would test the strategy  $s = 1211$  that has a complexity of

$$1 + \frac{1}{3} \left( 1 + \frac{1}{3} \left( 1 + \frac{5}{12} \cdot 1 \right) \right) = \frac{161}{108}$$

This is greater than  $\frac{53}{36}$ , the complexity of the strategy 111.

Next, we note that we may have no optimal strategy as the following example shows.

*Example 15 (Distribution with No Optimal Strategy).* Let  $q_i$  be an increasing sequence of probabilities which tends towards 1 without reaching it. Let  $D_i = [q_i, q_i, \dots, q_i, 1]$  of support  $n$ . We have  $C(i^n) = \frac{1}{q_i}(1 - (1 - q_i)^n)$  which tends towards 1 as  $i$  grows. So, 1 is the best lower bound of the complexity of full strategies. But there is no full strategy of complexity 1.

When the number of different distributions is finite, optimal strategies exist.

**Lemma 16 (Existence of an Optimal Full Strategy).** *Let  $D = (D_1, D_2, \dots)$  be a sequence of distributions. We assume that we have in  $D$  a finite number of different distributions. There exists a full strategy  $s$  such that  $C_D(s)$  is minimal.*

*Proof.* Clearly,  $c = \inf C_D(s)$  over all full strategies  $s$  is well defined. Essentially, we want to prove that  $c$  is reached by one strategy, i.e. that the infimum is a minimum. First, if  $c = \infty$ , all full strategies have infinite complexity, and the result is trivial. So, we now assume that  $c < +\infty$  and we prove the result by a diagonal argument.

We now construct  $s = s_1 s_2 \dots$  by recursion. We assume that  $s_1 s_2 \dots s_r$  is constructed such that  $\min C(s_1 s_2 \dots s_r) = c$ . We concatenate  $s_1, \dots, s_r$  to  $i^m$  where  $m$  is such that  $\Pr_D[i^{m-1} | \neg s_1 \dots s_r] = 0$  and  $\Pr_D[i^m | \neg s_1 \dots s_r] > 0$ . The values of  $i$  to try are the ones such that  $i$  appears in  $s_1, \dots, s_r$  (we have a finite

number of them), and the ones which do not appear, but we can try only one for each different  $D_i$ . We take the choice minimizing  $\min C(s_1 s_2 \cdots s_r i^m)$  and set  $s_{r+1} = i^m$ . So, we construct a strategy  $s$ .

If one key  $K_i$  is tested until exhaustion, we have  $\Pr_D(s) = 1$ . If no key is tested until exhaustion, there is an infinite number of keys with same distribution  $D_i$  which are tested. If  $p = \Pr_D[i^m]$  is the nonzero probability with the smallest  $m$  of this distribution, there is an infinite number of tests which succeed with probability  $p$ . So,  $\Pr_D(s) \geq 1 - (1 - p)^\infty = 1$ . In all cases, as  $s$  has a probability to succeed of 1,  $s$  is a full strategy.

What remains to be proven is that  $C_D(s) = c$ . We now denote by  $s_i$  the  $i$ th step of  $s$ .

Let  $q_t$  be the probability that  $s$  fails on the first  $t - 1$  steps. We have  $C_D(s) = \sum_{t=1}^{|s|} q_t$ . Let  $\varepsilon > 0$ . For each  $r$ , by construction, there exists a tail strategy  $v$  such that  $C_D(s_1 \cdots s_{r-1} v) \leq c + \varepsilon$ . Since  $q_t$  is also the probability that  $s_1 \cdots s_{r-1} v$  fails on the first  $t - 1$  steps for  $t \leq r$ , we have  $\sum_{t=1}^r q_t \leq C_D(s_1 \cdots s_{r-1} v) \leq c + \varepsilon$ . This holds for all  $r$ . So, we have  $C_D(s) \leq c + \varepsilon$ . Since this holds for all  $\varepsilon > 0$ , we have  $C_D(s) \leq c$ . Consequently,  $C_D(s) = c$ :  $s$  is an optimal and full strategy.  $\square$

The following two results show what is the structure of an optimal strategy.

**Theorem 17.** *Let  $D = (D_1, D_2, \dots)$  be a sequence of distributions. We assume that we have in  $D$  a finite number of pairwise different distributions but an infinite number of copies of each of them in  $D$ . There exists a sequence of indices  $i_1 < i_2 < \dots$  and an integer  $m$  such that  $D_{i_1} = D_{i_2} = \dots$  and  $s = i_1^m i_2^m \cdots$  is an optimal strategy of complexity  $\frac{C_D(i_1^m)}{\Pr_D(i_1^m)}$ .*

Here are examples of optimal  $m$  for different distributions.

*Example 18 (Uniform Distribution).* For the uniform distribution  $p_i = \frac{1}{n}$ , with  $1 \leq i \leq n$ . We get  $\Pr_D(1^m) = \frac{m}{n}$  and  $G_m = \frac{m+1}{2}$ . With this we obtain  $C_D(1^m 2^m \cdots) = n - \frac{m-1}{2}$ . Thus, the value of  $m$  that minimizes the complexity is  $m = n$  and  $C_D(1^m 2^m \cdots) = \frac{n-1}{2}$ . The best strategy is to exhaustively search the key until it is found.

*Example 19 (Geometric Distribution).* For the geometric distribution with parameter  $p$ , we have  $p_i = (1 - p)^{i-1} p$ , with  $i = 1, 2, \dots$  or  $D_i = [p, p, \dots]$ . Due to Lemma 5, we can see that for every infinite strategy  $s$ ,  $C_D(s) = \frac{1}{p}$ .

In Appendix A we study concatenations of uniform distributions.

We note that Theorem 17 does not extend if some distribution has a finite number of copies as the following example shows.

*Example 20 (Distribution with No Optimal Strategy of the Form  $i_1^m i_2^m \cdots$ ).* Let  $D_1 = [1 - \varepsilon, \varepsilon, \varepsilon, \dots, \varepsilon, 1]$  of support  $n$  and  $D_2 = D_3 = \dots = [p, \dots, p, 1]$  for  $\varepsilon < p \leq \frac{1}{2}$  and  $n$  large enough. Given a full strategy  $s$ , the formula in Lemma 5 defines a sequence  $q_t(s) = p'_{s_t, \#occ_{s_1 \dots s_t}(s_t)}$ . We can see that for all full strategies  $s$  and  $s'$ , if  $|s| \leq |s'|$  and  $q_t(s) \geq q_t(s')$  for  $t = 1, \dots, |s|$ , then  $C_D(s) \leq C_D(s')$ .

With this, we can see that  $s = 12^n$  is better than all full strategies with length at least  $n + 1$ . There are only two full strategies with smaller length:  $1^n$  and  $2^n$ . We have  $C_D(2^n) = \frac{1-(1-p)^n}{p} \approx \frac{1}{p} \geq 2$  as  $n$  grows. We have  $C_D(12^n) = 1 + \varepsilon \frac{1-(1-p)^n}{p} \approx 1 + \frac{\varepsilon}{p}$  as  $n$  grows, so  $C_D(12^n) < C_D(2^n)$  for  $n$  large enough. We have  $C_D(1^n) = 1 + \varepsilon \frac{1-(1-\varepsilon)^{n-1}}{\varepsilon} = 2 - (1 - \varepsilon)^{n-1} \approx 2$  so  $C_D(12^n) < C_D(1^n)$  for  $n$  large enough. For all strategies of length at least  $n + 1$ ,  $s = 12^n$  collected the largest possible  $p'$  values. So, the best strategy is  $s = 12^n$ . It is better than any strategy of form  $i_1^{m_1} i_2^{m_2} \dots$ .

When we have a finite number of distributions, we may have no optimal strategy of the form in Theorem 17. We may have multiple layers of repetition of  $i^m$  as the following result shows.

**Theorem 21.** *Let  $D_1$  be a distribution of finite support  $n$ . Let  $D = (D_1, D_2, \dots, D_{|D|})$  be a finite sequence of length  $|D|$  in which  $D_1 = D_2 = \dots = D_{|D|}$ . There exists a sequence  $m_1, \dots, m_r$  such that the strategy*

$$s = 1^{m_1} 2^{m_1} \dots |D|^{m_1} 1^{m_2} 2^{m_2} \dots |D|^{m_2} \dots 1^{m_r}$$

*is optimal.*

We provide toy examples below.

*Example 22.* We take  $D = (D_1, D_2)$  with  $D_1 = D_2 = (\frac{3}{5}, \frac{9}{25}, \frac{1}{50}, \frac{1}{50}) = [\frac{3}{5}, \frac{18}{20}, \frac{1}{2}, 1]$ . Here are the complexities of some full strategies.

$$\begin{aligned} C_D(1111) &= \frac{146}{100} = 1.46 \\ C_D(12111) &= \frac{792}{500} = 1.584 \\ C_D(11211) &= \frac{732}{500} = 1.464 \\ C_D(121211) &= \frac{7892}{5000} = 1.5784 \\ C_D(112211) &= \frac{7292}{5000} = 1.4584 \end{aligned}$$

so the last strategy is the best one. Notice that this is also a greedy strategy.

*Example 23.* We take  $D = (D_1, D_2)$  with  $D_1 = D_2 = (\frac{70}{100}, \frac{20}{100}, \frac{5}{100}, \frac{3}{100}, \frac{1}{100}, \frac{1}{100}) = [\frac{70}{100}, \frac{2}{3}, \frac{1}{2}, \frac{3}{5}, \frac{1}{2}, 1]$ . Here are the complexities of some full strategies.

$$\begin{aligned} C_D(111111) &= 1.48 \\ C_D(1211111) &= 1.44 \\ C_D(12121111) &= 1.438 \\ C_D(121212111) &= 1.439 \\ C_D(121122111) &= 1.444 \end{aligned}$$

so  $s = 12121111$  is the best one. For this example we have that the optimal strategy requires  $m_1 = 1$ ,  $m_2 = 1$  and  $m_3 = 4$ . It is also greedy.

**3.1 Proof of Theorem 17**

To prove the result, we first state a useful lemma.

**Lemma 24 (Is It Better to Do  $s$  or  $s'$  First?).** *If  $s$  and  $s'$  are non-empty and have no index in common (i.e., if  $s_t \neq s'_t$  for all  $t$  and  $t'$ ), then  $ss' \leq_D s's$  if and only if  $\frac{C_D(s)}{\Pr_D(s)} \leq \frac{C_D(s')}{\Pr_D(s')}$  in  $[0, +\infty]$ , with the convention that  $\frac{c}{p} = +\infty$  for  $c > 0$  and  $p = 0$ .*

*Proof.* Due to Lemma 5, when  $\Pr_D(s) < 1$  we have

$$C_D(ss') = C_D(s) + \left(1 - \Pr_D(s)\right) C_D(s'| \neg s)$$

Since  $s'$  does not make use of the distributions which are dropped in  $D| \neg s$ , we have  $C_D(s'| \neg s) = C_D(s')$ . So,

$$C_D(ss') = C_D(s) + \left(1 - \Pr_D(s)\right) C_D(s')$$

This is also clearly the case when  $\Pr_D(s) = 1$ . Similarly,

$$C_D(s's) = C_D(s') + \left(1 - \Pr_D(s')\right) C_D(s)$$

So,  $C_D(ss') \leq C_D(s's)$  is equivalent to

$$C_D(s) + \left(1 - \Pr_D(s)\right) C_D(s') \leq C_D(s') + \left(1 - \Pr_D(s')\right) C_D(s)$$

So, this inequality is equivalent to  $\frac{C_D(s)}{\Pr_D(s)} \leq \frac{C_D(s')}{\Pr_D(s')}$ . □

We can now prove Theorem 17.

*Proof (of Theorem 17).* Due to Lemma 16, we know that optimal full strategies exist. Let  $s$  be one of these. We let  $i$  be the index of an arbitrary key which is tested in  $s$ . We can write  $s = u_0 i^{m_1} u_1 i^{m_2} \dots i^{m_r} u_r$  where  $i$  appears in no  $u_j$  and  $m_j > 0$  for all  $j$ , and  $u_1, \dots, u_{r-1}$  are non-empty.

Since  $s$  is optimal, by permuting  $i^{m_j}$  and either  $u_{j-1}$  or  $u_j$ , we obtain larger complexities. So, by applying Lemma 24, we obtain

$$\frac{C_D(i^{m_1})}{\Pr_D(i^{m_1})} \leq \frac{C_D(u_1 | \neg u_0)}{\Pr_D(u_1 | \neg u_0)} \leq \frac{C_D(i^{m_2} | \neg i^{m_1})}{\Pr_D(i^{m_2} | \neg i^{m_1})} \leq \dots \leq C_D(u_r | \neg u_0 \dots u_{r-1})$$

We now want to replace  $u_r$  in  $s$  by some isomorphic copy of  $s$  which is not overlapping with  $u_0 i^{m_1} u_1 i^{m_2} \dots i^{m_r}$ . Due to the optimality of  $s$ , we would deduce

$$C_D(u_r | \neg u_0 \dots u_{r-1}) \leq C_D(s | \neg u_0 \dots u_{r-1}) = C_D(s)$$

so  $\frac{C_D(i^{m_1})}{\Pr_D(i^{m_1})} \leq C_D(s)$  which would imply that the repetition of isomorphic copies of  $i^{m_1}$  are at least as good as  $s$ , so  $\frac{C_D(i^{m_1})}{\Pr_D(i^{m_1})} = C_D(s)$  due to the optimality of  $s$ .

But to replace  $u_r$  in  $s$  by the isomorphic copy of  $s$ , we need to rewrite the original  $s$  containing  $u_r$  by some isomorphic copy in which indices are left free to implement another isomorphic copy of  $s$ .

For that, we split the sequence  $(1, 2, 3, \dots)$  into two subsequences  $v$  and  $v'$  which are non-overlapping (i.e.  $v_t \neq v_{t'}$  for all  $t$  and  $t'$ ), complete (i.e. for every integer  $j$ ,  $v$  contains  $j$  or  $v'$  contains  $j$ ), and representing each distribution with infinite number of occurrences (i.e. for all  $j$ , there exist infinite sequences  $t_1 < t_2 < \dots$  and  $t'_1 < t'_2 < \dots$  such that  $D_j = D_{v_{t_\ell}} = D_{v'_{t'_\ell}}$  for all  $\ell$ ). For that, we can just construct  $v$  and  $v'$  iteratively: for each  $j$ , if the number of  $j' < j$  such that  $D_{j'} = D_j$  in  $v$  or  $v'$  is the same, we put  $j$  in  $v$ , otherwise (we may have only one more instance in  $v$ ), we put  $j$  in  $v'$  (to balance again). For instance, if all  $D_i$  are equal, this construction puts all odd  $j$  in  $v$  and all even  $j$  in  $v'$ . Hence, we can define  $s' = \text{new}_v(s)$  and  $s'' = \text{new}_{v'}(s)$ .  $s'$  will thus only use indices in  $v'$  while  $s''$  will only use indices in  $v$ . Therefore,  $s'$  and  $s''$  will be isomorphic, with no index in common. So,  $C_D(s) = C_D(s') = C_D(s'')$ .

Following the split of  $s$ , the strategy  $s'$  can be written  $s' = u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r} u'_r$  with

$$\frac{C_D(i'^{m_1})}{\text{Pr}_D(i'^{m_1})} = \frac{C_D(i'^{m_1})}{\text{Pr}_D(i'^{m_1})} \leq C_D(u'_r | -u'_0 \dots u'_{r-1}) = C_D(u'_r | -u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r})$$

If we replace  $u'_r$  in  $s'$  by  $s''$ , since  $s'$  is optimal, we obtain a larger complexity. So,

$$C_D(u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r} u'_r) \leq C_D(u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r} s'')$$

These two strategies have the prefix  $u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r}$  in common. We can write their complexities by splitting this common prefix using Lemma 5. By eliminating the common terms, we deduce

$$C_D(u'_r | -u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r}) \leq C_D(s'' | -u'_0 i'^{m_1} u'_1 i'^{m_2} \dots i'^{m_r}) = C_D(s'') = C_D(s)$$

We deduce

$$\frac{C_D(i'^{m_1})}{\text{Pr}_D(i'^{m_1})} \leq C_D(s)$$

Let  $i_1 < i_2 < \dots$  be a sequence of keys using the distribution  $D_i$ . By Lemma 9, the strategy  $i_1^m i_2^m \dots$  has complexity  $\frac{C_D(i'^{m_1})}{\text{Pr}_D(i'^{m_1})}$ . Since  $s$  is optimal, we have  $\frac{C_D(i'^{m_1})}{\text{Pr}_D(i'^{m_1})} \geq C_D(s)$ . Therefore,  $\frac{C_D(i'^{m_1})}{\text{Pr}_D(i'^{m_1})} = C_D(s)$ .  $\square$

### 3.2 Proof of Theorem 21

For the proof of Theorem 21 we need the result of the following lemma.

**Lemma 25.** *Let  $s = ui^a v j^b w$  be an optimal strategy with  $n$  occurrences of each key. We assume that  $i \neq j$ ,  $a < b$ ,  $u$  does not end with  $i$ ,  $v$  has no occurrence of either  $i$  or  $j$ , and  $w$  has equal number of occurrences for  $i$  and  $j$ . Furthermore, we assume that either  $a \neq 0$ , or  $v$  is nonempty and starts with some  $k$  such that  $u$  does not end with  $k$ . Then,  $C_D(s) = C_D(uj^{b-a}i^a v j^a w)$ .*

Lemma 25 will be used in two ways.

1. For  $s = u' j^c v j^b w$  with  $c > 0, b > 0, v$  with no  $i$  or  $j$ , and balanced occurrences of  $i$  and  $j$  in  $w$ , which has the same complexity as  $s' = u' j^{b+c} v w$  (so, to apply the lemma we define  $a = 0, u = u' j^c, k = j$ , and  $s = u' j^c i^0 v j^b w$ ; all hypotheses are verified except  $v$  non-empty, but the result is trivial for empty  $v$ ). This means that we can regroup  $j^c$  and  $j^b$  when there are separated by a  $v$  with no  $i$  and followed by a balanced tail  $w$ .
2. For  $s = ui^a v j^b w$  with  $0 < a < b, v$  with no  $i$  or  $j$ , and balanced occurrences of  $i$  and  $j$  in  $w$ , which has the same complexity as  $s' = u j^{b-a} i^a v j^a w$ . This means that we can balance  $i^a$  and  $j^b$  when there are separated by a  $v$  with no  $i$  or  $j$  and followed by a balanced tail  $w$ .

The proof of Lemma 25 is given in Appendix B.

In what follows, we say that a strategy is in a *normal form* if for all  $t, i \mapsto \#occ_{s_1 \dots s_t}(i)$  is a non-increasing function, i.e.  $\#occ_{s_1 \dots s_t}(i) \geq \#occ_{s_1 \dots s_t}(i + 1)$  for all  $i$ . For instance, 1112322133 is normal as the number of STEP(1) is at no time lower than the number of STEP(2) and the same for the number of STEP(2) and STEP(3).

Since all distributions are the same, all strategies can be rewritten into an equivalent one in a normal form: for this, for the smallest  $t$  such that there exists  $i$  such that  $\#occ_{s_1 \dots s_t}(i) < \#occ_{s_1 \dots s_t}(i + 1)$ , it must be that  $s_t = i + 1$  and  $\#occ_{s_1 \dots s_{t-1}}(i) = \#occ_{s_1 \dots s_{t-1}}(i + 1)$ . We can permute all values  $i$  and  $i + 1$  in the tail  $s_t s_{t+1} \dots$  and obtain an equivalent strategy on which the function becomes non-increasing at step  $t$  and is unchanged before. By performing enough such rewriting, we obtain an equivalent strategy in normal form. For instance, 12231332 is not normal. The smallest  $t$  is  $t = 3$  when we make a second STEP(2) while we only did a single STEP(1). So, we permute 1 and 2 at this time and obtain 12132331. Then, we have  $t = 7$  and permute 2 and 3 to obtain 12132321. Then, again  $t = 7$  to permute 1 and 2 to obtain 12132312 which is normal.

We now prove Theorem 21.

*Proof (of Theorem 21).* Let  $s$  be an optimal strategy. Due to the assumptions, it must be finite. We assume w.l.o.g. that  $s$  is in normal form. We note that we can always complete  $s$  in a form  $s 2^{a_2} 3^{a_3} \dots$  so that the final strategy has exactly  $n$  occurrences of each  $i$ . So, we assume w.l.o.g. that  $s$  has equal number of occurrences. We write  $s = 1^{m_1} x_1 1^{m_2} x_2 \dots 1^{m_r} x_r$  where the  $x_t$ 's are non-empty and with no 1 inside.

As detailed below, we rewrite  $x_r$  (and push some steps earlier in  $x_{r-1}$ ) so that we obtain a permutation of the blocks  $2^{m_r}, \dots, |D|^{m_r}$ . The rewriting is done by preserving the probability of success (which is 1) and the complexity (which is the optimal complexity). Then, we do the same operation in  $x_{r-1}$

and continue until  $x_1$ . When we are done, each  $x_t$  becomes a permutation of the blocks  $2^{m_t}, \dots, |D|^{m_t}$ . Finally, we normalize the obtained rewriting of  $s$  and obtain the result.

We assume that  $s$  has already been rewritten so that for each  $t' = t+1, \dots, r$ , the  $x_{t'}$  sub-strategy is a permutation of the blocks  $2^{m_{t'}}, \dots, |D|^{m_{t'}}$ . Then, we explain how to rewrite  $x_t$ . We make a loop for  $j = 2$  to  $|D|$ . In the loop, we first regroup all blocks of  $j$ 's by using Lemma 25 with  $i = 1$ : while we can write  $x_t = u'j^c v j^b w'$  where  $c > 0$ ,  $b > 0$ ,  $v$  is non-empty with no  $j$ , and  $w'$  has no  $j$ , we write  $u = 1^{m_1} x_1 1^{m_2} x_2 \dots 1^{m_t} u'$  and  $w = w' 1^{m_{t+1}} x_{t+1} \dots 1^{m_r} x_r$ , and set  $a = 0$  and  $i = 1$ . This rewrites  $x_t = u' j^{b+c} v w'$  by preserving the complexity and making a permutation. When this while loop is complete, we can only find a single block of  $j$ 's in  $x_t$  and write  $x_t = v j^b w'$ , where  $v$  and  $w'$  have no  $j$ . So, we apply again Lemma 25 to balance  $1^{m_t}$  and  $j^b$ : we write  $u = 1^{m_1} x_1 1^{m_2} x_2 \dots x_{t-1}$  and  $w = w' 1^{m_{t+1}} x_{t+1} \dots 1^{m_r} x_r$ , and set  $a = m_t$  and  $i = 1$ . This rewrites  $1^{m_t} x_t$  to  $j^{b-m_t} 1^{m_t} v j^{m_t} w'$  by preserving the complexity and making a permutation. So, this rewrites  $x_t$  to  $v j^{m_t} w'$  and  $x_{t-1}$  to  $x_{t-1} j^{b-m_t}$ . When the loop of  $j$  is complete,  $x_t$  is a permutation of the blocks  $2^{m_t}, \dots, |D|^{m_t}$ .

Interestingly, the sequence  $m_1, \dots, m_r$  is unchanged from our starting optimal normal full strategy  $s$ . If we rather start from an optimal full strategy  $s$  which is not in normal form, we can still see how to obtain this sequence: for each  $t$ ,  $m_1 + \dots + m_t$  is the next record number of steps for an attack  $i$  after the  $m_1 + \dots + m_{t-1}$  record. That is the number of steps for the attack  $i$  when  $s$  decides to move to another attack.  $\square$

### 3.3 Finding the Optimal $m$

We provide here a simple criterion for the optimal  $m$  of Theorem 17.

**Lemma 26.** *We let  $D_1 = (p_1, \dots, p_n) = [p'_1, \dots, p'_n]$  be a distribution and define  $D = (D_1, D_1, \dots)$ . Let  $m$  be such that  $s = 1^m 2^m \dots$  is an optimal strategy based on Theorem 17. We have  $\frac{1}{p'_m} \leq C_D(1^m 2^m \dots) \leq \frac{1}{p_{m+1}}$ .*

*Proof.* We let  $s = 2^m 3^m \dots$ . We know that  $C_D(1^{m+1} s) \geq C_D(1^m s)$  since  $1^m s$  is optimal. So,

$$\begin{aligned} 0 &\leq C_D(1^{m+1} s) - C_D(1^m s) \\ &= (1 - \Pr_D(1^m))(C_D(1s|1^m) - C_D(s)) \\ &= (1 - \Pr_D(1^m))(1 - p'_{m+1} \cdot C_D(s)) \end{aligned}$$

from which we deduce  $\frac{1}{p'_{m+1}} \geq C_D(s)$ . Similarly, we have

$$\begin{aligned} 0 &\geq C_D(1^m s) - C_D(1^{m-1} s) \\ &= (1 - \Pr_D(1^{m-1}))(C_D(1s|1^{m-1}) - C_D(s)) \\ &= (1 - \Pr_D(1^{m-1}))(1 - p'_m \cdot C_D(s)) \end{aligned}$$

from which we deduce  $\frac{1}{p'_m} \leq C_D(s)$ .  $\square$



We note that if  $p_m = p_{m+1}$ , then

$$p'_{m+1} = \frac{p_{m+1}}{p_{m+1} + \dots + p_n} = \frac{p_m}{p_{m+1} + \dots + p_n} > \frac{p_m}{p_m + p_{m+1} + \dots + p_n} = p'_m$$

which is impossible (given the result from Lemma 26). Consequently, we must have  $p_m \neq p_{m+1}$ . So, in distributions when we have sequences of equal probabilities  $p_t$ , we can just look at the largest index  $t$  in the sequence as a possible candidate for being the value  $m$ .

Lemma 26 has an equivalent for Theorem 21 (given in the full version of this paper due to lack of space).

## 4 Applications

### 4.1 Solving Sparse LPN

We will model the Learning Parity with Noise (LPN) problem in our STEP game. As we will see, we use the noise bits as the keys the adversary  $\mathcal{A}$  is trying to guess. First of all, we formally give the definition of the LPN problem.

**Definition 27 (Search LPN).** *Let  $s \xleftarrow{U} \mathbb{Z}_2^k$ , let  $\tau \in ]0, \frac{1}{2}[$  be a constant noise parameter and let  $\text{Ber}_\tau$  be the Bernoulli distribution with parameter  $\tau$ . Denote by  $D_{s,\tau}$  the distribution defined as*

$$\{(v, c) \mid v \xleftarrow{U} \mathbb{Z}_2^k, c = \langle v, s \rangle \oplus d, d \leftarrow \text{Ber}_\tau\} \in \mathbb{Z}_2^{k+1}.$$

An LPN oracle  $\mathcal{O}_{s,\tau}^{\text{LPN}}$  is an oracle which outputs independent random samples according to  $D_{s,\tau}$ .

Given queries from the oracle  $\mathcal{O}_{s,\tau}^{\text{LPN}}$ , the search LPN problem is to find the secret  $s$ .

As studied in [6], the LPN-solving algorithms which are based on BKW [5] have a complexity  $\text{poly} \cdot 2^{\frac{k}{\log_2 k}}$ . The naive algorithm guessing that the noise is 0 and running a Gaussian elimination until this finds the correct solution works with complexity  $\text{poly} \cdot (1 - \tau)^{-k}$ . So, the latter is much better as soon as  $\tau < \frac{\ln 2}{\log_2 k}$ , and in particular for  $\tau = k^{-\frac{1}{2}}$  which is the case for some applications [1,9]. Experiments reported in [6] also show that for  $\tau = k^{-\frac{1}{2}}$ , the Gaussian elimination outperforms the BKW variants for  $k > 500$ .

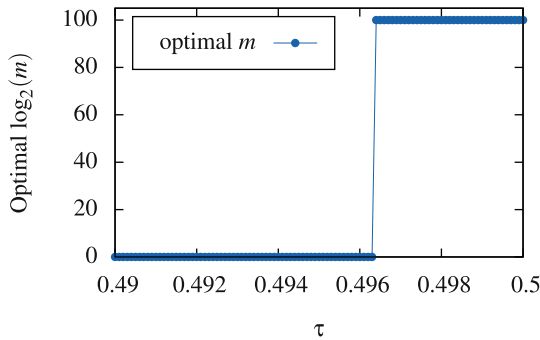
The Gaussian elimination algorithm just reduces to finding a  $k$ -bit noise vector. It guesses that this vector is 0. If this does not work, the algorithm tries again with new LPN queries. We can see this as guessing at least one  $k$ -bit biased vector  $K_i$  which follows the distribution  $D_i = \text{Ber}_\tau^k$  defined by  $\Pr[K_i = v] = \tau^{\text{HW}(v)}(1 - \tau)^{k - \text{HW}(v)}$  in our framework. The most probable vector is  $v = 0$  which has probability  $\Pr[K_i = 0] = (1 - \tau)^k$ . The above algorithm corresponds to trying  $K_1 = 0$  then  $K_2 = 0, \dots$  i.e., the strategy  $123 \dots$  in our framework. We can wonder if there is a better  $1^m 2^m 3^m \dots$ . This is the problem

we study below. We will see that the answer is no: using  $m = 1$  is the best option as soon as  $\tau$  is less than  $\frac{1}{2} - \varepsilon$  for  $\varepsilon = \frac{\ln 2}{2k}$  which is pretty small.

For instance, for  $\text{LPN}_{768, \frac{1}{\sqrt{768}}}$  we obtain  $C_D(12 \dots) = 2^{41}$ . I.e.,  $2^{41}$  calls to the STEP command which corresponds to collecting  $k$  LPN queries and making a Gaussian elimination to recover the secret based on the assumption that the error bits are all 0. If we add up the cost of running Gaussian elimination in order to recover the secret, we obtain a complexity of  $2^{70}$ . This outperforms all the BKW variants and proves that  $\text{LPN}_{768, \frac{1}{\sqrt{768}}}$  is not a secure instance for a 80-bit security. Furthermore, this algorithm outperforms even the covering code algorithm [12]. Our results are strengthened by the results from [6] where we see that there is a big difference between the performance of  $C_D(12 \dots)$  and the one of the covering code algorithm.

$D_i$  is a composite distribution of uniform ones in the sense defined in Appendix A. Namely,  $D_i = \sum_{w=0}^k \tau^k (1 - \tau)^{k-w} U_w$  where  $U_w$  is uniform of support  $\binom{k}{w}$ . By Theorem 17, we know that there exists a magic  $m$  for which the strategy  $s = 1^m 2^m \dots$  is optimal. The analysis of composite distributions further says that  $m$  must be of form  $m = B_w = \sum_{i=0}^w \binom{k}{i}$  for some magic  $w$ . Let  $c_m$  be the complexity of  $1^m 2^m \dots$ . A value  $w = k$ , i.e.  $m = n$  corresponds to the exhaustive search of the noise bits. For  $w = 0$ , i.e.  $m = 1$ , the adversary assumes that the noise is 0 every time he receives  $k$  queries from the LPN oracle.

We first computed experimentally the optimal  $m$  for the  $\text{LPN}_{100, \tau}$  instance where we take  $0 < \tau < \frac{1}{2}$ . The magic  $m$  takes the value 1 for a  $\tau$  which is not close to  $\frac{1}{2}$ . As shown on Fig. 1, it changes to  $n = 2^{100}$  around the value  $\tau = 0.4965$ . This boundary between two different strategies corresponds to the value  $\tau = \frac{1}{2} - \frac{\ln 2}{2k}$  computed in our analysis below. Interestingly, there is no intermediate optimal  $m$  between 1 and  $n$ .



**Fig. 1.** The change of optimal  $m$  for solving  $\text{LPN}_{100, \tau}$

*For Cryptographic Parameters,  $c_1$  is Optimal.* The optimal  $w$  depends on  $\tau$ . The case when  $\tau$  is lower than  $\frac{1}{k}$  is not interesting as it is likely that no error occurs

so all  $w$  lead to a complexity which is very close to 1. Conversely, for  $\tau = \frac{1}{2}$ , the exhaustive search has a complexity of  $c_n = \frac{1}{2}(2^k + 1)$  and  $w = 0$  has a complexity of  $c_1 = 2^k$ . Actually,  $D_i$  is uniform in this case and we know that the optimal  $m$  completes batches of equal consecutive probabilities. So, the optimal strategy is the exhaustive search.

We now show that for  $\tau < 0.16$ , the best strategy is obtained for  $w = 0$ .

Below, we use  $p_{B_w} = \tau^w(1 - \tau)^{k-w}$  and  $c_1 = (1 - \tau)^{-k}$ .

Let  $w_c$  be a threshold weight and let  $\alpha = \Pr(1^{B_{w_c}})$ . For  $0 < w \leq w_c$ , due to Lemma 26, if  $c_{B_w}$  is optimal we have

$$c_{B_w} \geq \frac{1}{p'_{B_w}} = \frac{\Pr_D(-1^{B_w-1})}{p_{B_w}} \geq \frac{\Pr_D(-1^{B_{w_c}})}{p_{B_w}} = \frac{1 - \alpha}{p_{B_w}} = \frac{1 - \alpha}{\left(\frac{\tau}{1-\tau}\right)^w} c_1 \geq \frac{1 - \alpha}{\frac{\tau}{1-\tau}} c_1$$

For  $\tau < 0.16$ , we have  $\frac{\tau}{1-\tau} < 0.20$ . So, if  $\alpha \leq \frac{4}{5}$  we obtain  $c_{B_w} > c_1$ . This contradicts that  $w$  is optimal. For  $w_c = \tau k$ , the Central Limit Theorem gives us that  $\alpha \approx \frac{1}{2}$  which is less than  $\frac{4}{5}$ . So, no  $w$  such that  $0 < w \leq \tau k$  is optimal.

Now, for  $w \geq w_c$ , we have

$$c_w = \frac{C_D(1^{B_w})}{\Pr_D(1^{B_w})} \geq C_D(1^{B_w}) = \sum_{i=1}^{B_w} ip_i + B_w \Pr_D(-1^{B_w}) \geq B_{w_c} \Pr_D(-1^{B_{w_c}}) = (1 - \alpha)B_{w_c}$$

By using the bound  $B_{w_c} \geq \left(\frac{k}{w_c}\right)^{w_c}$ , for  $w_c = \tau k$  we have  $\alpha \approx \frac{1}{2}$  and we obtain  $c_w \geq \frac{1}{2}\tau^{-\tau k}$ . We want to compare this to  $c_1 = (1 - \tau)^{-k}$ . We look at the variations of the function  $\tau \mapsto -k\tau \ln \tau - \ln 2 + k \ln(1 - \tau)$ . We can see by derivating twice that for  $\tau \in [0, \frac{1}{2}]$ , this function increases then decreases. For  $\tau = 0.16$ , it is positive. For  $\tau = \frac{1}{k}$ , it is also positive. So, for  $\tau \in [\frac{1}{k}, 0.16]$ , we have  $c_{B_w} \geq c_1$ .

Therefore, for all  $\tau < 0.16$ ,  $c_1$  is the best complexity so  $m = 0$  is the magic value. Experiment shows that this remains true for all  $\tau < \frac{1}{2} - \frac{\ln 2}{2k}$ . Actually, we can easily see that  $c_1$  becomes lower than  $\frac{2^k+1}{2}$  for  $\tau \approx \frac{1}{2} - \frac{\ln 2}{2k}$ . We will discuss this in Sect. 5.

*Solving LPN with  $\mathcal{O}(k)$  Queries.* We now concentrate on the  $m = n$  case to limit the query complexity to  $\mathcal{O}(k)$ . (In our framework, we need only  $k$  queries but we would practically need more to check that we did find the correct value). So, we estimate the complexity of the full exhaustive search on one error vector  $x$  of  $k$  bits for LPN, i.e.,  $C_D(1^n)$ . If  $p_t$  is the probability that  $x$  is the  $t$ -th enumerated vector, we have  $C_D(1^n) = \sum_{t=1}^n tp_t$ . For  $t$  between  $B_{w-1} + 1$  and  $B_w$ , the sum of the  $p_t$ 's is the probability that we have exactly  $w$  errors. So,  $C_D(1^n) \leq \sum_{w=0}^k B_w \Pr[w \text{ errors}]$ . We approximate  $\Pr[w \text{ errors}]$  to the continuous distribution. So, the Hamming weight has a normal distribution, with mean  $k\tau$  and standard deviation  $\sigma = \sqrt{k\tau(1 - \tau)}$ . We do the same for

$B_w \approx \frac{2^k}{\sqrt{2\pi}} \int_{-\infty}^{\frac{2w-k}{\sqrt{k}}} e^{-\frac{v^2}{2}} dv$ . With the change of variables  $w = k\tau + t\sigma$ , we have

$$\begin{aligned} C_D(1^n) &\leq \sum_{w=0}^k B_w \Pr[w \text{ errors}] \\ &\approx \frac{2^k}{2\pi} \int_{-\infty}^{+\infty} \left( \int_{-\infty}^{\frac{2w-k}{\sqrt{k}}} e^{-\frac{v^2}{2}} dv \right) \frac{1}{\sigma} e^{-\frac{(w-k\tau)^2}{2\sigma^2}} dw \\ &= \frac{2^k}{2\pi} \iint_{v \leq \frac{2k\tau-k+2t\sigma}{\sqrt{k}}} e^{-\frac{t^2+v^2}{2}} dv dt \end{aligned}$$

The distance between the origin  $(t, v) = (0, 0)$  and the line  $v = \frac{2k\tau-k+2t\sigma}{\sqrt{k}}$  is

$$d = \sqrt{k} \frac{1 - 2\tau}{\sqrt{1 + 4\tau(1 - \tau)}}$$

By rotating the region on which we sum, we obtain

$$C_D(1^n) \approx \frac{2^k}{2\pi} \iint_{x \geq d} e^{-\frac{x^2+y^2}{2}} dx dy = \frac{2^k}{\sqrt{2\pi}} \int_d^{+\infty} e^{-\frac{x^2}{2}} dx \sim \frac{2^k}{d\sqrt{2\pi}} e^{-\frac{d^2}{2}}$$

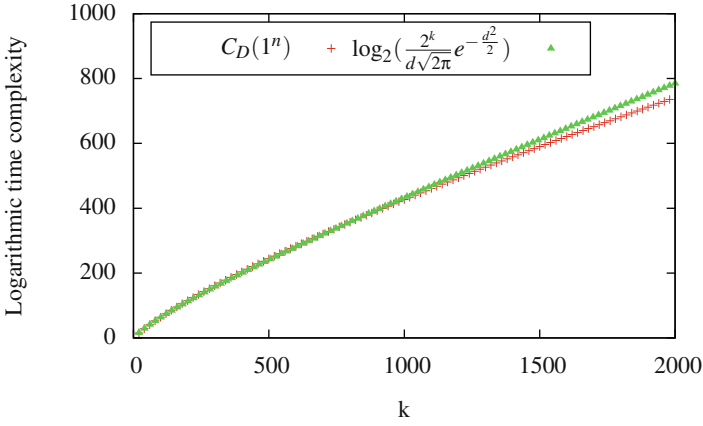
On Fig. 2 we can see that this approximation of  $C_D(1^n)$  is very good for  $\tau = k^{-\frac{1}{2}}$ .

So, the complexity  $C_D(1^n)$  is asymptotically  $2^{k(1 - \frac{1}{2\ln 2}) + \mathcal{O}(\sqrt{k})}$ . Interestingly, the dominant part of  $\log_2 C_D(1^n)$  is  $0.2788 \times k$  and does not depend on  $\tau$  as long as  $\frac{1}{k} \ll \tau \ll \frac{1}{2}$ . Although very good for the low  $k$  that we consider, this approximation of  $C_D(1^n)$  deviates, probably because of the imprecise approximation of the  $B_w$ 's. Next, we derive a bound which is much higher but asymptotically better (the curves crossing for  $k \approx 50\,000$ ). We now use the bound  $B_w \leq k^w$  and do the same computation as before. We have

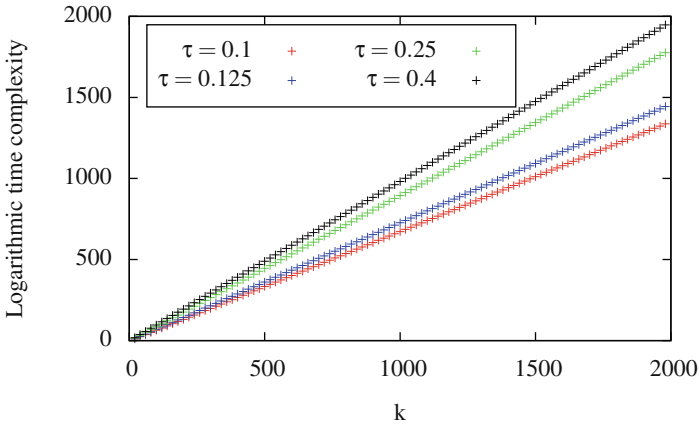
$$\begin{aligned} C_D(1^n) &\leq \sum_{w=0}^k k^w \Pr[w \text{ errors}] \\ &\approx \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} k^{k\tau+t\sigma} e^{-\frac{t^2}{2}} dt \\ &= \frac{e^{\frac{1}{2}(\sigma \ln k)^2 + k\tau \ln k}}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-\frac{(t - \sigma \ln k)^2}{2}} dt \\ &= e^{\frac{1}{2}(\sigma \ln k)^2 + k\tau \ln k} \end{aligned}$$

So,  $C_D(1^n) = e^{\frac{1}{2}\sqrt{k}(\ln k)^2 + \mathcal{O}(\sqrt{k} \ln k)}$  for  $\tau = k^{-\frac{1}{2}}$ . It is better than the  $e^{\mathcal{O}(\frac{k}{\ln k})}$  of Lyubashevsky [16] in the sense that it is asymptotically better and that we use  $\mathcal{O}(k)$  queries instead of  $k^{1+\varepsilon}$ . However, this new bound for  $C_D(1^n)$  is very loose.

Outside the scenario of a sparse LPN, we display in Fig. 3 the logarithmic complexity to solve LPN in our STEP game when the noise parameter is constant.



**Fig. 2.**  $\log_2(C_D(1^n))$  vs.  $\log_2\left(\frac{2^k}{d\sqrt{2\pi}}e^{-\frac{d^2}{2}}\right)$  for  $\tau = k^{-\frac{1}{2}}$



**Fig. 3.**  $\log_2(C_D(1^n))$  for constant  $\tau$

**Table 1.**  $\log_2(C_D(1^n))$  vs.  $\log_2\left(\frac{2^k}{d\sqrt{2\pi}}e^{-\frac{d^2}{2}}\right)$  for  $k = 2000$

$\tau$	$\log_2(C_D(1^n))$	$\log_2\left(\frac{2^k}{d\sqrt{2\pi}}e^{-\frac{d^2}{2}}\right)$
0.1	1350.04	1314.81
0.125	1458.86	1429.33
0.25	1794.57	1788.49
0.4	1966.67	1966.55

Comparing  $\log_2(C_D(1^n))$  with the approximation we obtained, i.e.  $\log_2\left(\frac{2^k}{d\sqrt{2\pi}}e^{-\frac{d^2}{2}}\right)$ , we obtain the following results which validate our approximations (See Table 1).

## 4.2 Password Recovery

There are many news nowadays with attacks and leaks of passwords from different famous companies. From these leaks the community has studied what are the worst passwords used by the users. Having in mind these statistics, we are interested to see what is the best strategy of an outsider that tries to get access to a system having access to a list of users. The goal of the attacker is to hack one account. He can try to hack several accounts in parallel. Within our framework, we compute to see what is the optimal  $m$  for the strategy  $1^m 2^m \dots$ . In this given scenario, the strategy corresponds to making  $m$  guesses for each user until it reaches the end of the list and starting again with new guesses.

We consider the statistics that we have found for the 10 000 Top Passwords<sup>3</sup> and the one done for the database with passwords in clear from the RockYou hack<sup>4</sup>. Studies on the distribution of user's passwords were also done in [7, 10, 22, 23]. The first case-study analyses what are the top 10 000 passwords from a total 6.5 million username-passwords leaked. The most frequent passwords are the following:

password	$p_1 = 0.00493$
123456	$p_2 = 0.00400$
12345678	$p_3 = 0.00133$
1234	$p_4 = 0.00089$

In the case of the RockYou hack, where 32 million of passwords were leaked, we have that the most frequent passwords and their probability of usage is:

123456	$p_1 = 0.009085$
12345	$p_2 = 0.002471$
123456789	$p_3 = 0.002400$
Password	$p_4 = 0.000194$

Moreover, approximately 20 % of the users used the most frequent 5 000 passwords. What these statistics show is that users frequently choose poor and predictable passwords. While dictionary attacks are very efficient, we study here the case where the attacker wants to minimize the number of trials until he gets access to the system, with no pre-computation done. By using our formulas of computing  $C_D(1^m 2^m \dots)$ , we obtain in both of the above distributions that  $m = 1$  is the optimal one. This means that the attacker tries for each username the most probable password and in average after couple of hundred of users (for the two studies we obtain  $C_D$  to be  $\approx 203$  and  $\approx 110$ ), he will manage to access

<sup>3</sup> <https://xato.net/passwords/more-top-worst-passwords/#.VNiORvnF-xW>.

<sup>4</sup> [http://www.imperva.com/docs/WP\\_Consumer\\_Password\\_Worst\\_Practices.pdf](http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf).

the system. We note that having  $m = 1$  is very nice as for the typical password guessing scenario, we need to have a small  $m$  to avoid complications of blocking accounts and triggering an alarm that the system is under an attack.

### 5 On the Phase Transition

Given the experience of the previous applications, we can see that for “regular” distributions, the optimal  $m$  falls from  $m = n$  to the minimal  $m$  as the bias of the distribution increases. We let  $n_1$  be such that  $p_1 = p_2 = \dots = p_{n_1} \neq p_{n_1+1}$  and  $n_2$  be such that  $p_{n_1+1} = \dots = p_{n_1+n_2} \neq p_{n_1+n_2+1}$ . Due to Lemma 26, the magic value  $m$  can only be  $n_1, n_1 + n_2$ , or more. We study here when the curves of  $C_D(1^{n_1}2^{n_1} \dots), C_D(1^{n_1+n_2}2^{n_1+n_2} \dots)$ , and  $C_U(1^n) = \frac{n+1}{2}$  cross each other.

**Lemma 28.** *We consider a composite distribution  $D_1 = \alpha U_1 + \beta U_2 + (1 - \alpha - \beta)D'$ , where  $U_1$  and  $U_2$  are uniform of support  $n_1$  and  $n_2$ . For  $U$  uniform, we have*

$$C_D(1^{n_1}2^{n_1} \dots) \leq C_D(1^{n_1+n_2}2^{n_1+n_2} \dots) \iff \alpha - \beta \frac{n_1}{n_2} \geq \alpha \left( \alpha + \beta \frac{1 - n_1/n_2}{2} \right)$$

$$C_D(1^{n_1}2^{n_1} \dots) \leq C_U(1^n) \iff \frac{n/n_1 + 1}{2} \geq \frac{1}{\alpha}$$

Note that for  $2^{-H_\infty} \geq \frac{2}{n}$ , we have  $\frac{\alpha}{n_1} \geq \frac{2}{n}$  so the second property is satisfied.

As an example, for  $n_1 = n_2 = 1$ , the first condition becomes  $\alpha - \beta \geq \alpha^2$  which is the case of all the distribution we tried for password recovery. The second condition becomes  $2^{-H_\infty} \geq \frac{2}{n+1}$ , which is also always satisfied.

For LPN, we have  $n_1 = 1, n_2 = k, \alpha = (1 - \tau)^k$ , and  $\beta = n_2\tau(1 - \tau)^{k-1}$ . The first and second conditions become

$$(1 - \tau)^k \leq \frac{1 - 2\tau}{1 + \frac{k-3}{2}\tau} \quad \text{and} \quad (1 - \tau)^k \geq \frac{2}{2^k + 1}$$

respectively. They are always satisfied unless  $\tau$  is very close to  $\frac{1}{2}$ : by letting  $\tau = \frac{1}{2} - \varepsilon$  with  $\varepsilon \rightarrow 0$ , the right-hand term of the first condition is asymptotically equivalent to  $\frac{8\varepsilon}{k+1}$  and the left-hand term tends towards  $2^{-k}$ . The balance is thus for  $\tau \approx \frac{1}{2} - \frac{k+1}{8}2^{-k}$ . The second condition gives

$$\tau \leq 1 - \left( \frac{2^k + 1}{2} \right)^{-\frac{1}{k}} = \frac{1}{2} - \frac{\ln 2}{2k} - o\left(\frac{1}{k}\right)$$

So, we can explain the phase transition in  $\text{LPN}_{k,\tau}$  as follows: if we make  $\tau$  decrease from  $\frac{1}{2}$ , for each fixed  $m$ , the complexity of all possible  $C_D(1^m)$  smoothly decrease. The function for  $m = n_1$  crosses the one of  $m = n_1 + n_2$  before it crosses  $\frac{n+1}{2}$  which is close to the value of the one for  $m = n$ . So, the curve for  $m = n_1$  becomes interesting *after* having beaten the curve for  $m = n_1 + n_2$ . This proves that we never have a magic  $m$  equal to  $n_1 + n_2$ . Presumably, it is the case for all other curves as well. This explains the abrupt fall from  $m = n$  to  $m = 1$  which we observed on Fig. 1.

*Proof.* We have

$$C_D(1^{n_1}2^{n_1} \dots) = \frac{C_D(1^{n_1})}{\Pr_D(1^{n_1})} = \frac{\alpha \frac{n_1+1}{2} + (1-\alpha)n_1}{\alpha}$$

and

$$C_D(1^{n_1+n_2}2^{n_1+n_2} \dots) = \frac{C_D(1^{n_1+n_2})}{\Pr_D(1^{n_1+n_2})} = \frac{\alpha \frac{n_1+1}{2} + \beta \left(n_1 + \frac{n_2+1}{2}\right) + (1-\alpha-\beta)(n_1+n_2)}{\alpha+\beta}$$

so

$$\begin{aligned} \frac{C_D(1^{n_1})}{\Pr_D(1^{n_1})} &\leq \frac{C_D(1^{n_1+n_2})}{\Pr_D(1^{n_1+n_2})} \iff \\ \frac{\alpha \frac{n_1+1}{2} + (1-\alpha)n_1}{\alpha} &\leq \frac{\alpha \frac{n_1+1}{2} + \beta \left(n_1 + \frac{n_2+1}{2}\right) + (1-\alpha-\beta)(n_1+n_2)}{\alpha+\beta} \iff \\ \alpha - \beta \frac{n_1}{n_2} &\geq \alpha \left( \alpha + \beta \frac{1-n_1/n_2}{2} \right) \end{aligned}$$

For the second property, we have

$$\begin{aligned} C_D(1^{n_1}2^{n_1} \dots) \leq C_U(1^n) &\iff \frac{C_D(1^{n_1})}{\Pr_D(1^{n_1})} \leq C_U(1^n) \\ &\iff \frac{\alpha \frac{n_1+1}{2} + (1-\alpha)n_1}{\alpha} \leq \frac{n+1}{2} \\ &\iff \frac{n/n_1 + 1}{2} \geq \frac{1}{\alpha} \end{aligned}$$

□

## 6 Conclusions

Our framework enables the analysis of different strategies to sequentialize algorithms when the objective is to make one succeed as soon as possible.

When the algorithms have the same distribution and are unlimited in number, the optimal strategy is of form  $1^m 2^m \dots$  for some magic  $m$ . As the distribution becomes biased, we observe a phase transition from the regular single-algorithm run  $1^n$  (i.e.,  $m = n$ ) to the single-step multiple algorithms  $123 \dots$  (i.e.,  $m = 1$ ) which is very abrupt in the application we considered: LPN and password recovery.

The phase transition phenomenon is further studied. In particular, we show that the fall from  $m = n$  to  $m = 1$  does not go through any  $m \in \{2, \dots, \frac{k(k+1)}{2}\}$ .

For LPN, the solving algorithm we obtain outperforms the classical ones.

When we have a limited number of algorithms, the optimal strategy has the form  $1^{m_1} \dots |D|^{m_1} 1^{m_2} \dots |D|^{m_2} \dots$ . For LPN, this simple algorithm outperforms the classical ones, even the one from Asiacrypt 2014 [12] for the relevant parameters using  $\tau \sim k^{-\frac{1}{2}}$ .



## A Composite Distributions

We give a formula to compute the optimal strategies for distributions obtained by composing several distributions. The formula is useful when we want to regroup equal consecutive  $p_j$ 's in a distribution  $D_1$  so that  $D_1$  appears as a composition of uniform distributions.

**Lemma 29.** *Let  $U_1, \dots, U_k$  be independent distributions of support  $n_1, \dots, n_k$ , respectively. Let  $U_i = (p_{i,1}, \dots, p_{i,n_i})$ . Given a distribution  $(\alpha_1, \dots, \alpha_k)$  of support  $k$ , we define  $D_1 = \alpha_1 U_1 + \alpha_2 U_2 + \dots + \alpha_k U_k$  by  $D_1 = (\alpha_1 p_{1,1}, \dots, \alpha_1 p_{1,n_1}, \alpha_2 p_{2,1}, \dots, \alpha_k p_{k,n_k})$ .*

Let  $m = \sum_{j=1}^i n_j$ . We have

$$\Pr_{D_1}(1^{n_1} 1^{n_2} \dots 1^{n_i}) = \alpha_1 + \dots + \alpha_i$$

$$C_{D_1}(1^{n_1} 1^{n_2} \dots 1^{n_i}) = \sum_{j=1}^i \alpha_j C_{U_j}(1^{n_j}) + \sum_{j=1}^i n_j \left(1 - \sum_{k=1}^j \alpha_k\right)$$

We note that if all  $U_i$  are ordered and if  $\alpha_i p_{i,n_i} \geq \alpha_{i+1} p_{i+1,1}$  for all  $1 \leq i < k$ , then  $D_1$  is ordered as well.

We let  $D = (D_1, D_1, \dots)$ . If we assume that  $U_i$  are uniform distributions, we can use the observation following Lemma 26 to deduce from Theorem 17 that the optimal strategy is  $1^m 2^m \dots$  for  $m = \sum_{j=1}^i n_j$  and  $i$  minimizing

$$\min C_D(\emptyset) = \min_i \left( \frac{\sum_{j=1}^i \alpha_j C_{U_j}(1^{n_j}) + \sum_{j=1}^i n_j \left(1 - \sum_{k=1}^j \alpha_k\right)}{\sum_{j=1}^i \alpha_j} \right)$$

*Proof.* We prove it by induction on  $i$ . It is trivial for  $i = 0$ . We assume the result holds for  $i - 1$ . By induction, we have

$$\begin{aligned} C_{D_1}(1^{n_1} \dots 1^{n_i}) &= C_{D_1}(1^{n_1} \dots 1^{n_{i-1}}) + (1 - \Pr_{D_1}(1^{n_1} \dots 1^{n_{i-1}})) C_{D_1}(1^{n_i} | \neg(1^{n_1} \dots 1^{n_{i-1}})) \\ &= \sum_{j=1}^{i-1} \alpha_j C_{U_j}(1^{n_j}) + \sum_{j=1}^{i-1} n_j \left(1 - \sum_{k=1}^j \alpha_k\right) + \alpha_i C_{U_i}(1^{n_i}) + n_i \left(1 - \sum_{k=1}^i \alpha_k\right) \\ &= \sum_{j=1}^i \alpha_j C_{U_j}(1^{n_j}) + \sum_{j=1}^i n_j \left(1 - \sum_{k=1}^j \alpha_k\right) \end{aligned}$$

The second equality is obtained from the fact that

$$\begin{aligned} C_{D_1}(1^{n_i} | \neg(1^{n_1} \dots 1^{n_{i-1}})) &= \frac{\alpha_i}{\alpha_i + \dots + \alpha_k} (p_{i,1} + 2p_{i,2} + \dots + n_i p_{i,n_i}) + n_i \left(\frac{\alpha_{i+1} + \dots + \alpha_k}{\alpha_i + \dots + \alpha_k}\right) \\ &= \frac{\alpha_i}{1 - \Pr_{D_1}(1^{n_1} \dots 1^{n_{i-1}})} C_{U_i}(1^{n_i}) + n_i \left(\frac{1 - \Pr_{D_1}(1^{n_1} \dots 1^{n_{i-1}}) - \alpha_i}{1 - \Pr_{D_1}(1^{n_1} \dots 1^{n_{i-1}})}\right) \end{aligned}$$

□

## B Proof of Lemma 25

*Proof.* We will show below that there exists  $d > 0$  such that  $a \leq b - d$  and  $C_D(s) = C_D(uj^d i^a v j^{b-d} w)$ . Hence, we can rewrite  $s$  by replacing  $u$  by  $uj^d$  and  $b$  by  $b - d$ . Since  $d > 0$  and  $a \leq b - d$ , we can just apply this rewriting rule enough time until  $b$  is lowered down to  $a$ . Hence, we obtain the result.

To find  $d$ , we first write  $s = u_0 i^{m_1} u_1 i^{m_2} \dots i^{m_r} u_r i^a v j^b w$  where  $i$  appears in no  $u_t$ , the  $m_t$  are nonzero, and  $u_1, \dots, u_r$  are non-empty. (Note that since  $a < b$ , we must have  $m_1 + \dots + m_r > 0$  so  $r \geq 1$ .) Let  $n'$  be the equal number of occurrences of  $i$  and  $j$  in  $ui^a v j^b$ . Let  $t$  be the smallest index such that  $m_1 + \dots + m_t > n' - b$ . (For  $t = 0$ , the left-hand term is 0 but  $n' \geq b$ ; for  $t = r$ , the left-hand term is  $n' - a$  and we know that  $a < b$ ; so,  $t$  exists and  $t > 0$ .) We write  $m_t = m' + d$  such that  $m_1 + \dots + m_{t-1} + m' = n' - b$ . So,  $d > 0$ . Note that  $b - d = b - m_t + m' = n' - m_1 - \dots - m_t = m_{t+1} + \dots + m_r + a$ . So,  $b - d \geq a$ . Clearly,  $d \leq b$ . We write  $s = Hi^d Bi^a v j^d T$  with head  $H = u_0 i^{m_1} u_1 i^{m_2} \dots u_{t-1} i^{m'}$ , body  $B = u_t i^{m_{t+1}} \dots i^{m_r} u_r$ , and tail  $T = j^{b-d} w$ . Clearly,  $H$  has  $n' - b$  occurrences of  $i$  and  $Hi^d Bi^a v$  has  $n' - b$  occurrences of  $j$ . Since  $s$  is optimal for  $D$ ,  $i^d Bi^a v j^d$  is optimal for  $D|H$ . We note that  $B$  does not start with  $i$  ( $t$  is between 1 and  $r$  and  $u_t$  is nonempty and with no  $i$ ) and that  $i^a v$  is non-empty and with no  $j$  (either  $a \neq 0$  or  $v$  is nonempty and with no  $j$ ). We split  $i^d Bi^a v j^d = i^d x_1 \dots x_\ell i^a y_1 \dots y_{\ell'} j^d$  where two consecutive blocks in the list  $i^d, x_1, \dots, x_\ell, i^a, y_1, \dots, y_{\ell'}, j^d$  have no key in common. (For  $a = 0$ , we can always split so that  $x_\ell$  and  $y_1$  have no key in common by using the first term  $k$  of  $v$  which is not the last of  $u$ : we just take  $y_1$  as a block of  $k$ 's and  $x_\ell$  as a block with no  $k$ .) We can apply Lemma 24 and obtain

$$\frac{C_D(i^d | \neg i^{n'-b})}{\Pr_D(i^d | \neg i^{n'-b})} \leq \frac{C_D(i^a | \neg i^{n'-a})}{\Pr_D(i^a | \neg i^{n'-a})} \leq \frac{C_D(y_1 | \neg \dots)}{\Pr_D(y_1 | \neg \dots)} \leq \frac{C_D(y_{\ell'} | \neg \dots)}{\Pr_D(y_{\ell'} | \neg \dots)} \leq \frac{C_D(j^d | \neg j^{n'-b})}{\Pr_D(j^d | \neg j^{n'-b})}$$

Since the first and the last terms are equal, all of them are equal. So, we can permute two consecutive blocks which have no index in common. Hence, we can propagate  $j^d$  earlier until it is stepped before  $i^a$ , since we know there is no other occurrence of  $j$  in the exchanged blocks. We obtain that

$$C_D(Hi^d Bi^a v j^d T) = C_D(Hi^d B j^d i^a v T)$$

as announced. □

## References

1. Alekhovich, M.: More on average case vs approximation complexity. In: Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003), 11–14 October 2003, pp. 298–307. IEEE Computer Society, Cambridge (2003)
2. Avoine, G., Bourgeois, A., Carpent, X.: Analysis of rainbow tables with fingerprints. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, vol. 9144, pp. 356–374. Springer, Heidelberg (2015)

3. Avoine, G., Carpent, X.: Optimal storage for rainbow tables. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 144–157. Springer, Heidelberg (2014)
4. Avoine, G., Junod, P., Oechslin, P.: Time-memory trade-offs: false alarm detection using checkpoints. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 183–196. Springer, Heidelberg (2005)
5. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: Frances Yao, F., Luks, E.M. (eds.) Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, Portland, OR, USA, 21–23 May 2000, pp. 435–440. ACM (2000)
6. Bogos, S., Tramèr, F., Vaudenay, S.: On solving LPN using BKW and variants. *Crypt. Commun.* (2015, to appear)
7. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: IEEE Symposium on Security and Privacy, SP 2012, San Francisco, California, USA, 21–23 May 2012, pp. 538–552. IEEE Computer Society (2012)
8. Carrijo, J., Tonicelli, R., Imai, H., Nascimento, A.C.A.: A novel probabilistic passive attack on the protocols HB and HB<sup>+</sup>. *IEICE Trans.* **92-A**(2), 658–662 (2009)
9. Damgård, I., Park, S.: Is public-key encryption based on LPN practical? *IACR Cryptology ePrint Archive*, 2012:699 (2012)
10. Dell’Amico, M., Michiardi, P., Roudier, Y.: Password strength: an empirical analysis. In: INFOCOM 2010, 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, San Diego, CA, USA, 15–19 March 2010, pp. 983–991. IEEE (2010)
11. Fossorier, M.P.C., Mihaljević, M.J., Imai, H., Cui, Y., Matsuura, K.: An algorithm for solving the LPN problem and its application to security evaluation of the HB protocols for RFID authentication. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 48–62. Springer, Heidelberg (2006)
12. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 1–20. Springer, Heidelberg (2014)
13. Hellman, M.E.: A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory* **26**(4), 401–406 (1980)
14. Huang, J., Vaudenay, S., Lai, X., Nyberg, K.: Capacity and data complexity in multidimensional linear attack. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 141–160. Springer, Heidelberg (2015)
15. Leveil, 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)
16. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX and RANDOM 2005. LNCS, vol. 3624, pp. 378–389. Springer, Heidelberg (2005)
17. Massey, J.L.: Guessing and entropy. In: Proceedings of the 1994 IEEE International Symposium on Information Theory, p. 204, June 1994
18. Meier, W., Staffelbach, O.: Analysis of pseudo random sequences generated by cellular automata. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 186–199. Springer, Heidelberg (1991)
19. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: Atluri, V., Meadows, C., Juels, A. (eds.) Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, 7–11 November 2005, pp. 364–372. ACM (2005)
20. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003)

21. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56**(6), 1–40 (2009)
22. Schneier, B.: Real-world passwords, December 2006. [https://www.schneier.com/blog/archives/2006/12/realworld\\_passw.html](https://www.schneier.com/blog/archives/2006/12/realworld_passw.html)
23. Weir, M., Aggarwal, S., Collins, M.P., Stern, H.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, 4–8 October 2010*, pp. 162–175. ACM (2010)