Universal Randomized Guessing with Application to Asynchronous Decentralized Brute–Force Attacks

Neri Merhav[*] Asaf Cohen[†]

November 10, 2018

Abstract

Consider the problem of guessing the realization of a random vector $\boldsymbol{X}$ by repeatedly submitting queries (guesses) of the form "Is $\boldsymbol{X}$ equal to $\boldsymbol{x}$?" until an affirmative answer is obtained. In this setup, a key figure of merit is the number of queries required until the right vector is identified, a number that is termed the *guesswork*. Typically, one wishes to devise a guessing strategy which minimizes a certain guesswork moment.

In this work, we study a universal, decentralized scenario where the guesser does not know the distribution of $\boldsymbol{X}$, and is not allowed to use a strategy which prepares a list of words to be guessed in advance, or even remember which words were already used. Such a scenario is useful, for example, if bots within a Botnet carry out a brute–force attack in order to guess a password or decrypt a message, yet cannot coordinate the guesses between them or even know how many bots actually participate in the attack.

We devise universal decentralized guessing strategies, first, for memoryless sources, and then generalize them for finite–state sources. In each case, we derive the guessing exponent, and then prove its asymptotic optimality by deriving a compatible converse bound. The strategies are based on randomized guessing using a universal distribution. We also extend the results to guessing with side information. Finally, for all above scenarios, we design efficient algorithms in order to sample from the universal distributions, resulting in strategies which do not depend on the source distribution, are efficient to implement, and can be used asynchronously by multiple agents.

Index Terms: guesswork; universal guessing strategy; randomized guessing; decentralized guessing; guessing with side information; Lempel–Ziv algorithm; efficient sampling from a distribution.

[*]N. Merhav is with the Andrew and Erna Viterbi Faculty of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel. E-mail: `merhav@technion.ac.il`

[†]A. Cohen is with the Department of Communication System Engineering, Ben Gurion University of the Negev, Beer Sheva 84105, Israel. E-mail: `coasaf@bgu.ac.il`

1 Introduction

Consider the problem of guessing the realization of a random n–vector $\boldsymbol{X}$ using a sequence of yes/no queries of the form: "Is $\boldsymbol{X} = \boldsymbol{x}_1$?", "Is $\boldsymbol{X} = \boldsymbol{x}_2$?" and so on, until an affirmative answer is obtained. Given a distribution on $\boldsymbol{X}$, a basic figure of merit in such a guessing game is the *guesswork*, defined as the number of trials required until guessing the right vector.

Devising *guessing strategies* to minimize the guesswork, and obtaining a handle on key analytic aspects of it, such as its *moments* or its *large deviations* rate function, has numerous applications in information theory and beyond. For example, sequential decoding [1, 2] or guessing a codeword which satisfies certain constraints [3]. In fact, since the ordering of all sequences of length n in a descending order of probabilities is, as expected, the optimal strategy under many optimality criteria[1], the guessing problem is intimately related to fixed-to-variable source coding without the prefix constraint, or *one-shot coding*, where it is clear that one wishes to order the possible sequences in a descending probability of appearance before assigning them codewords [4, 5, 6].

Contemporary applications of guesswork focus on information security, that is, guessing passwords or decrypting messages protected by random keys. E.g., one may use guessing strategies and their guesswork exponents while proactively trying to crack passwords, as a mean of assessing password security within an organization [7, 8]. Indeed, it is increasingly important to be able to assess password strength [9], especially under complex (e.g., non-i.i.d.) password composition requirements. While the literature includes several studies assessing strength by measuring how hard it is for common cracking methods to break a certain set of passwords [8, 9] or by estimating the entropy of passwords created under certain rules [10], the guesswork remains a key *analytic tool* in assessing password strength for a given sequence length and distribution. As stated in [11], "we are yet to see compelling evidence that motivated users can choose passwords which resist guessing by a capable attacker". Thus, analyzing the guesswork is useful in assessing how strong a key–generation system is, how hard will it be for a malicious party to break it, or, from the malicious side point of view, how better is one guessing strategy compared to the other.

Arguably, human–created passwords may be of a finite, relatively small length, rather

[1]Specifically, if $G(\boldsymbol{X})$ is the guesswork, this order minimizes $\boldsymbol{E}\{F[G(\boldsymbol{X})]\}$ for any monotone non-decreasing function F.

than long sequences which justify asymptotic analysis of the guesswork. Yet, as mentioned above, the guesswork, as a key figure of merit, may be used to aid in assessing computer generated keys [12] or passwords as well. For example, a random key might be of tens or even hundreds of bits long, and passwords saved on servers are often *salted* before being hashed, resulting in increased length [13]. Moreover, experiments done on finite block lengths agree with the insights gained from the asymptotic analysis [14]. As a result, large deviations and asymptotic analysis remain as key analytic tools in assessing password strength [15, 16, 14, 17, 18]. Such asymptotic analysis provides us, via tractable expressions, the means to understand the guesswork behaviour, the effect various problem parameters have on its value, and the fundamental information measures which govern it. E.g., while the entropy is indeed a relevant measure for "randomness" in passwords [10], via asymptotic analysis of the guesswork [2] we now know that the Rényi entropy is the right measure when *guessing* or even a distributed brute–force attack [18, 19]. Non–asymptotic results, such as the converse result in [20], then give us finer understanding of the dependence on the sequence length.

Keeping the above applications in mind, it is clear the vanilla model of a single, all-capable attacker, guessing a password $\boldsymbol{X}$ drawn from an i.i.d. source of a known distribution, is rarely the case of interest. In practical scenarios, several intricacies complicate the problem. While optimal passwords should have maximum entropy, namely, be memoryless and uniformly distributed over the alphabet, human-created passwords are hardly ever such. They tend to have memory and a non–uniform distribution [21], due to the need to remember them as well as many other practical considerations (e.g., keyboard structure or the native language of the user) [22, 23]. Thus, the ability to efficiently guess non-memoryless passwords and analyze the performance of such guessing strategies is crucial.

Moreover, the underlying true distribution is also rarely known. In [21], the authors investigated the distribution of passwords from four known databases, and tried to fit a Zipf distribution[2]. While there was no clear match, it was clear that a small parameter s is required, to account for a heavy tail. Naturally, [21] also stated that "If the right distribution of passwords can be identified, the cost of guessing a password can be reduced".

Last but not least, from the attacker's side, there might be additional information which

[2]In this model, the probability of password with rank i is $P_i = Ki^s$, where s is a parameter and K is a normalizing constant

facilitates the guessing procedure on the one hand, yet there might be restrictions that prevent him/her from carrying out the optimal guessing strategy. That is, on the one hand, the attacker might have *side information*, e.g., passwords for other services which are correlated with the one currently attacked, and thereby significantly decrease the guesswork [2, 24, 15, 18]. On the other hand, most modern systems will limit the ability of an attacker to submit too many queries from a single IP address, hence to still submit a large amount, these must be submitted from different machines. Such machines may not be synchronized[3], namely, one may not know which queries were already submitted by the other. Moreover, storing a large (usually, an exponentially large) list of queries to be guessed might be a too heavy burden, especially for small bots in the botnet (e.g., IoT devices). The attacker is thus restricted to *distributed brute force attacks*, where numerous devices send their queries simultaneously, yet without the ability to synchronize, without knowing which queries were already sent, or which bots are currently active and which ones failed [19].

Main Contributions

In this paper, we devise *universal*, *randomized* (hence, decentralized) guessing strategies for a wide family of information sources, and assess their performance by analyzing their guessing moments, as well as exponentially matching converse bounds, thereby proving their asymptotic optimality.

Specifically, we begin from the class of memoryless sources, and propose a guessing strategy. The strategy is universal both in the underlying source distribution and in the guesswork moment to be optimized. It is based on a *randomized* approach to guessing, as opposed to an ordered list of guesses, and thus it can be used by asynchronous agents that submit their guesses concurrently. We prove that it achieves the optimal guesswork exponent, we provide an efficient implementation for the random selection of guesses, and finally, extend the results to guessing with side information.

Next, we broaden the scope to a wider family of non–unifilar finite–state sources, namely, hidden Markov sources. We begin with a general converse theorem and then provide a simple matching direct theorem, based on deterministic guessing. We then provide an alternative direct theorem, that employs a randomized strategy, which builds on the Lempel–Ziv (LZ) algorithm [27]. Once again, both results are tight in terms of the guesswork exponent, and

[3]When bots or processes working in parallel are able to be completely synchronized, they may use a pre-compiled list of usernames and passwords - “hard-coding” the guessing strategy [25, 26].

are universal in the source distribution and the moment.

A critical factor in guessing strategies is their implementation. Deterministic approaches require hard-coding long lists (exponentially large in the block length), and hence are memory consuming, while in a randomized approach, one needs to sample from a specific distribution, which might require computing an exponential sum. In this paper, we give two efficient algorithms to sample from the universal distribution we propose. The first algorithm is based on (a repeated) random walk on a growing tree, thus randomly and independently generating new LZ phrases, to be used as guesses. The second algorithm is based on feeding a (slightly modified) LZ decoder with purely random bits. Finally, the results and algorithms are extended to the case with side information.

The rest of this paper is organized as follows. In Section 2, we review the current literature with references both to information–theoretic results and to key findings regarding brute force attacks on passwords. In Section 3, we formally define the problem and our objectives. Section 4 describes the results for memoryless sources, while Section 5 describes the results for sources with memory. Section 6 concludes the paper.

2 Related Work

The first information–theoretic study on guesswork was carried out by Massey [28]. Arikan [2] showed, among other things, that the exponential rate of the number of guesses required for memoryless sources is given by the Rènyi entropy of order $\frac{1}{2}$. Guesswork under a distortion constraint was studied by Arikan and Merhav [29], who also derived a guessing strategy for discrete memoryless sources (DMS's), which is universally asymptotically optimal, both in the unknown memoryless source and the moment order of the guesswork being analyzed.

Guesswork for Markov processes was studied by Malone and Sullivan [30], and extended to a large class of stationary measures by Pfister and Sullivan [3]. In [31], Hanawal and Sundaresan proposed a large deviations approach. They derived the guesswork exponent for sources satisfying a large deviations principle (LDP), and thereby generalized the results in [2] and [30]. In [16], again via large deviations, Christiansen *et al.* proposed an approximation to the *distribution* of the guesswork. In [32], Sundaresan considered guessing under source uncertainty. The redundancy as a function of the *radius* of the family of possible distributions was defined and quantified in a few cases. For the special class of discrete memoryless sources, as already suggested in [29], this redundancy tends to zero as the length of the sequence grows without bound.

In [33], Christiansen *et al.* considered a multi-user case, where an adversary (inquisitor) has to guess U out of V strings, chosen at random from some string-source μ^n. Later, Beirami *et al.* [34] further defined the *inscrutability* $S^n(U, V, \mu^n)$ of a string-source, the *inscrutability rate* as the exponential rate of $S^n(U, V, \mu^n)$, and gave upper and lower bounds on this rate by identifying the appropriate string-source distributions. They also showed that ordering strings by their type-size in ascending order is a universal guessing strategy. Note, however, that both [33, 34] considered a single attacker, with the ability to create a list of strings and guess one string after the other.

Following Weinberger *et al.* [35], ordering strings by the size of their type-class before assigning them *codewords in a fixed-to-variable source coding scheme* was also found useful by Kosut and Sankar in [6] to minimize the third order term of the minimal number of bits required in lossless source coding (the first being the entropy, while the second is the dispersion). A geometric approach to guesswork was proposed by Beirami *et al.* in [36], showing that indeed the dominating type in guesswork (the position of a given string in the list) is the largest among all types whose elements are more likely than the given string. Here we show that a similar ordering is also beneficial for universal, *decentralized guessing*, though the sequences are not ordered in practice, and merely assigned *probabilities to be guessed* based on their type or LZ complexity.

Guesswork over a binary erasure channel was studied by Christiansen *et al.* in [15]. While the underlying sequence to be guessed was assumed i.i.d., the results therein apply to channels with memory as well (yet satisfying an LDP). Interestingly, it was shown that the guesswork exponent is higher than the noiseless exponent times the average fraction of erased symbols, and one pays a non-negligible toll for the randomness in the erasures pattern.

In [18], Salamatian *et al.* considered multi-agent guesswork with side information. The effect of *synchronizing the side information* among the agents was discussed, and its affect on the exponent was quantified. Multi-agent guesswork was then also studied in [19], this time devising a randomized guessing strategy, which can be used by asynchronous agents. The strategy in [19], however, is hard to implement in practice, as it depends on both the source distribution and the moment of the guesswork considered, and requires computing an exponential sum.[4]

Note that besides the standard application of guessing a password for a certain service, while knowing a password of the same user to another service, guessing with side information may also be applicable when breaking lists of hashed *honeywords* [38, 39]. In this scenario, an attacker is faced with a list of hashed sweatwords, where one is the hash of the true password while the rest are hashes of decoy honeywords, created *with strong correlation* to the real password. If one is broken, using it as side information can significantly reduce the time required to break the others. Furthermore, guessing with side information is also related the problem of *guessing using hints* [40]. In this scenario, a legitimate decoder should be able to guess a password (alternatively, a *task* to be carried out) using several hints, in the sense of having a low expected conditional guesswork, yet an eavesdropper knowing only a subset of the hints, should need a large number of guesses. In that case, the expected conditional guesswork generalizes secret sharing schemes by quantifying the amount of work Bob and Eve have to do.

From a more practical viewpoint, trying to create passwords based on real data, Weir *et al.* [41] suggested a context-free grammar to create passwords at a descending order of probabilities, where the grammar rules as well as the probabilities of the generalized letters (sequences of English letters, sequences of digits or sequences of special characters) were learned based on a given training set. In [8], Dell'Amico *et al.* evaluated experimentally the probability of guessing passwords using dictionary-based, grammar-free and Markov chain strategies, using existing data sets of passwords for validation. Not only was it clear that complex guessing strategies, which take into account the memory, perform better, but moreover, the authors stress out the *need to fine-tune memory parameters* (e.g., the length of sub-strings tested), strengthening the necessity for a universal, parameter-free guessing strategy. In [11] Bonneau also implicitly mentions the problems coping with passwords from an unknown distribution, or an unknown mixture of several known distributions.

[4]An algorithm which produces a Markov chain, whose distribution converges to this sum was suggested in the context of randomized guessing tools in [37], yet only asymptotically and only for fixed, known distributions.

3 Notation Conventions, Problem Statement and Objectives

3.1 Notation Conventions

Throughout the paper, random variables will be denoted by capital letters, specific values they may take will be denoted by the corresponding lower case letters, and their alphabets will be denoted by calligraphic letters. Random vectors and their realizations will be denoted, respectively, by capital letters and the corresponding lower case letters, both in the bold face font. Their alphabets will be superscripted by their dimensions. For example, the random vector $\boldsymbol{X} = (X_1, \ldots, X_n)$, ($n$ – positive integer) may take a specific vector value $\boldsymbol{x} = (x_1, \ldots, x_n)$ in $\mathcal{X}^n$, the n–th order Cartesian power of $\mathcal{X}$, which is the alphabet of each component of this vector. Sources and channels will be denoted by the letter P or Q with or without some subscripts. When there is no room for ambiguity, these subscripts will be omitted. The expectation operator will be denoted by $\boldsymbol{E}\{\cdot\}$. The entropy of a generic distribution Q on $\mathcal{X}$ will be denoted by $H_Q(X)$ where X designates a random variable drawn by Q. For two positive sequences a_n and b_n, the notation $a_n \doteq b_n$ will stand for equality in the exponential scale, that is, $\lim_{n\to\infty} \frac{1}{n} \log \frac{a_n}{b_n} = 0$. Similarly, $a_n \overset{\cdot}{\le} b_n$ means that $\limsup_{n\to\infty} \frac{1}{n} \log \frac{a_n}{b_n} \le 0$, and so on. When both sequences depend on a vector, $\boldsymbol{x} \in \mathcal{X}^n$, namely, $a_n = a_n(\boldsymbol{x})$ and $b_n = b_n(\boldsymbol{x})$, the notation $a_n(\boldsymbol{x}) \doteq b_n(\boldsymbol{x})$ means that the asymptotic convergence is uniform, namely, $\lim_{n\to\infty} \max_{\boldsymbol{x}\in\mathcal{X}^n} |\frac{1}{n} \log \frac{a_n(\boldsymbol{x})}{b_n(\boldsymbol{x})}| = 0$. Likewise, $a_n(\boldsymbol{x}) \overset{\cdot}{\le} b_n(\boldsymbol{x})$ means $\limsup_{n\to\infty} \max_{\boldsymbol{x}\in\mathcal{X}^n} \frac{1}{n} \log \frac{a_n(\boldsymbol{x})}{b_n(\boldsymbol{x})} \le 0$, and so on.

The empirical distribution of a sequence $\boldsymbol{x} \in \mathcal{X}^n$, which will be denoted by $\hat{P}_{\boldsymbol{x}}$, is the vector of relative frequencies $\hat{P}_{\boldsymbol{x}}(x)$ of each symbol $x \in \mathcal{X}$ in $\boldsymbol{x}$. The type class of $\boldsymbol{x} \in \mathcal{X}^n$, denoted $\mathcal{T}(\boldsymbol{x})$, is the set of all vectors $\boldsymbol{x}'$ with $\hat{P}_{\boldsymbol{x}'} = \hat{P}_{\boldsymbol{x}}$. Information measures associated with empirical distributions will be denoted with 'hats' and will be subscripted by the sequences from which they are induced. For example, the entropy associated with $\hat{P}_{\boldsymbol{x}}$, which is the empirical entropy of $\boldsymbol{x}$, will be denoted by $\hat{H}_{\boldsymbol{x}}(X)$. Similar conventions will apply to the joint empirical distribution, the joint type class, the conditional empirical distributions and the conditional type classes associated with pairs of sequences of length n. Accordingly, $\hat{P}_{\boldsymbol{xy}}$ would be the joint empirical distribution of $(\boldsymbol{x}, \boldsymbol{y}) = \{(x_i, y_i)\}_{i=1}^n$, $\mathcal{T}(\boldsymbol{x}, \boldsymbol{y})$ or $\mathcal{T}(\hat{P}_{\boldsymbol{xy}})$ will denote the joint type class of $(\boldsymbol{x}, \boldsymbol{y})$, $\mathcal{T}(\boldsymbol{x}|\boldsymbol{y})$ will stand for the conditional type class of $\boldsymbol{x}$ given $\boldsymbol{y}$, $\hat{H}_{\boldsymbol{xy}}(X|Y)$ will be the empirical conditional entropy, and so on.

In Section IV, the broader notion of a type class, which applies beyond the memoryless case, will be adopted: the type class of $\boldsymbol{x}$ w.r.t. a given class of sources $\mathcal{P}$, will be defined as

$$\mathcal{T}(\boldsymbol{x}) = \bigcap_{P \in \mathcal{P}} \{\boldsymbol{x}' : \; P(\boldsymbol{x}') = P(\boldsymbol{x})\}. \tag{1}$$

Obviously, the various type classes, $\{\mathcal{T}(\boldsymbol{x})\}_{\boldsymbol{x} \in \mathcal{X}^n}$, are equivalence classes, and therefore, form a partition of $\mathcal{X}^n$. Of course, when $\mathcal{P}$ is the class of memoryless sources over $\mathcal{X}$, this definition of $\mathcal{T}(\boldsymbol{x})$ is equivalent to the earlier one, provided in the previous paragraph.

3.2 Problem Statement and Objectives

In this paper, we focus on the guessing problem that is defined as follows. Alice selects a secret random n–vector $\boldsymbol{X}$, drawn from a finite alphabet source P. Bob, which is unaware of the realization of $\boldsymbol{X}$, submits a sequence of guesses in the form of yes/no queries: "Is $\boldsymbol{X} = \boldsymbol{x}_1$?" "Is $\boldsymbol{X} = \boldsymbol{x}_2$?", and so on, until receiving an affirmative answer. A *guessing list*, $\mathcal{G}_n$, is an ordered list of all members of $\mathcal{X}^n$, that is, $\mathcal{G} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$, $M = |\mathcal{X}|^n$, and it is associated with a *guessing function*, $G(\boldsymbol{x})$, which is the function that maps $\mathcal{X}^n$ onto $\{1, 2, \ldots, M\}$ by assigning to each $\boldsymbol{x} \in \mathcal{X}^n$ the integer k for which $\boldsymbol{x}_k = \boldsymbol{x}$, namely, the k–th element of $\mathcal{G}_n$. In other words, $G(\boldsymbol{x})$ is the number of guesses required until success, using $\mathcal{G}_n$, when $\boldsymbol{X} = \boldsymbol{x}$.

The guessing problem is about devising a guessing list $\mathcal{G}_n$ that minimizes a certain moment of $G(\boldsymbol{X})$, namely, $\boldsymbol{E}\{G^\rho(\boldsymbol{X})\}$, where $\rho > 0$ is a given positive real (not necessarily a natural number). Clearly, when the source P is known and ρ is arbitrary, the optimal guessing list orders the members of $\mathcal{X}^n$ in the order of non–increasing probabilities. When P is unknown, but known to belong to a given parametric class $\mathcal{P}$, like the class of memoryless sources, or the class of finite–state sources with a given number of states, we are interested in a *universal guessing list*, which is asymptotically optimal in the sense of minimizing the *guessing exponent*, namely, achieving

$$E(\rho) = \limsup_{n \to \infty} \min_{\mathcal{G}_n} \frac{\log \boldsymbol{E}\{G^\rho(\boldsymbol{X})\}}{n}, \tag{2}$$

uniformly for all sources in $\mathcal{P}$ and all positive real values of ρ.

Motivated by applications of distributed, asynchronous guessing by several agents (see Introduction), we will also be interested in *randomized guessing* schemes, which have the advantages of: (i) relaxing the need to consume large volumes of memory (compared to deterministic guessing which needs the storage of the guessing list $\mathcal{G}_n$) and (ii) dropping the need for synchronization among the various guessing agents (see [19]). In randomized guessing, the guesser sequentially submits a sequence of random guesses, each one distributed independently according to a certain probability distribution $\tilde{P}(\boldsymbol{x})$. We would like the distribution $\tilde{P}$ to be universally asymptotically optimal in the sense of achieving (on the average) the optimal guessing exponent, while being independent of the unknown source P and independent of ρ. Another desirable feature of the random guessing distribution $\tilde{P}$ is that it would be easy to implement in practice. This is especially important when n is large, as it is not trivial to implement a general distribution over $\mathcal{X}^n$ in the absence of any structure to this distribution.

We begin our discussion from the case where the class of sources, $\mathcal{P}$, is the class of memoryless sources over a finite alphabet $\mathcal{X}$ of size α. In this case, some of the results we will mention are already known, but it will be helpful, as a preparatory step, before we address the more interesting and challenging case, where $\mathcal{P}$ is the class of all non–unifilar, finite–state sources, a.k.a. hidden Markov sources (over the same finite alphabet $\mathcal{X}$), where even the number of states is unknown to the guesser, let alone, the parameters of the source for a given number of states. In both cases, we also extend the study to the case where the guesser is equipped with side information (SI) $\boldsymbol{Y}$, correlated to the vector $\boldsymbol{X}$ to be guessed.

4 Guessing for Memoryless Sources

4.1 Background

Following [28], Arikan [2] has established some important bounds associated with guessing moments with relation the Rényi entropy, with and without side information, where the main application he had in mind was sequential decoding. Some of Arikan's results set the stage for guessing n–vectors emitted from memoryless sources. Some of these results were later extended to the case of lossy guessing[5] [29] with a certain emphasis on universality issues. In particular, narrowing down the main result of [29] to the case of lossless guessing considered here, it was shown that the best achievable guessing exponent, $E(\rho)$, is given by the following single–letter expression for a given memoryless source P:

$$E(\rho) = \max_Q[\rho H_Q(X) - D(Q\|P)] = \rho H^{1/(1+\rho)}(X), \tag{3}$$

[5]Here, by "lossy guessing" we mean that a guess is considered successful if its distance (in the sense of a given distortion function) from the underlying source vector, does not exceed a given distortion level.

where Q is an auxiliary distribution over $\mathcal{X}$ to be optimized, and $H^{\alpha}(X)$ designates the Rényi entropy of order α,

$$H^{\alpha}(X) = \frac{1}{1-\alpha}\ln\left[\sum_{x\in\mathcal{X}} P^{\alpha}(x)\right], \tag{4}$$

which is asymptotically achieved using a universal deterministic guessing list, $\mathcal{G}_n$, that orders the members of $\mathcal{X}^n$ according to a non–decreasing order of their empirical entropies, namely,

$$\hat{H}_{\boldsymbol{x}_1}(X) \le \hat{H}_{\boldsymbol{x}_2}(X) \le \ldots \le \hat{H}_{\boldsymbol{x}_M}(X). \tag{5}$$

In the presence of correlated side information $\boldsymbol{Y}$, generated from $\boldsymbol{X}$ by a discrete memoryless channel (DMC), the above findings continue to hold, with the modifications that: (i) $H_Q(X)$ is replaced by $H_Q(X|Y)$, (ii) $D(Q\|P)$ is understood to be the divergence between the two joint distributions of the pair (X,Y) (which in turn implies that $H^{1/(1+\rho)}(X)$ is replaced by the corresponding conditional Rényi entropy of X given Y), and (iii) $\hat{H}_{\boldsymbol{x}_k}(X)$ is replaced by $\hat{H}_{\boldsymbol{x}_k\boldsymbol{y}}(X|Y)$, $k = 1, 2, \ldots, M$.

4.2 Randomized Guessing and its Efficient Implementation

For universal randomized guessing, we consider the following guessing distribution

$$\tilde{P}(\boldsymbol{x}) = \frac{2^{-n\hat{H}_{\boldsymbol{x}}(X)}}{\sum_{\boldsymbol{x}'} 2^{-n\hat{H}_{\boldsymbol{x}'}(X)}}. \tag{6}$$

We then have the following result:

Theorem 1 *Randomized guessing according to eq. (6) achieves the optimal guessing exponent (3).*

Proof. We begin from the following lemma, whose proof is deferred to the appendix.

Lemma 1 *For given $a \ge 0$ and $\rho > 0$,*

$$\sum_{k=1}^{\infty} k^{\rho}(1-e^{-na})^{k-1} \overset{\cdot}{\le} e^{(1+\rho)na}. \tag{7}$$

Denoting by $\mathcal{P}_n$ the set of probability distributions over $\mathcal{X}$ with rational letter probabilities of denominator n (empirical distributions), we observe that since

$$1 \quad \le \quad \sum_{\boldsymbol{x}} 2^{-n\hat{H}_{\boldsymbol{x}}(X)}$$

$$
\begin{aligned}
&= \sum_{\boldsymbol{x}} \max_{Q\in\mathcal{P}} Q(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}} \max_{Q\in\mathcal{P}_n} Q(\boldsymbol{x}) \\
&\le \sum_{\boldsymbol{x}} \sum_{Q\in\mathcal{P}_n} Q(\boldsymbol{x}) \\
&= \sum_{Q\in\mathcal{P}_n} \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) \\
&= |\mathcal{P}_n| \\
&\le (n+1)^{|\mathcal{X}|-1}, \qquad (8)
\end{aligned}
$$

it follows that

$$\tilde{P}(\boldsymbol{x}) \doteq 2^{-n\hat{H}_{\boldsymbol{x}}(X)}. \qquad (9)$$

Given that $\boldsymbol{X} = \boldsymbol{x}$, the ρ–th moment of the number of guesses under $\tilde{P}$ is given by

$$
\begin{aligned}
&\sum_{k=1}^{\infty} k^{\rho}[1-\tilde{P}(\boldsymbol{x})]^{k-1}\tilde{P}(\boldsymbol{x}) \\
&= \tilde{P}(\boldsymbol{x})\sum_{k=1}^{\infty} k^{\rho}[1-\tilde{P}(\boldsymbol{x})]^{k-1} \\
&\doteq 2^{-n\hat{H}_{\boldsymbol{x}}(X)}\sum_{k=1}^{\infty} k^{\rho}[1-2^{-n\hat{H}_{\boldsymbol{x}}(X)}]^{k-1} \\
&\overset{\cdot}{\le} 2^{-n\hat{H}_{\boldsymbol{x}}(X)}2^{n(1+\rho)\hat{H}_{\boldsymbol{x}}(X)} \\
&= 2^{n\rho\hat{H}_{\boldsymbol{x}}(X)}, \qquad (10)
\end{aligned}
$$

where in the inequality, we have used Lemma 1 with the assignment $a = \hat{H}_{\boldsymbol{x}}(X)\ln 2$. Taking the expectation of $2^{n\rho\hat{H}_{\boldsymbol{x}}(X)}$ w.r.t. $P(\boldsymbol{x})$, using the method of types [42], one easily obtains (see also [29]) the exponential order of $2^{nE(\rho)}$, with $E(\rho)$ as defined in (3). This completes the proof of Theorem 1. $\square$

Remark: It is easy to see that the random guessing scheme has an additional important feature: not only the expectation of $G^{\rho}(\boldsymbol{x})$ (w.r.t. the randomness of the guesses) has the optimal exponential order of $2^{n\rho\hat{H}_{\boldsymbol{x}}(X)}$ for each and every $\boldsymbol{x}$, but moreover, the probability that $G(\boldsymbol{x})$ would exceed $2^{n[\hat{H}_{\boldsymbol{x}}(X)+\epsilon]}$ decays double–exponentially rapidly for every $\epsilon > 0$. This follows from the following simple chain of inequalities:

$$
\begin{aligned}
\Pr\left\{G(\boldsymbol{x}) \ge 2^{n[\hat{H}_{\boldsymbol{x}}(X)+\epsilon]}\right\} &\doteq \left[1-2^{-n\hat{H}_{\boldsymbol{x}}(X)}\right]^{2^{n[\hat{H}_{\boldsymbol{x}}(X)+\epsilon]}} \\
&= \exp\left\{2^{n[\hat{H}_{\boldsymbol{x}}(X)+\epsilon]}\ln\left[1-2^{-n\hat{H}_{\boldsymbol{x}}(X)}\right]\right\}
\end{aligned}
$$

$$\begin{aligned} &\le \exp\left\{-2^{n[\hat{H}_{\boldsymbol{x}}(X)+\epsilon]}\cdot 2^{-n\hat{H}_{\boldsymbol{x}}(X)}\right\} \\ &= \exp\left\{-2^{n\epsilon}\right\}. \end{aligned} \tag{11}$$

A similar comment will apply also to the random guessing scheme of Section 5.

The random guessing distribution (6) is asymptotically equivalent (in the exponential scale) to a class of mixtures of all memoryless sources over $\mathcal{X}$, having the form

$$M(\boldsymbol{x}) = \int_{\mathcal{S}} \mu(Q)\cdot Q(\boldsymbol{x})\mathrm{d}Q \tag{12}$$

where $\mu(\cdot)$ is a density defined on the simplex of all distributions on $\mathcal{X}$, and where it is assumed that $\mu(\cdot)$ is bounded away from zero and from infinity, and that it is independent of n. As mentioned in [43], one of the popular choices of $\mu(\cdot)$ is the Dirichlet distribution, parametrized by $\lambda > 0$,

$$\mu(Q) = \frac{\Gamma(\lambda|\mathcal{X}|)}{\Gamma^{|\mathcal{X}|}(\lambda)}\cdot\left[\prod_{x\in\mathcal{X}} Q(x)\right]^{\lambda-1}, \tag{13}$$

where

$$\Gamma(s) \stackrel{\triangle}{=} \int_0^\infty x^{s-1}e^{-x}\mathrm{d}x, \tag{14}$$

and we remind that for a positive integer n,

$$\Gamma(n) = (n-1)! \tag{15}$$

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi} \tag{16}$$

$$\Gamma\left(n+\frac{1}{2}\right) = \sqrt{\pi}\cdot\frac{1}{2}\cdot\frac{3}{2}\cdots\left(n-\frac{1}{2}\right), \quad n\ge 1. \tag{17}$$

For example, with the choice $\lambda = 1/2$, the mixture becomes

$$M(\boldsymbol{x}) = \Gamma\left(\frac{|\mathcal{X}|}{2}\right)\cdot\frac{\prod_{x\in\mathcal{X}}\Gamma(n\hat{P}_{\boldsymbol{x}}(x)+1/2)}{\pi^{|\mathcal{X}|/2}\Gamma(n+|\mathcal{X}|/2)}. \tag{18}$$

This mixture distribution can be implemented sequentially, as

$$M(\boldsymbol{x}) = \prod_{t=0}^{n-1} M(x_{t+1}|x^t), \tag{19}$$

where

$$M(x_{t+1}|x^t) = \frac{M(x^{t+1})}{M(x^t)} = \frac{t\hat{P}_{x^t}(x_{t+1})+1/2}{t+|\mathcal{X}|/2}, \tag{20}$$

where $\hat{P}_{x^t}(x)$ is the relative frequency of $x\in\mathcal{X}$ in $x^t = (x_1,\ldots,x_t)$. So the sequential implementation is rather simple: draw the first symbol, X_1, according to the uniform distribution. Then, for $t = 1, 2, \ldots, n-1$, draw the next symbol, X_{t+1}, according to the last equation, taking into account the relative frequencies of the various letters drawn so far.

4.3 Side Information

All the above findings extend straightforwardly to the case of a guesser that is equipped with SI $\boldsymbol{Y}$, correlated to the random vector $\boldsymbol{X}$ to be guessed, where it is assumed that $(\boldsymbol{X}, \boldsymbol{Y})$ is a sequence of n independent copies of a pair of random variables (X, Y) jointly distributed according to P_{XY}.

The only modification required is that the universal randomized guessing distribution will now by proportional (and exponentially equivalent) to $2^{-n\hat{H}_{\boldsymbol{xy}}(X|Y)}$ instead of $2^{-n\hat{H}_{\boldsymbol{x}}(X)}$, and in the sequential implementation, the mixture and hence also the relative frequency counts will be applied to each SI letter $y \in \mathcal{Y}$ separately. Consequently, the conditional distribution $M(x_{t+1}|x^t)$ above would be replaced by

$$M(x_{t+1}|x^t, y^t) = \frac{t\hat{P}_{x^t y^t}(x_{t+1}, y_{t+1}) + 1/2}{t\hat{P}_{y^t}(y_{t+1}) + |\mathcal{X}|/2}. \tag{21}$$

5 Guessing for Finite–State Sources

We now extend the scope to a much more general class of sources – the class of non–unifilar finite–state sources, namely, hidden Markov sources [44]. Specifically, we assume that $\boldsymbol{X}$ is drawn by a distribution P given by

$$P(\boldsymbol{x}) = \sum_{\boldsymbol{z}} \prod_{i=1}^{n} P(x_i, z_{i+1}|z_i), \tag{22}$$

where $\{x_i\}$ is the source sequence as before, whose elements take on values in a finite alphabet $\mathcal{X}$ of size α, and where $\{z_i\}$ is the underlying state sequence, whose elements take on values in a finite set of states, $\mathcal{Z}$ of size s, and where the initial state, z_1, is assumed to be a fixed member of $\mathcal{Z}$. The parameter set $\{P(x, z'|z), x \in \mathcal{X},\ z, z' \in \mathcal{Z}\}$ is unknown the guesser. In fact, even the number of states, s, is not known, and we seek a universal guessing strategy.

5.1 Converse Theorem

Let us parse $\boldsymbol{x}$ into $c = c(\boldsymbol{x})$ distinct phrases, by using, for example, the incremental parsing procedure[6] of the Lempel–Ziv (LZ) algorithm [27] (see also [45, Subsection 13.4.2]). The following is a converse theorem concerning the best achievable guessing performance.

Theorem 2 *Given a finite–state source (22), any guessing function satisfies the following inequality:*

$$\boldsymbol{E}\{G^\rho(\boldsymbol{X})\} \ge 2^{-n\Delta_n}\boldsymbol{E}\left[\exp_2\{\rho c(\boldsymbol{X})\log c(\boldsymbol{X})\}\right], \tag{23}$$

where Δ_n is a function of s, α and n, that tends to zero as $n \to \infty$ for fixed s and α.

Proof. Without essential loss of generality, let ℓ divide n and consider the segmentation of $\boldsymbol{x} = (x_1, \ldots, x_n)$ into n/ℓ non–overlapping sub–blocks, $\boldsymbol{x}_i = (x_{i\ell+1}, x_{i\ell+2}, \ldots, x_{(i+1)\ell})$, $i = 0, 1, \ldots, n/\ell - 1$. Let $\boldsymbol{z}^\ell = (z_1, z_{\ell+1}, z_{2\ell+1}, \ldots, z_{n+1})$ be the (diluted) state sequence pertaining to the boundaries between neighboring sub–blocks. Then,

$$P(\boldsymbol{x}, \boldsymbol{z}^\ell) = \prod_{i=0}^{n/\ell-1} P(\boldsymbol{x}_i, z_{(i+1)\ell+1}|z_{i\ell+1}). \tag{24}$$

For a given $\boldsymbol{z}^\ell$, let $\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^\ell)$ be the set of all sequences $\{\boldsymbol{x}'\}$ that are obtained by permuting different sub–blocks that both begin at the same state and end at the same state. Owing to the product form of $P(\boldsymbol{x}, \boldsymbol{z}^\ell)$, it is clear that $P(\boldsymbol{x}', \boldsymbol{z}^\ell) = P(\boldsymbol{x}, \boldsymbol{z}^\ell)$ whenever $\boldsymbol{x}' \in \mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^\ell)$. It was shown in [46, Eq. (47) and Appendix A] that

$$|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^\ell)| \ge \exp_2\{c(\boldsymbol{x})\log c(\boldsymbol{x}) - n\delta(n,\ell)\}, \tag{25}$$

independently of $\boldsymbol{z}^\ell$, where $\delta(n,\ell)$ tends to C/ℓ ($C > 0$ – constant) as $n \to \infty$ for fixed ℓ. Furthermore, by choosing $\ell = \ell_n = \sqrt{\log n}$, we have that $\delta(n, \ell_n) = O(1/\sqrt{\log n})$. We then have the following chain of inequalities:

$$\begin{aligned}
\boldsymbol{E}\{G^\rho(\boldsymbol{X})\} &= \sum_{\boldsymbol{z}^{\ell_n}}\sum_{\boldsymbol{x}} P(\boldsymbol{x}, \boldsymbol{z}^{\ell_n})G^\rho(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{z}^{\ell_n}}\sum_{\{T(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})\}}\sum_{\boldsymbol{x}'\in\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})} P(\boldsymbol{x}, \boldsymbol{z}^{\ell_n})G^\rho(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{z}^{\ell_n}}\sum_{\{T(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})\}} P(\boldsymbol{x}, \boldsymbol{z}^{\ell_n})\sum_{\boldsymbol{x}'\in\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})} G^\rho(\boldsymbol{x})
\end{aligned}$$

[6]The incremental parsing procedure is a sequential procedure of parsing a sequence, such that each new parsed phrase is the shortest string that has not been obtained before as a phrase.

$$
\begin{aligned}
&= \sum_{\boldsymbol{z}^{\ell_n}} \sum_{\{T(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})\}} P(\boldsymbol{x},\boldsymbol{z}^{\ell_n}) \cdot |\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})| \cdot \sum_{\boldsymbol{x}'\in\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})} \frac{G^\rho(\boldsymbol{x})}{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|} \\
&\geq \sum_{\boldsymbol{z}^{\ell_n}} \sum_{\{T(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})\}} P(\boldsymbol{x},\boldsymbol{z}^{\ell_n}) \cdot |\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})| \cdot \frac{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|^\rho}{1+\rho} \\
&\geq \frac{1}{1+\rho} \sum_{\boldsymbol{z}^{\ell_n}} \sum_{\{T(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})\}} P(\boldsymbol{x},\boldsymbol{z}^{\ell_n}) \cdot |\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})| \cdot \exp_2\{\rho[c(\boldsymbol{x})\log c(\boldsymbol{x}) - n\delta(n,\ell_n)]\} \\
&= \frac{2^{-n\delta(n,\ell_n)}}{1+\rho} \boldsymbol{E}\left[\exp_2\{\rho \cdot c(\boldsymbol{X})\log c(\boldsymbol{X})\}\right] \\
&\stackrel{\triangle}{=} 2^{-n\Delta_n} \boldsymbol{E}\left[\exp_2\{\rho \cdot c(\boldsymbol{X})\log c(\boldsymbol{X})\}\right], \qquad (26)
\end{aligned}
$$

where the first inequality follows from the following genie–aided argument: The inner–most summation in the fourth line of the above chain can be viewed as the guessing moment of a guesser that is *informed* that $\boldsymbol{X}$ falls within a given $\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})$. Since the distribution within $\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})$ is uniform, no matter what guessing strategy may be,

$$
\begin{aligned}
\sum_{\boldsymbol{x}'\in\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})} \frac{G^\rho(\boldsymbol{x})}{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|} &\geq \frac{1}{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|} \sum_{k=1}^{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|} k^\rho \\
&\geq |\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|^\rho \cdot \frac{1}{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|} \sum_{k=1}^{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|} \left(\frac{k}{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|}\right)^\rho \\
&\geq |\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|^\rho \cdot \int_0^1 u^\rho \mathrm{d}u \\
&= \frac{|\mathcal{T}(\boldsymbol{x}|\boldsymbol{z}^{\ell_n})|^\rho}{1+\rho}. \qquad (27)
\end{aligned}
$$

This completes the proof of Theorem 2. □

5.2 Direct Theorem

We now present a matching direct theorem, which asymptotically achieves the converse bound in the exponential scale.

Theorem 3 *Given a finite–state source (22), there exists a universal guessing list that satisfies the following inequality:*

$$\boldsymbol{E}\{G^\rho(\boldsymbol{X})\} \leq 2^{n\Delta'_n} \boldsymbol{E}\left[\exp_2\{\rho c(\boldsymbol{X})\log c(\boldsymbol{X})\}\right], \qquad (28)$$

where Δ'_n is a function of s, α and n, that tends to zero as $n \to \infty$ for fixed s and α.

Proof. The proposed deterministic guessing list orders all members of $\mathcal{X}^n$ in non–decreasing order of their Lempel–Ziv code–lengths [27, Theorem 2]. Denoting the LZ code–length of

$\boldsymbol{x}$ by $LZ(\boldsymbol{x})$, we then have

$$\begin{aligned} G(\boldsymbol{x}) &\le |\{\boldsymbol{x}':\ LZ(\boldsymbol{x}') \le LZ(\boldsymbol{x})\}| \\ &= \sum_{i=1}^{LZ(\boldsymbol{x})} |\{\boldsymbol{x}':\ LZ(\boldsymbol{x}') = i\}| \\ &\le \sum_{i=1}^{LZ(\boldsymbol{x})} 2^i \\ &< 2^{LZ(\boldsymbol{x})+1} \\ &\le \exp_2\{[c(\boldsymbol{x})+1]\log(2\alpha[c(\boldsymbol{x})+1])+1\} \\ &\doteq \exp_2\{c(\boldsymbol{x})\log c(\boldsymbol{x})\}, \end{aligned} \tag{29}$$

where the inequality $|\{\boldsymbol{x}' :\ LZ(\boldsymbol{x}') = i\}| \le 2^i$ is due to the fact that the LZ code is uniquely decipherable (UD) and the last inequality is from Theorem 2 of [27]. By raising this inequality to the power of ρ and taking the expectation of both sides, Theorem 3 is readily proved.

An alternative, randomized guessing strategy pertains to independent random guesses according to the following universal distribution,

$$\tilde{P}(\boldsymbol{x}) = \frac{2^{-LZ(\boldsymbol{x})}}{\sum_{\boldsymbol{x}'} 2^{-LZ(\boldsymbol{x}')}}. \tag{30}$$

Since the LZ code is UD, it satisfies the Kraft inequality, and so, the denominator cannot be larger than 1, which means that

$$\tilde{P}(\boldsymbol{x}) \ge 2^{-LZ(\boldsymbol{x})} \ge \exp_2\{-[c(\boldsymbol{x})+1]\log(2\alpha[c(\boldsymbol{x})+1])\}. \tag{31}$$

Similarly as in (10), applying Lemma 1 to (30) (or to (31)), this time with $a = \frac{\ln 2}{n}[c(\boldsymbol{x}) + 1]\log(2\alpha[c(\boldsymbol{x})+1])$, we obtain that the ρ–th moment of $G(\boldsymbol{X})$ given that $\boldsymbol{X} = \boldsymbol{x}$ is upper bounded by an expression of the exponential of $\exp_2\{\rho[c(\boldsymbol{x})+1]\log(2\alpha[c(\boldsymbol{x})+1])\} \doteq \exp_2\{\rho c(\boldsymbol{x})\log c(\boldsymbol{x})\}$, and then upon taking the expectation w.r.t. the randomness of $\boldsymbol{X}$, we readily obtain the achievability result once again. $\square$

5.3 Algorithms for Sampling From the Universal Guessing Distribution

Similarly as in Section 4, we are interested in efficient algorithms for sampling from the universal distribution, (30). In fact, it is enough to have an efficient implementation of an algorithm that efficiently samples from a distribution $\tilde{P}$ that satisfies $\tilde{P}(\boldsymbol{x}) \overset{\cdot}{\ge} 2^{-c(\boldsymbol{x})\log c(\boldsymbol{x})}$. We propose two different algorithms, the first is inspired by the predictive point of view associated with LZ parsing [47], [48], and the second one is based on the simple idea of feeding the LZ decoder with purely random bits. The latter algorithm turns out to lend itself more easily to generalization for the case of guessing in the presence of SI. Both algorithms are described in terms of walks on a growing tree, but the difference is that in the first algorithm, the tree is constructed in the domain of the source sequences, whereas in the second algorithm, the tree is in the domain of the compressed bit-stream.

First algorithm. As said, the idea is in the spirit of the predictive probability assignment mechanism proposed in [47] and [48, Sect. V], but here, instead of using the incremental parsing mechanism for prediction, we use it for random selection.

As mentioned before, the algorithm is described as a process generated by a repeated walk on a growing tree, beginning, each time, from the root and ending at one of the leaves. Consider a tree which is initially composed of a root connected to α leaves, each one corresponding to one alphabet letter, $x \in \mathcal{X}$. We always assign to each leaf a weight of 1 and to each internal node – the sum of weights of its immediate off-springs, and so, the initial weight of the root is α. We begin by drawing the first symbol, X_1, such that the probability of $X_1 = x$ is given by the weight of x (which is 1) divided by the weight of the current node, which is the root (i.e., a weight of α, as said). In other words, we randomly select X_1 according to the uniform distribution over $\mathcal{X}$. The leaf corresponding to the outcome of X_1, call it x_1, will now become an internal node by adding to the tree its α off-springs, thus growing the tree to have $2\alpha - 1$ leaves. Each one of the leaves of the extended tree has now weight 1, and then, the weight of the their common ancestor (formerly, the leaf of x_1), becomes the sum of their weights, namely, α, and similarly, the weights of all ancestors of x_1, all the way up to the root, are now sequentially updated to become the sum of the weights of their immediate off-springs.

We now start again from the root of the tree to randomly draw the next symbol, X_2, such that the probability that $X_2 = x$, is given by the weight of the node x divided by the weight of the current node, which is again the root, and then we move from the root to its corresponding off-spring pertaining to X_2, that was just randomly drawn. If we have reached a leaf, then again this leaf gives birth to α new off-springs, each assigned with weight 1, then all corresponding weights are updated as described before, and finally, we move back to the root, etc. If we are still in an internal node, then again, we draw the next symbol according to the ratio between the weight of the node pertaining to the next symbol and the weight of the current node, and so on. The process continues until n symbols, $X_1, X_2, \ldots, X_n$ have been generated.

Note that every time we restart from the root and move along the tree until we reach a leaf, we generate a new LZ phrase that has not been obtained before. Let $c(\boldsymbol{x})$ be the number of phrases generated. Along each path from the root to a leaf, we implement a telescopic product of conditional probabilities, where the numerator pertaining to the last conditional probability is the weight of the leaf, which is 1, and the denominator of the first probability is the total number of leaves after i rounds, which is $\alpha + i(\alpha - 1)$ (because after every birth of a new generation of leaves, the total number of leaves is increased by $\alpha - 1$). All other numerators and denominators of the conditional probabilities along the path cancel each other telescopically. The result is that the induced probability distribution along the various leaves is uniform. Precisely, after i phrases have been generated, the probability of each leaf is exactly $1/[\alpha + i(\alpha - 1)]$. Therefore,

$$\begin{aligned} P(\boldsymbol{x}) &= \prod_{i=0}^{c(\boldsymbol{x})-1} \frac{1}{\alpha + i(\alpha-1)} \\ &\geq \prod_{i=0}^{c(\boldsymbol{x})-1} \frac{1}{\alpha + [c(\boldsymbol{x})-1](\alpha-1)} \\ &= \frac{1}{\{\alpha + [c(\boldsymbol{x})-1](\alpha-1)\}^{c(\boldsymbol{x})}} \\ &= 2^{-c(\boldsymbol{x})\log\{[c(\boldsymbol{x})-1](\alpha-1)+\alpha\}}, \end{aligned} \tag{32}$$

which is of the exponential order of $2^{-c(\boldsymbol{x})\log c(\boldsymbol{x})}$.

Second algorithm. As said, the second method for efficiently generating random guesses according to the LZ distribution is based on the simple idea of feeding purely random bits into the LZ decoder until a decoded sequence of length n is obtained. To describe it, we refer to the coding scheme proposed in [27, Theorem 2], but with a slight modification. Recall that according to this coding scheme, for the i–th parsed phrase, $\boldsymbol{x}_{n_{j-1}+1}^{n_j}$, one encodes two integers: the index $0 \leq \pi(j) \leq j-1$ of the matching past phrase and the index the additional source symbol, $0 \leq I_A(x_{n_j}) \leq \alpha - 1$. These two integers are mapped together bijectively into one integer, $I(\boldsymbol{x}_{n_{j-1}+1}^{n_j}) = \pi(j) \cdot \alpha + I_A(x_{n_j})$, which takes on values in the set $\{0, 1, 2, \ldots, j\alpha - 1\}$, and so, according to [27], it can be encoded using $L_j = \lceil \log(j\alpha) \rceil$

bits. Here, instead, we will encode $I(\boldsymbol{x}_{n_{j-1}+1}^{n_j})$ with a tiny modification that will make the encoding equivalent to a walk on a *complete binary tree* [7] from the root to a leaf. Considering the fact that (by definition of L_j), $2^{L_j-1} < j\alpha \le 2^{L_j}$, we first construct a full binary tree with 2^{L_j-1} leaves at depth $L_j - 1$, and then convert $j\alpha - 2^{L_j-1}$ leaves to internal nodes by generating their off-springs. The resulting complete binary tree will then have exactly $j\alpha$ leaves, some of them at depth $L_j - 1$ and some - at depth L_j. Each leaf of this tree will now correspond to one value of $I(\boldsymbol{x}_{n_{j-1}+1}^{n_j})$, and hence to a certain decoded phrase. Let $\hat{L}_j$ denote the length of the codeword for $I(\boldsymbol{x}_{n_{j-1}+1}^{n_j})$. Obviously, either $\hat{L}_j = L_j - 1$ or $\hat{L}_j = L_j$. Consider now what happens if we feed the decoder of this encoder by a sequence of purely random bits (generated by a binary symmetric source): every leaf at depth $\hat{L}_j$ will be obtained with probability $2^{-\hat{L}_j}$, and since the tree is complete, these probabilities sum up to unity. The probability of obtaining $\boldsymbol{x}$ at the decoder output is, therefore, equal to the probability of the sequence of bits pertaining to its compressed form, namely,

$$\begin{aligned}
\tilde{P}(\boldsymbol{x}) &= \prod_{j=1}^{c(\boldsymbol{x})+1} 2^{-\hat{L}_j} \\
&= \exp_2\left\{-\sum_{j=1}^{c(\boldsymbol{x})+1} \hat{L}_j\right\} \\
&\ge \exp_2\left\{-\sum_{j=1}^{c(\boldsymbol{x})+1} L_j\right\} \\
&\ge \exp_2\left\{-\sum_{j=1}^{c(\boldsymbol{x})+1} \log(2j\alpha)\right\} \\
&\ge \exp_2\left\{-[c(\boldsymbol{x})+1]\log[2c(\boldsymbol{x})\alpha]\right\} \\
&= \exp_2\{-[c(\boldsymbol{x})+1]\log c(\boldsymbol{x}) - [c(\boldsymbol{x})+1]\log(2\alpha)\}, \qquad (33)
\end{aligned}$$

which is again of the exponential order of $2^{-c(\boldsymbol{x})\log c(\boldsymbol{x})}$.

5.4 Side Information

As we have done at the end of Section 4, here too, we describe how our results extend to the case where the guesser is equipped with SI. The parts that extend straightforwardly will be described briefly, whereas the parts whose extension is non–trivial will be more detailed.

[7] By "complete binary tree", we mean a binary tree where each node is either a leaf or has two off-springs. The reason for the need of a complete binary tree is that for the algorithm to be valid, every possible sequence of randomly chosen bits must be a legitimate compressed bit-stream so that it would be decodable by the LZ decoder.

Consider the pair process $\{(X_t, Y_t)\}$, jointly distributed according to a hidden Markov model,

$$P(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{z}} \prod_{t=1}^{n} P(x_t, y_t, z_{t+1}|z_t), \tag{34}$$

where, as before, z_t is the state at time t, taking on values in a finite set of states $\mathcal{Z}$ of cardinality s.

Here, our objective is to guess $\boldsymbol{x}$ when $\boldsymbol{y}$ is available to the guesser as SI. Most of our earlier results extend quite easily to this case. Basically, the only modification needed is to replace the LZ complexity of $\boldsymbol{x}$ by the conditional LZ complexity of $\boldsymbol{x}$ given $\boldsymbol{y}$, which is defined as in [49] and [50]. In particular, consider the joint parsing of the sequence of pairs, $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, let $c(\boldsymbol{x}, \boldsymbol{y})$ denote the number of phrases, $c(\boldsymbol{y})$ – the number of distinct $\boldsymbol{y}$-phrases, $\boldsymbol{y}(\ell)$ – the ℓ-th distinct $\boldsymbol{y}$-phrase, $1 \leq \ell \leq c(\boldsymbol{y})$, and finally, let $c_\ell(\boldsymbol{x}|\boldsymbol{y})$ denote the number of times $\boldsymbol{y}(\ell)$ appears as a phrase, or, equivalently, the number of distinct $\boldsymbol{x}$-phrases that appear jointly with $\boldsymbol{y}(\ell)$, so that $\sum_{\ell=1}^{c(\boldsymbol{y})} c_\ell(\boldsymbol{x}|\boldsymbol{y}) = c(\boldsymbol{x}, \boldsymbol{y})$. Then, we define

$$u(\boldsymbol{x}|\boldsymbol{y}) = \sum_{\ell=1}^{c(\boldsymbol{y})} c_\ell(\boldsymbol{x}|\boldsymbol{y}) \log c_\ell(\boldsymbol{x}|\boldsymbol{y}). \tag{35}$$

For the converse theorem (lower bound), the proof is the same as the proof of Theorem 2, except that here, we need a lower bound on the size of a "conditional type" of $\boldsymbol{x}$ given $\boldsymbol{y}$. This lower bound turns to be of the exponential order of $2^{u(\boldsymbol{x}|\boldsymbol{y})}$, as can be seen in [51, Lemma 1]. Thus, the lower bound on the guessing moment is of the exponential order of $\boldsymbol{E}[\exp_2\{\rho u(\boldsymbol{X}|\boldsymbol{Y})\}]$.

For the direct theorem (upper bound), we can either create a deterministic guessing list by ordering the members of $\mathcal{X}^n$ according to increasing order of their conditional LZ code–length function values, $LZ(\boldsymbol{x}|\boldsymbol{y}) \approx u(\boldsymbol{x}|\boldsymbol{y})$, [49, p. 2617], [50, page 460, proof of Lemma 2], or randomly draw guesses according to

$$\tilde{P}(\boldsymbol{x}|\boldsymbol{y}) = \frac{2^{-LZ(\boldsymbol{x}|\boldsymbol{y})}}{\sum_{\boldsymbol{x}'} 2^{-LZ(\boldsymbol{x}'|\boldsymbol{y})}}. \tag{36}$$

Following Subsection 5.C, we wish to have an efficient algorithm for sampling from the distribution (36), or, more generally, for implementing a conditional distribution that satisfies $\tilde{P}(\boldsymbol{x}|\boldsymbol{y}) \stackrel{\cdot}{\geq} 2^{-LZ(\boldsymbol{x}|\boldsymbol{y})} \stackrel{\cdot}{=} 2^{-u(\boldsymbol{x}|\boldsymbol{y})}$.

While we have not been able to find an extension of the first algorithm of Subsection 5.C to the case of SI, the second algorithm therein turns out to lend itself fairly easily to such an extension. Once again, generally speaking, the idea is to feed a sequence of purely random bits as inputs to the decoder pertaining to the conditional LZ decoder, equipped with $\boldsymbol{y}$ as SI, and wait until exactly n symbols, $x_1, \ldots, x_n$, have been obtained at the output of the decoder. We need, however, a few slight modifications in conditional LZ code, in order to ensure that any sequence of randomly drawn bits would be legitimate as the output of the encoder, and hence be also decodable by the decoder. Once again, to this end, we must use complete binary trees for the prefix codes for the various components of the conditional LZ code.

As can be seen in [49], [50], the conditional LZ compression algorithm sequentially encodes $\boldsymbol{x}$ phrase by phrase, where the code for each phrase consists of three parts:

1. A code for the length of the phrase, $L[\boldsymbol{y}(\ell)]$.
2. A code for the location of the matching $\boldsymbol{x}$–phrase among all previous phrases with the same $\boldsymbol{y}$–phrase.
3. A code for the index of the last symbol of the $\boldsymbol{x}$–phrase among all members of $\mathcal{X}$.

Parts 2 and 3 are similar to those of the ordinary LZ algorithm and they in fact can even be united, as described before, into a single code for both indices (although this is not necessary). Part 1 requires a code for the integers, which can be implemented by the Elias code, as described in [49]. However, for the sake of conceptual simplicity of describing the required complete binary tree, consider the following alternative option. Define the following distribution on the natural numbers,

$$Q(i) = \frac{6}{\pi^2 i^2}, \qquad i = 1, 2, 3, \ldots \tag{37}$$

and construct a prefix tree for the corresponding Shannon code, whose length function is given by

$$\mathcal{L}(i) = \lceil -\log Q(i) \rceil. \tag{38}$$

Next prune the tree by eliminating all leaves that correspond to values of $i = L[\boldsymbol{y}(\ell)]$ that cannot be obtained at the current phrase: the length $L[\boldsymbol{y}(\ell)]$ cannot be larger than the maximum possible phrase length and cannot correspond to a string that has not been obtained as a $\boldsymbol{y}$–phrase before.[8] Finally, shorten the tree by eliminating branches that

[8]This is doable since both the encoder and the decoder have this information at the beginning of the current phrase.

emanate from any node that has one off-spring only. At the end of this process, we have a complete binary tree where the resulting code length for every possible value of $L[\boldsymbol{y}(\ell)]$ cannot be larger than its original value (38).

The probability of obtaining a given $\boldsymbol{x}$ at the output of the above–described conditional LZ decoder is equal to the probability of randomly selecting the bit-stream that generates $\boldsymbol{x}$ (in the presence of $\boldsymbol{y}$ as SI), as the response to this bit-stream. Thus,

$$\begin{aligned}
\tilde{P}(\boldsymbol{x}|\boldsymbol{y}) &\geq \prod_{\ell=1}^{c(\boldsymbol{y})}\prod_{j=1}^{c_\ell(\boldsymbol{x}|\boldsymbol{y})} \exp_2\{-\lceil\log(j\alpha)\rceil - \mathcal{L}(L[\boldsymbol{y}(\ell)])\} \\
&\geq \exp_2\left\{-\sum_{\ell=1}^{c(\boldsymbol{y})}\sum_{j=1}^{c_\ell(\boldsymbol{x}|\boldsymbol{y})}\left[\log(2j\alpha) + 2\log L[\boldsymbol{y}(\ell)] + \log\frac{\pi^2}{6} + 1\right]\right\} \\
&\geq \exp_2\left\{-\sum_{\ell=1}^{c(\boldsymbol{y})} c_\ell(\boldsymbol{x}|\boldsymbol{y})\log[2\alpha c_\ell(\boldsymbol{x}|\boldsymbol{y})] - 2\sum_{\ell=1}^{c(\boldsymbol{y})} c_\ell(\boldsymbol{x}|\boldsymbol{y})\log L[\boldsymbol{y}(\ell)] - \right. \\
&\qquad \left. c(\boldsymbol{x},\boldsymbol{y})\left[\log\frac{\pi^2}{6}+1\right]\right\} \\
&\doteq \exp_2\{-u(\boldsymbol{x}|\boldsymbol{y})\},
\end{aligned} \tag{39}$$

where the last step follows from the observation [50, p. 460] that

$$\begin{aligned}
\sum_{\ell=1}^{c(\boldsymbol{y})} c_\ell(\boldsymbol{x}|\boldsymbol{y})\log L[\boldsymbol{y}(\ell)] &= c(\boldsymbol{x},\boldsymbol{y})\sum_{\ell=1}^{c(\boldsymbol{y})}\frac{c_\ell(\boldsymbol{x}|\boldsymbol{y})}{c(\boldsymbol{x},\boldsymbol{y})}\log L[\boldsymbol{y}(\ell)] \\
&\leq c(\boldsymbol{x},\boldsymbol{y})\log\left[\frac{\sum_{\ell=1}^{c(\boldsymbol{y})} c_\ell(\boldsymbol{x}|\boldsymbol{y})L[\boldsymbol{y}(\ell)]}{c(\boldsymbol{x},\boldsymbol{y})}\right] \\
&= c(\boldsymbol{x},\boldsymbol{y})\log\frac{n}{c(\boldsymbol{x},\boldsymbol{y})},
\end{aligned} \tag{40}$$

and the fact that $c(\boldsymbol{x},\boldsymbol{y})$ cannot be larger than $O(n/\log n)$ [27].

6 Conclusion

In this work, we studied the guesswork problem under a very general setup of unknown source distribution and decentralized operation. Specifically, we designed and analyzed guessing strategies which do not require the source distribution, the exact guesswork moment to be optimized, or any synchronization between the guesses, yet achieve the optimal guesswork exponent as if all this information was known and full synchronization was possible. Furthermore, we designed efficient algorithms in order to sample guesses from the universal distributions suggested. We believe such sampling methods may be interesting on their own, and find applications outside the guesswork regime.

Appendix

Proof of Lemma 1. We denote

$$S = \sum_{k=1}^{\infty} k^{\rho}(1-e^{-na})^{k-1}. \tag{A.1}$$

For a given, arbitrarily small $\epsilon > 0$, we first decompose S as follows.

$$S = \sum_{k=1}^{e^{n(a+\epsilon)}} k^{\rho}(1-e^{-na})^{k-1} + \sum_{k=e^{n(a+\epsilon)}+1}^{\infty} k^{\rho}(1-e^{-na})^{k-1} \stackrel{\triangle}{=} A + B. \tag{A.2}$$

Now,

$$\begin{aligned}
A &\leq \sum_{k=1}^{e^{n(a+\epsilon)}} e^{n(a+\epsilon)\rho}(1-e^{-na})^{k-1} \\
&= e^{n(a+\epsilon)\rho} \sum_{k=1}^{e^{n(a+\epsilon)}} (1-e^{-na})^{k-1} \\
&\leq e^{n(a+\epsilon)\rho} \sum_{k=1}^{\infty} (1-e^{-na})^{k-1} \\
&= e^{n(a+\epsilon)\rho} \cdot \frac{1}{1-(1-e^{-na})} \\
&= e^{na} \cdot e^{n(a+\epsilon)\rho} \\
&\leq e^{n(1+\rho)(a+\epsilon)}.
\end{aligned} \tag{A.3}$$

It remains to show then that B has a negligible contribution for large enough n. Indeed, we next show that B decays double–exponentially rapidly in n for every $\epsilon > 0$.

$$\begin{aligned}
B &= \sum_{k=e^{n(a+\epsilon)}+1}^{\infty} k^{\rho}(1-e^{-na})^{k-1} \\
&= \sum_{k=e^{n(a+\epsilon)}+1}^{\infty} \exp\{(k-1)\ln(1-e^{-na}) + \rho \ln k\} \\
&= \sum_{k=e^{n(a+\epsilon)}}^{\infty} \exp\{k \ln(1-e^{-na}) + \rho \ln(k+1)\} \\
&\leq \sum_{k=e^{n(a+\epsilon)}}^{\infty} \exp\{-k \cdot e^{-na} + \rho \ln(k+1)\} \\
&= \sum_{k=e^{n(a+\epsilon)}}^{\infty} \exp\left\{-k \cdot \left[e^{-na} - \frac{\rho \ln(k+1)}{k}\right]\right\}
\end{aligned} \tag{A.4}$$

Since $\{[\ln(k+1)]/k\}_{k\geq 1}$ is a monotonically decreasing sequence, then for all $k \geq e^{n(a+\epsilon)}$,

$$\frac{\rho \ln(k+1)}{k} \leq \frac{\rho \ln[e^{n(a+\epsilon)}+1]}{e^{n(a+\epsilon)}} = \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)}+1].$$

Thus,

$$\begin{aligned}
B &\leq \sum_{k=e^{n(a+\epsilon)}}^{\infty} \exp\{-k \cdot (e^{-na} - \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)} + 1])\} \\
&= \frac{\exp\{-e^{n(a+\epsilon)}(e^{-na} - \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)} + 1])\}}{1 - \exp\{-(e^{-na} - \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)} + 1])\}} \\
&= \frac{\exp\{-(e^{n\epsilon} - \rho \ln[e^{n(a+\epsilon)} + 1])\}}{1 - \exp\{-(e^{-na} - \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)} + 1])\}} \\
&= \frac{[e^{n(a+\epsilon)} + 1]^{\rho} \exp\{-e^{n\epsilon}\}}{1 - \exp\{-(e^{-na} - \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)} + 1])\}} \qquad \text{(A.5)}
\end{aligned}$$

Now, for small x, we have $1 - e^{-x} = x + O(x^2)$, and so, the factor

$$[1 - \exp\{-(e^{-na} - \rho e^{-n(a+\epsilon)} \ln[e^{n(a+\epsilon)} + 1])\}]^{-1}$$

is of the exponential order of e^{na}, which does not affect the double exponential decay due to the term $e^{-e^{n\epsilon}}$. The proof of the lemma is completed by taking into account the arbitrariness of $\epsilon > 0$ (in particular, one may let ϵ decay sufficiently slowly with n).

References

[1] J. M. Wozencraft, "Sequential decoding for reliable communication," Ph.D. dissertation, Research Laboratory of Electronics, Massachusetts Institute of Technology, May 1957.

[2] E. Arikan, "An inequality on guessing and its application to sequential decoding," *IEEE Transactions on Information Theory*, vol. 42, no. 1, pp. 99–105, January 1996.

[3] C. E. Pfister and W. G. Sullivan, "Rényi entropy, guesswork moments, and large deviations," *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2794–2800, November 2004.

[4] W. Szpankowski and S. Verdú, "Minimum expected length of fixed-to-variable lossless compression without prefix constraints," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4017–4025, July 2011.

[5] I. Kontoyiannis and S. Verdú, "Optimal lossless data compression: Non-asymptotics and asymptotics," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 777–795, February 2014.

[6] O. Kosut and L. Sankar, “Asymptotics and non-asymptotics for universal fixed-to-variable source coding,” *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3757–3772, June 2017.

[7] M. Bishop and D. V. Klein, “Improving system security via proactive password checking,” *Computers & Security*, vol. 14, no. 3, pp. 233–249, 1995.

[8] M. D. Amico, P. Michiardi, and Y. Roudier, “Password strength: An empirical analysis,” in *Proceedings IEEE Infocom*, March 2010, pp. 1–9.

[9] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, “Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms,” in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 523–537.

[10] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, “Of passwords and people: Measuring the effect of password-composition policies,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’11. New York, NY, USA: ACM, 2011, pp. 2595–2604.

[11] J. Bonneau, “The science of guessing: Analyzing an anonymized corpus of 70 million passwords,” in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 538–552.

[12] J. O. Pliam, “On the incomparability of entropy and marginal guesswork in brute-force attacks,” in *Progress in Cryptology —INDOCRYPT 2000*, B. Roy and E. Okamoto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 67–79.

[13] P. Gauravaram, “Security analysis of salt——password hashes,” in *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, November 2012, pp. 25–30.

[14] A. Rezaee, A. Beirami, A. Makhdoumi, M. Médard, and K. Duffy, “Guesswork subject to a total entropy budget,” in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, October 2017, pp. 1008–1015.

[15] M. M. Christiansen, K. R. Duffy, F. du Pin Calmon, and M. Médard, “Guessing a password over a wireless channel (on the effect of noise non-uniformity),” in *2013 Asilomar Conference on Signals, Systems and Computers*, November 2013, pp. 51–55.

[16] M. M. Christiansen and K. R. Duffy, "Guesswork, large deviations, and shannon entropy," *IEEE Transactions Information Theory*, vol. 59, no. 2, pp. 796–802, February 2013.

[17] Y. Yona and S. Diggavi, "The effect of bias on the guesswork of hash functions," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 2248–2252.

[18] S. Salamatian, A. Beirami, A. Cohen, and M. Médard, "Centralized vs. decentralized multi-agent guesswork," in *Information Theory (ISIT), 2017 IEEE International Symposium on.* IEEE, 2017, pp. 2258–2262.

[19] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen, and M. Médard, "Why botnets work: Distributed brute-force attacks need no synchronization," *in revision, IEEE Transactions on Information Forensics and Security*, 2018.

[20] T. A. Courtade and S. Verdú, "Variable-length lossy compression and channel coding: Non-asymptotic converses via cumulant generating functions," in *2014 IEEE International Symposium on Information Theory*, June 2014, pp. 2499–2503.

[21] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proceedings of the 21st international conference on World Wide Web.* ACM, 2012, pp. 301–310.

[22] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: empirical results," *IEEE Security Privacy*, vol. 2, no. 5, pp. 25–31, September 2004.

[23] D. Vishwakarma and C. E. V. Madhavan, "Efficient dictionary for salted password analysis," in *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, January 2014, pp. 1–6.

[24] R. Sundaresan, "Guessing under source uncertainty with side information," in *2006 IEEE International Symposium on Information Theory*, July 2006, pp. 2438–2440.

[25] J. Owens and J. Matthews, "A study of passwords and methods used in brute-force SSH attacks," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[26] E. Tirado, B. Turpin, C. Beltz, P. Roshon, R. Judge, and K. Gagneja, “A new distributed brute-force password cracking technique,” in *International Conference on Future Network Systems and Security.* Springer, 2018, pp. 117–127.

[27] J. Ziv and A. Lempel, “Compression of individual sequences via variable-rate coding,” *IEEE transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, September 1978.

[28] J. L. Massey, “Guessing and entropy,” in *Proceedings of IEEE International Symposium on Information Theory*, 1994, p. 204.

[29] E. Arikan and N. Merhav, “Guessing subject to distortion,” *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 1041–1056, May 1998.

[30] D. Malone and W. G. Sullivan, “Guesswork and entropy,” *IEEE Transactions on Information Theory*, vol. 50, no. 3, pp. 525–526, March 2004.

[31] M. K. Hanawal and R. Sundaresan, “Guessing revisited: A large deviations approach,” *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 70–78, January 2011.

[32] R. Sundaresan, “Guessing under source uncertainty,” *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 269–287, January 2007.

[33] M. M. Christiansen, K. R. Duffy, F. du Pin Calmon, and M. Médard, “Multi-user guesswork and brute force security,” *IEEE Transactions on Information Theory*, vol. 61, no. 12, pp. 6876–6886, December 2015.

[34] A. Beirami, R. Calderbank, K. Duffy, and M. Médard, “Quantifying computational security subject to source constraints, guesswork and inscrutability,” in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 2757–2761.

[35] M. J. Weinberger, J. Ziv, and A. Lempel, “On the optimal asymptotic performance of universal ordering and of discrimination of individual sequences,” *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 380–385, March 1992.

[36] A. Beirami, R. Calderbank, M. Christiansen, K. Duffy, A. Makhdoumi, and M. Mdard, “A geometric perspective on guesswork,” in *2005 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, September 2015, pp. 941–948.

[37] M. K. Hanawal and R. Sundaresan, "Randomised attacks on passwords," *DRDO-IISc Programme on Advanced Research in Mathematical Engineering*, 2010.

[38] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 145–160.

[39] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A security analysis of honeywords," https://tinyurl.com/y87ffmny, NDSS, 2018.

[40] A. Bracher, E. Hof, and A. Lapidoth, "Guessing attacks on distributed-storage systems," *arXiv preprint arXiv:1701.01981*, 2017.

[41] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *2009 30th IEEE Symposium on Security and Privacy*, May 2009, pp. 391–405.

[42] I. Csiszár and J. Körner, *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

[43] R. Krichevsky and V. Trofimov, "The performance of universal encoding," *IEEE Transactions on Information Theory*, vol. 27, no. 2, pp. 199–207, March 1981.

[44] Y. Ephraim and N. Merhav, "Hidden markov processes," *IEEE Transactions on information theory*, vol. 48, no. 6, pp. 1518–1569, June 2002.

[45] T. M. Cover and J. A. Thomas, *Elements of information theory*, 2nd ed. New York, NY, USA: John Wiley & Sons, 2006.

[46] N. Merhav, "Universal coding with minimum probability of codeword length overflow," *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 556–563, May 1991.

[47] M. Feder, "Gambling using a finite state machine," *IEEE Transactions on Information Theory*, vol. 37, no. 5, pp. 1459–1465, September 1991.

[48] M. Feder, N. Merhav, and M. Gutman, "Universal prediction of individual sequences," *IEEE transactions on Information Theory*, vol. 38, no. 4, pp. 1258–1270, July 1992.

[49] T. Uyematsu and S. Kuzuoka, “Conditional lempel-ziv complexity and its application to source coding theorem with side information,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 86, no. 10, pp. 2615–2617, 2003.

[50] J. Ziv, “Universal decoding for finite-state channels,” *IEEE Transactions on Information Theory*, vol. IT-31, no. 4, pp. 453–460, July 1985.

[51] N. Merhav, “Universal detection of messages via finite-state channels,” *IEEE Transactions on Information Theory*, vol. 46, no. 6, pp. 2242–2246, April 2000.